Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

MODULO MICROCOMPUTADOR PARA LA IMPLEMENTACION DE CONTROLADORES INDUSTRIALES

T E S I S

Que para obtener el Título de
INGENIERO EN COMPUTACION
P R E S E N T A

ALBERTO ADUÑA GRAJEDA

Director: ING. JUAN B. MARTINEZ GARCIA

México, D. F. 1988

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

1) INTRODUCCION

1.1) AUTOMATICA Y SOCIEDAD.

Desde la mañana, al levantarse, el hombre está rodeado por la automática: lo despierta un reloj despertador que suena sólo a la hora exacta; el agua al baño la envía una estación de bombeo que opera sola; la temperatura del agua caliente la mantiene, de manera autónoma, un dispositivo especial; los productos frescos para el desayuno los conserva un refrigerador cuya temperatura es regualda por él mismo; en el camino al trabajo el tráfico lo regulan los semáforos sin que requieran supervisión humana, ... nuestra forma de vida actual sería imposible sin la automática.

Como es conocido, el reloj, el primer autómata creado para fines prácticos, sugirió al hombre la idea de aplicar los autómatas en la producción. De esta manera, primero con la fuerza del vapor y después con la de la electricidad, hicieron su aparición en las fábricas, iniciando una verdadera revolución en todos los ámbitos de la sociedad: la primera Revolución Industrial.

La automática se convirtió en la tendencia principal en el desarrollo de la técnica. El arribo de la electrónica permitió tener dispositivos de control más rápidos, compactos y flexibles. Posteriormente, con el desarrollo de la cibernética, los autómatas incorporaron una característica más: la capacidad de almacenar y procesar Información.

La introducción en la producción de máquinas y dispositivos automáticos, de medios de control y de dirección se denomina automatización. Conduce a la sustitución del hombre por la máquina en la realización de determinadas tareas, especialmente las relativas al control y regulación. Su finalidad es lograr el mejor modo de utilizar al hombre y la máquina para conseguir un producto con el menor gasto de material y capital.

El problema básico de la automática es el control. Se requiere cuidar, por ejemplo, de las dimensiones de las piezas, la temperatura, la presión, el color del material, la concentración de soluciones, etc. La implementación del controlador puede realizarse con dispositivos mecánicos, eléctricos, neumáticos, etc. En aplicaciones Industriales, por las ventajas que ofrece, la implementación actual está basada en dispositivos electrónicos. Hasta hace poco, la electrónica analógica era la que imperaba en este campo. Pero hoy, con el gran desarrollo que ha tenido y sigue teniendo la tecnología digital, ésta va ganando cada vez más terreno sobre la tecnología analógica en muchas aplicaciones, incluyendo la de los controladores.

Como las variables de un proceso cambian mucho más lentamente que las operaciones de microsegundos de una computadora, entonces una microcomputadora con alguna circuitería de entrada y salida simple, puede desarrollar las funciones de la circuitería analógica equivalente para implementar un sistema de control. Los dispositivos de control de procesos basados en microcomputadora se denominan Controladores Pro-

gramables, ya que adquieren esta última característica que los hace sumamente versátiles.

1.2) ANTECEDENTES Y DEFINICION DEL PROYECTO

En el diseño actual de sistemas se sigue el enfoque de la modularidad: se diseñan módulos completamente funcionales para, en base a ellos, crear diversos sistemas, más rápida y económicamente. La modularidad permite diseñar por separado cada uno de los bloques que se requieren para implementar el sistema de control.

Este ha sido el enfoque seguido en la Coordinación de Automatización del Instituto de Ingeniería de la UNAM para el desarrollo de diversos sistemas electrónicos de control digital.

En 1984 se diseñó ahí un controlador programable, basado en una microcomputadora a la que se denominó PAT86, la cual usaba el microprocesador Z80. El PAT86 ocupaba una sola tarjeta y se programaba a través de una terminal de video, directamente en lenguaje máquina del microprocesador Z80, o bien usando un lenguaje Basic particular que se le adaptó.

El PAT86 se empleó durante 1985, 1986 y 1987 en el desarrollo de varios proyectos dentro de la misma Coordinación de Automatización, tales como sistemas remotos de adquisición de datos, control de pozos municipales, concentradores de datos, entre otras aplicaciones.

El PAT86 es aún un sistema vigente en muchas aplica-

ciones, principalmente considerando el costo (el Z80 es un microprocesador barato y fácil de conseguir en nuestro país). Sin embargo, la Coordinación de Automatización planeó actualizar su diseño incorporando un microprocesador más actual, sobre todo previendo a futuro la evolución hacia sistemas más complejos. De esta manera surgen los objetivos del presente proyecto, en el cual se busca la actualización del equipo PAT86. Estos objetivos se describen a continuación.

OBJETIVOS:

a) El diseño y construcción de una microcomputadora en una sola tarjeta, a la que se denominará SUPERPAT, pensada con un enfoque modular y orientada hacia aplicaciones de control industrial; esto es, será un módulo completamente funcional que permita, conectando módulos adicionales según la aplicación particular, implementar diversos sistemas electrónicos de control industrial. El software del SUPERPAT permitirá la programación del sistema a través de un lenguaje Basic particular.

b) Incorporar todo el soporte necesario -apoyados en la utilización de una computadora IBM PC o compatibles- para agregar facilidades al Sistema SUPERPAT, de manera que se obtenga todo un sistema de desarrollo completo para la implementación de sistemas electrónicos de control industrial.

2) LAS MICROCOMPUTADORAS EN EL CONTROL INDUSTRIAL

2.1) MICROPROCESADORES

El inicio de los 70's marcó el inicio de una revolución en el mundo de la electrónica: la revolución del microprocesador.

En 1971, la Compañía Intel había creado un nuevo circuito integrado en base a tecnología de transistores de efecto de campo (FET), el 4004, "anunciando una nueva era en la electrónica integrada" [referencia 1]. Pensado originalmente para funcionar como un controlador complejo, tenía capacidades para funcionar como una Unidad Central de Proceso (CPU) en una computadora. Los diseñadores del chip tenían duda de si alguien querría usar su controlador FET como computadora, cuando ya había disponibles minicomputadoras bipolares más avanzadas. Pero el nuevo chip tenía una ventaja sobre los CPUs ya existentes: su ECONOMIA. Este chip, al que más tarde se denominó microprocesador, permitía tener una capacidad real de cómputo disponible a cualquiera a un precio razonable. Por esta característica, además del amplio soporte y difusión que le dió Intel, el microprocesador logró tener una amplia aceptación, no como competidor en el mercado de las minicomputadoras, sino creando uno nuevo: el de las microcomputadoras.

Como su nombre lo indica, las microcomputadoras son computadoras pequeñas cuyo CPU lo constituye un microprocesador.

Las microcomputadoras se han utilizado en el campo tradicional de la computación, es decir, como máquinas programables de propósito general, en aquellas aplicaciones donde el volumen de datos no requiera de computadoras mayores. La computadora de escritorio - un sueño de antaño- fué de esta manera factible.

Pero además, debido a los bajos costos de utilizar microprocesadores (y chips VLSI en general), ha sido posible introducir en muchos sistemas digitales la capacidad de cómputo. Los sistemas digitales basados en microprocesador son microcomputadoras -de propósito específico- por la arquitectura que adquieren. Por tanto, estos sistemas son programables, lo que los hace fácilmente reconfigurables y capaces de realizar tareas complejas paso a paso, con una solución a las diversas aplicaciones por medio de desarrollo de software.

Los microprocesadores han llevado efectivamente a la electrónica a una nueva era. En diez años el dispositivo tuvo una penetración mas profunda en la sociedad que la de las mainframes en sus primeros veinte años. Más aún, provocó que los fabricantes de componentes y los usuarios finales repensaran el rol de la computadora. Lo que había sido una vez una máquina gigante, atendida por especialistas en un cuarto para ella sola, era ahora un circuito delgado, transparente para el usuario de automóviles, instrumentos, equipo de oficina, y una larga lista de otros productos. El microprocesador se ha incorporado en una variedad de productos ya existentes en el mercado y ha creado otros nuevos.

2.2) EL SISTEMA DE CONTROL DE PROCESOS

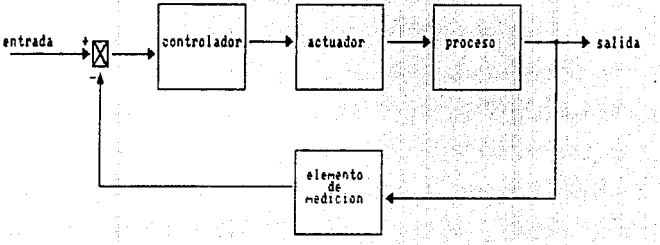
En el control de procesos intervienen tres entidades: la variable a controlar (cualquier cantidad física que pueda variar), el sistema de medición y el controlador. El proceso es cualquier sistema (físico, químico, etc.) que se vaya a controlar.

El sistema de control involucra primero un sistema de medición de variables, tales como velocidad de un motor, temperatura, flujo de reactantes, el nivel de un líquido en un tanque, etc.

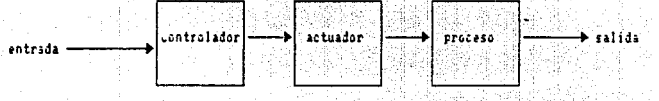
La tarea del controlador es ajustar el estado del proceso, manteniendo cada variable (V) tan cerca como sea posible de un valor predeterminado llamado "set point" (SP). La diferencia entre el valor actual de V y SP se denomina error (E):

$$E = V - SP$$

Un sistema de control de lazo cerrado es aquel en que la señal de salida tiene efecto directo sobre la acción de control. Para ello hace uso de un mecanismo conocido como realimentación, que consiste en comparar la salida con la entrada y utilizar la diferencia entre ambas (el error E) como parámetro de control. En la figura 1a se muestra el diagrama de bloques de un sistema de este tipo. Ahí se observa que el controlador, usando la información de error, actúa sobre el proceso a través de otro componente: el "elemento final de control" o "actuador", para cambiar el valor de la variable en la dirección deseada (hacia el set-point) y así reducir el error.



a) de lazo cerrado



b) de lazo abierto

Fig 1 - Diagrama de bloques de un sistema de control de procesos

Los sistemas de control de lazo abierto son aquéllos en los que la salida no tiene efecto sobre la acción de control, es decir, la salida ni se mide ni se realimenta para comparación con la entrada. En la figura 1b se muestra el diagrama de bloques de tal sistema. En la práctica, sólo se puede usar el control de lazo abierto si la relación entre la entrada y la salida se conoce. Debido a esto, en la mayoría de las aplicaciones se usa el control de lazo cerrado.

Si se usa un controlador automático (algún dispositivo reemplaza al operador humano), entonces el sistema de control se vuelve automático, lo que elimina cualquier error humano de operación. Existen numerosos ejemplos de sistemas de este tipo: los calentadores de agua, los sistemas de calefacción hogareña, etc.

En un sistema de control automático, el controlador tiene la función de compensar tan rápida y exactamente como sea posible cualquier cambio en el sistema que se está controlando. La manera como el controlador opera (su respuesta a la señal de error) se llama su "modo de control" o "algoritmo de control", lo que indica un método de cálculo para producir una salida de control.

En los controles automáticos industriales, son muy comunes los siguientes seis algoritmos de control: on-off, proporcional, integral, proporcional e integral, proporcional y derivativo, y proporcional-integral-derivativo. Un sistema se puede ajustar para óptima respuesta usando alguno de estos modos de control.

En aplicaciones industriales la implementación actual

del controlador está basada en dispositivos electrónicos digitales, concretamente, en microprocesadores, lo que conduce a los controladores programables.

En este tipo de sistemas, el control on-off es ampliamente usado. En el control on-off, el elemento actuador tiene solamente dos posiciones fijas (que en muchos casos son simplemente encendido y apagado). Como los dispositivos digitales manejan información en forma binaria (dos estados), sus salidas hacia el actuador son directas. Esto hace al control on-off relativamente simple y económico, y no requiere del modelado y uso de las ecuaciones del sistema a controlar. Por estas razones se utiliza ampliamente en sistemas de control, tanto industriales como domésticos.

En los sistemas de control basados en microprocesaador se pueden distinguir los siguientes bloques o módulos:

a) sensores

encargados de convertir las variaciones de la cantidades físicas en señales eléctricas.

b) un Sistema de Adquisición de Datos (SAD)

es el módulo encargado de convertir las señales analógicas de varios sensores a un valor digital que pueda leer la microcomputadora.

c)entrada/salida con el operador

puede consistir de teclado y unidad de despliegues que permitan al usuario introducir los valores de set-point y leer los valores actuales de las variables del proceso.

d) la microcomputadora

es la unidad de procesamiento de los datos entregados por el SAD de acuerdo al algoritmo de control programado y a los datos proporcionados por el operador.

e) interfaces y actuadores

relevadores, convertidores D/A, válvulas solenoides y otros actuadores se usan para controlar las variables del proceso bajo dirección del programa. Existe una circuitería de interface entre la microcomputadora y cada actuador.

Estos módulos quedan conectados entre sí para formar un sistema completo, como lo muestra la figura 2.

Dentro de los objetivos del presente proyecto está el diseño e implementación del módulo de la microcomputadora. La tarjeta SUPERPAT constituye tal módulo. De esta manera, siguiendo el enfoque del diseño modular, los diferentes módulos se pueden desarrollar y conectar al módulo SUPERPAT para formar un sistema de control industrial.

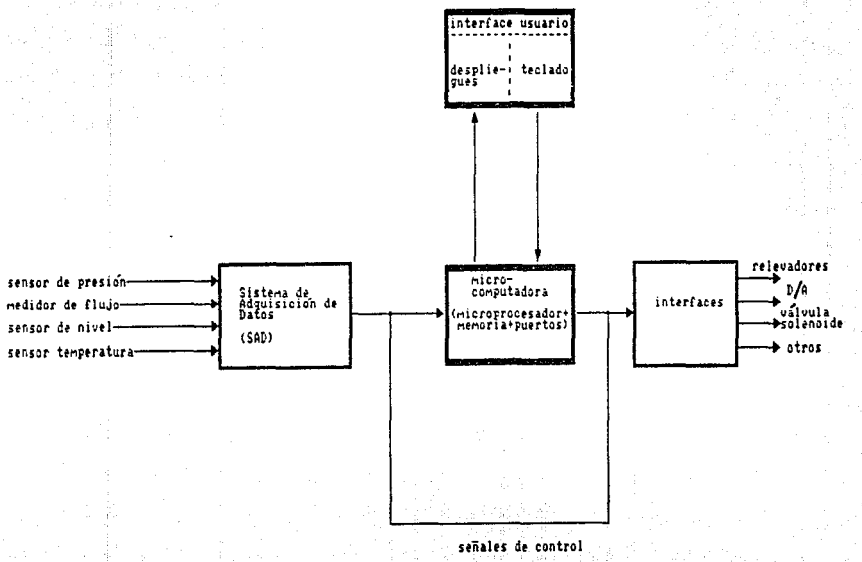


FIG 2 - Diagrama de bloques de un sistema de control de procesos basado en una microcomputadora.

3) DEFINICION DE LAS CARACTERISTICAS DEL SISTEMA SUPERPAT (S S P)

3.1) SISTEMAS MULTITAREA

En el área de la computación existen conceptos con significados diferentes según los diversos autores. Este es el caso de los conceptos relacionados con los sistemas de procesos concurrentes. A continuación se definen algunos de tales conceptos, según Andrews y Schneider [referencia 2], los cuales son los estándares más aceptados. Estas definiciones serán las utilizadas en el presente trabajo.

Un programa secuencial especifica la ejecución secuencial de una lista de declarativas; su ejecución se llama un proceso. Un programa concurrente especifica dos o más programas secuenciales que pueden ejecutarse concurrentemente como procesos paralelos.

Un programa concurrente puede ejecutarse permitiendo a varios procesos compartir uno o más procesadores, o bien corriendo cada proceso en su propio procesador. El primer enfoque se conoce como multiprogramación (multiprogramming), en el cual el sistema operativo multiplexa los procesos en el o los procesadores. El segundo enfoque se conoce como multiprocesamiento (multiprocessing) si los procesadores comparten una memoria común; o como procesamiento distribuido (distributed processing) si los procesadores están conectados por una red de comunicaciones.

Si sólo se dispone de un procesador (como es común), es posible simular el efecto del procesamiento en paralelo haciendo que el procesador vaya ejecutando cada uno de los procesos por turnos de pequeña duración, de manera que el sistema aparenta un comportamiento concurrente al ser analizado desde la perspectiva de una escala mayor de tiempo. Esto significa que un procesador puede ser compartido por varios procesos en un esquema de concurrencia aparente, por medio de alternar los distintos procesos en el procesador.

Algunos autores usan el término tarea como un sinónimo de proceso. Freedman [referencia 3] define una tarea como una unidad de ejecución, generada de un programa o de un módulo de un programa (en el esquema de la programación modular un programa se divide en módulos o rutinas independientes, cada una manejando una función particular; los diversos módulos se pueden desarrollar y probar por separado). La tarea consta de un código ejecutable (creado por la compilación), sus datos y un proceso asociado. De esta manera, se define el concepto multitarea (o multitasking) como la existencia concurrente de varias tareas generadas de distintos módulos de un mismo programa, aunque cabe la posibilidad de que sean generadas de dos o más programas (lo que lo hace sinónimo de multiprogramación).

Se usará el término multitarea con cualquiera de estas dos opciones (módulos concurrentes de un mismo programa o programas concurrentes), es decir, para nombrar cualquier sistema donde existan procesos concurrentes que comparten un mismo procesador.

Los sistemas multitarea permiten utilizar más eficazmente los recursos de la computadora, ya que el hecho de soportar varios procesos simultáneamente aumenta la eficacia del sistema, explotando las posibilidades de la CPU. De manera que sí se justifica la inversión de tiempo y dinero en el desarrollo de técnicas de multitasking.

En los sistemas multitasking es necesario algún esquema de gestión de memoria, el cual es una interfaz entre el esquema lógico de direccionamiento de memoria y la organización física de ésta; se encarga de calcular la dirección física a partir de la dirección lógica que el programador define.

Uno de estos esquemas es la segmentación. Consiste en dividir el espacio total de memoria en áreas llamadas segmentos, de manera que para accesar una dirección de memoria se dan dos componentes: un apuntador de segmento (que contiene la dirección de inicio del segmento), y un desplazamiento (una dirección dentro del segmento). La segmentación fomenta la tendencia actual de la modularidad al facilitar la escritura de módulos totalmente relocalizables, ya que el código máquina se genera usando sólo direcciones relativas dentro del segmento.

3.2) COMPLEJIDAD DEL SISTEMA SEGUN LA APLICACION

Las aplicaciones de los microprocesadores se puede clasificar, de acuerdo a la complejidad de los sistemas digitales en que se han utilizado, en las siguientes:

a) reemplazo de lógica simple

b) control inteligente

c) sistemas multifunciones

d) procesamiento distribuido

En la figura 3 se muestran estas aplicaciones y el crecimiento del sistema según éstas [referencia 4].

La aplicación inicial y más simple de los microprocesadores es como reemplazo de lógica discreta. En este caso se tienen programas pequeños y por tanto en un sólo módulo. Los distintos microprocesadores de 8 bits disponibles actualmente y el lenguaje ensamblador proveen el nivel necesario para implementar funciones lógicas, con un típico programa de 4 KB a 16 KB. En la figura 3a se muestra tal sistema.

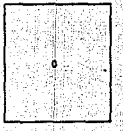
Un ejemplo de esta aplicación es el reemplazo de un mecanismo analógico que controla un sólo parámetro de un dispositivo, tal como el flujo de combustible, velocidad de un ventilador o a una válvula de humedad en una unidad de calefacción o enfriamiento.

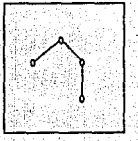
un solo procesador
un solo modulo

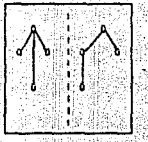
un solo procesador
varios modulos

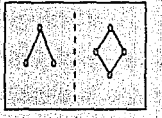
un solo procesador
multiproceso

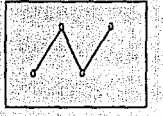
multiples procesadores











APLICACIONES:

a) reemplazo de lógica

b) control inteligente

c) multifunciones

d) control distribuido

Fig 3 - Evolución de la complejidad del sistema

En la figura 3b se muestra la siguiente aplicación, la cual involucra la implementación de funciones de control inteligente. En este caso se tienen programas más grandes, por lo que aparece la necesidad del enfoque de la programación modular, esto es, dividir el programa en módulos. Esto conduce a dos conceptos requeridos en la arquitectura, no presentes en la mayoría de los microprocesadores de 8 bits. Uno es la modularidad del programa, que permite mejor manejo y crecimiento de programas grandes. El otro es la orientación de lenguaje de alto nivel, ya que los módulos en programas grandes se implementan generalmente en un lenguaje de alto nivel.

Un ejemplo de microprocesadores que soportan este nivel de complejidad es la familia del 8086 de Intel, la cual está diseñada para soportar los requerimientos involucrados en los sistemas de software estructurado, incorporando dos conceptos en su arquitectura que ayudan a la implementación de tal software: el soporte de programación modular y la orientación de lenguaje de alto nivel.

El ejemplo de aplicación a este nivel es el control de una unidad de calefacción a través de realimentación de múltiples parámetros. Se pueden coordinar a la vez el flujo de combustible, la velocidad del ventilador, la válvula de humedad, etc.

En la siguiente aplicación, mostrada en la figura 3c, se tienen sistemas aún más grandes, que permiten la ejecución de múltiples procesos corriendo en un sólo procesador simultáneamente. En este caso se requieren los dos conceptos pre-

vios, además de un soporte de sistema operativo multitareas que permita la definición y ejecución correcta de los procèsos.

Continuando con el ejemplo, ahora se tiene control de múltiples unidades de calefacción, cada una manejada por su propio proceso. Las unidades están localizadas una cerca de otra, lo que permite el uso de un sólo procesador para todas.

En la última aplicación (fig. 3d) se tiene una configuración de multiples procesadores, realizando funciones de control distribuido. En este caso los distintos procesadores trabajan en cooperación para el logro de un objetivo común. Requiere de todos los conceptos de arquitectura del nivel previo, más algunos conceptos únicos del mismo. Las diversas formas y variantes de tales configuraciones requieren de una comunicación y coordinación entre procesadores, lo que conduce a cambios radicales en el sistema y la estructura del software que están más allá de los objetivos del diseño de sistemas tradicionales de un sólo procesador multiprogramado.

3.3) ELECCION DE LA ARQUITECTURA Y SOFTWARE ASOCIADO PARA EL SSP

En la figura 3 se observa que las líneas verticales que separan cada nivel de aplicación son significativas: representan umbrales de crecimiento. El cruzar un umbral requiere incorporar los nuevos conceptos involucrados, o puede ocurrir una "crisis de crecimiento": el sistema no podrá evolucionar y tendrá que crearse otro totalmente nuevo, lo que implica un enorme costo, tanto en diseño como en implementación.

Las características de los procesadores de la familia del 8086 de Intel soportan el crecimiento desde los antiguos ambientes de 8 bits hasta los de 16 bits. Además contienen dos conceptos claves: soporte de programación modular y orientación de lenguaje de alto nivel, que soportan el crecimiento a ambientes más complejos. Por estas ventajas se decidió usar un procesador de esta familia en el SSP.

El 8086 es un microprocesador de 16 bits, creado para usarse como CPU en una computadora. El término "16 bits" significa que su unidad aritmético-lógica, sus registros internos y la mayoría de sus instrucciones están diseñadas para trabajar en palabras binarias de 16 bits.

El 8086 tiene un bus de datos de 16 bits y un bus de direcciones de 20 bits, por lo que puede direccionar cualquiera de 1,048,576 localidades de memoria, cada una de un byte de tamaño; las palabras de 16 bits se almacenan en dos localidades consecutivas de memoria.

Otro procesador de esta familia es el 8088, el cual

tiene la misma unidad aritmético-lógica, los mismos registros y el mismo conjunto de instrucciones que el 8086. El 8088 también tiene un bus de direcciones de 20 bits, pero su bus de datos externo es de 8 bits. El 8088 fué escogido para usarse como CPU en la Computadora Personal de IBM (IBM PC) y algunas computadoras compatibles con ella, lo cual le dió a este procesador un liderazgo en el mercado.

El 8088, un microprocesador de 16 bits con bus de datos externo de 8 bits, se creó para prever la continuidad con los antiguos microprocesadores de 8 bits de Intel (el 8085). El 8088 puede reemplazar a uno de ésos en un sistema ya existente. También existe en el mercado mucho hardware disponible para microprocesadores con bus de datos de 8 bits.

Existen otros microprocesadores de Intel que pertenecen a esta familia, como el 80186 y 80188, que incorporan, además del CPU, dispositivos periféricos programables integrados en el mismo chip.

Las microcomputadoras de 16 bits, con su espacio de direccionamiento grande y sus aplicaciones de multitareas, requieren de esquemas de organización de la memoria. Intel ha incorporado en el propio chip 8088 y 8086 un esquema de gestión de memoria muy simple, del tipo de segmentación.

Internamente este procesador tiene 4 registros especiales, llamados registros de segmentación: uno para el código, dos para datos y uno para la pila. Estos registros almacenan la dirección de inicio del segmento dentro de cualquier rango del espacio total de memoria de 1 MegaByte .

La presencia de estos 4 registros permiten la partición

y organización de la memoria de manera modular. Los módulos de código, datos y pila del programa pueden fácilmente cambiarse de área en memoria, ya que son todos relocalizables. Esto permite que bloques de datos y código de varios procesos puedan estar continuamente intercambiándose dentro y fuera de memoria -según la demanda- lo que posibilita la implementación de ambientes multitasking.

En el sistema SUPERPAT, aprovechando las características de utilizar este procesador, se incorpora un software para soportar un ambiente multitasking. Como el SSP es también una herramienta para el desarrollo de sistemas de control, necesita de la incorporación de un lenguaje de programación que permita el desarrollo de programas para las diferentes aplicaciones. El lenguaje que se incorpora es un Basic particular, con instrucciones orientadas hacia aplicaciones de control industrial.

El microprocesador elegido para implementar la microcomputadora SUPERPAT es el 8088. Este procesador permite diseñar sistemas pequeños, y por tanto baratos, con un desempeño comparable al de los otros procesadores de 16 bits. La buena relación costo/rendimiento permite implementar microcomputadoras factibles de emplear en muchas aplicaciones.

Resumiendo, las características definidas para el SSP que sirven como base para su diseño y posterior implementación son:

hardware: microcomputadora basada en el 8088

software: lenguaje Basic que permite la programación en un ambiente multitasking.

3.4) DESCRIPCION GENERAL DEL SSP

Tomando en cuenta las consideraciones previas, se definen aquí las características generales del SSP como un sistema completo y funcional.

El SSP es una microcomputadora en una sola tarjeta basada en el microprocesador 8088, pensada para funcionar como un módulo para implementar diversos sistemas electrónicos de control de procesos industriales.

Debido a su enfoque modular, el diseño del sistema contempla la inserción de la tarjeta SUPERPAT en un portatarjetas a través de un conector de peine que contiene las señales de los buses del sistema (datos, direcciones y control). Se pueden introducir otros módulos, dependiendo del sistema particular que se desee implementar. Todos los módulos insertados en el portatarjetas quedarán comunicados a través de un bus común - el bus denominado OMNIBUS - a través de los conectores de peine.

El SSP es un sistema programable en un ambiente multitasking, es decir, permite la definición de múltiples tareas corriendo a la vez en un sólo procesador. Este nivel se alcanza gracias a la arquitectura y al software incorporados al sistema.

El software del sistema reside en ROM, y consiste en un lenguaje Basic que realiza las funciones de soporte y de coordinación entre procesos necesarias en los sistemas multitasking, y provee además funciones de soporte para aplicaciones de control. Este Basic particular, que se

denominará BASICM (de BASICMultitareas), fué desarrollado por la Cía. Octagon Systems (Westminster, Connecticut, USA), y sólo se adaptó para su uso dentro del SSP.

Debido a sus características, el SSP es útil en aplicaciones donde se requiera supervisión simultánea de múltiples dispositivos (sistemas multifunciones).

La elaboración y prueba de programas se lleva a cabo usando una terminal de video que se comunica con el SSP a través de una interface RS232. La velocidad a la cual se comunican se puede elegir de entre varias opciones, y se programa a través de "jumpers" que se encuentran en la tarjeta SUPERPAT.

El SSP cuenta con memoria RAM donde se almacenan los programas BASICM durante el periodo de desarrollo.

Una vez que los programas Basic han sido probados, se pueden grabar permanentemente en una EPROM. El SSP tiene un socket disponible para dicha EPROM donde el usuario tendrá grabado su propio programa de aplicación. BASICM da a este programa una importante característica, denominada "AUTORUN", y que consiste en que se ejecutará automáticamente, si así se desea, al encender el sistema, sin intervención de algún operador y sin requerir del uso de terminal. El desarrollo de programas, y por tanto la necesidad de usar terminal, es solo un paso en el proceso de implementar un sistema. Para el proceso de grabación de la EPROM se requiere de una tarjeta adicional, tal como se describe en el capítulo correspondiente.

4) SSP, DISEÑO ELECTRONICO Y CONSTRUCCION

4.1) DISEÑO EN DIAGRAMA DE BLOQUES

El SSP es una microcomputadora basada en el microprocesador 8088 de Intel.

Al igual que los más recientes microprocesadores, el 8088 requiere una sola señal de reloj. El 8088 no genera su propia señal de reloj, a diferencia de otros microprocesadores como el 8085. Necesita de un circuito generador de reloj, el 8284, creado por Intel específicamente para él. El 8284 usa un cristal oscilador para determinar la frecuencia del reloj. Intercambiando este cristal se pueden seleccionar diferentes velocidades de operación. La salida del 8284 es una señal de reloj con un ciclo de trabajo de un tercio y una frecuencia igual a la tercera parte de la frecuencia del cristal.

La frecuencia máxima a la que puede operar el 8088 es de 5 Mhz, y existe una versión, el 8088-2, que puede trabajar hasta 8 Mhz.

El bus de datos del 8088 es de 8 bits, por lo que puede conectarse directamente al hardware de 8 bits disponible en el mercado, tanto memorias como dispositivos de entrada/salida.

Su bus de direcciones es de 20 bits, direccionando un total de 1 Megabyte (1 MB) de localidades de memoria, aunque esta capacidad es excesiva para aplicaciones como la presente.

Para la implementación de BASICM se requiere un mínimo de 8 Kilobytes (8 KB) de memoria RAM en la parte inferior del espacio de memoria, para almacenamiento de parámetros y datos del sistema (incluyendo la tabla de vectores de interrupción del 8088, que ocupa el primer KB del espacio de memoria). En el SSP, una capacidad de RAM de 24 KB para programas Basic de usuario se considera suficiente para la mayoría de las aplicaciones de control. Por tanto, el requerimiento total de RAM es de 32K x 8. El circuito comercial usado para implementar esta memoria es el 43256, una RAM estática de 32K x 8.

Los requerimientos de ROM están determinados por el tamaño que ocupa el software del sistema (el BASICM). De acuerdo a los tamaños disponibles comercialmente, la capacidad para la ROM es de 32K x 8. Para grabar este software se utiliza el circuito 27256, una EPROM de dicha capacidad.

El último requerimiento de memoria es el espacio para la EPROM de usuario. De acuerdo a la capacidad instalada en RAM, este espacio se deja también de 32K x 8. El socket que se instala en la tarjeta SUPERPAT contempla la inserción de un chip(*) 27256 (el mismo usado para implementar la ROM del sistema) previamente grabado.

De esta manera, el SSP solo usa un total de 96 KB del espacio de 1 MB de memoria.

* NOTA: El término "chip" es ampliamente usado en el campo de la electrónica como sinónimo de Circuito Integrado, y se usará aquí con esta acepción.

El 8088 tiene otro espacio de direccionamiento para entrada/salida. Para direccionar este espacio solo usa los 16 bits inferiores del bus de direcciones, lo que dá un total de 64 K direcciones diferentes.

Una opción para implementar los puertos es usar chips con puertos programables. Tal es el caso del circuito 8255, creado por Intel, el cual permite el manejo de 24 líneas de entrada/salida divididas en 3 puertos de 8 bits cada uno; el chip es programable y tiene varios modos de operación, por lo que sus posibilidades son diversas [referencia 5].

Para el SSP se requiere además de una interface de comunicación serie para la conexión con una terminal de video, a través de un canal full-dúplex y con capacidad de usar varias velocidades.

La velocidad a la que se comunican el SSP y la terminal se programa a través de unos "jumpers" que se encuentran en la misma tarjeta SUPERPAT. La conexión de estos "jumpers" requiere de un puerto de entrada simple que se implementa con el circuito 74LS244, un buffer tristate de 8 bits.

Para la implementación de la interface serie existen diversas opciones en el mercado. Aprovechando las ventajas de la alta integración, algunos chips ofrecen varias funciones en el mismo circuito integrado. Este es el caso del circuito 8256 de Intel.

El 8256 es una MUART (Multifunction Universal Asynchronous Receiver Transmiter) o UART Multifunciones. Este

dispositivo está diseñado para usarse para comunicación asíncrona serie, pero también provee hardware que soporta entrada/salida paralela, temporizador/conteo y control de interrupciones [referencia 6]. Así, la MUART contiene 4 funciones de periféricos comunmente usadas:

1.- Receptor/transmisor asíncrono serie full dúplex, con generador de velocidad (baud rate) interno.

2.- Dos puertos de entrada/salida paralela de 8 bits cada uno.

3.- Cinco contadores/temporizadores de 8 bits cada uno.

4.- Un controlador de interrupciones con 8 niveles de prioridad.

Las salidas del transmisor y receptor de la MUART manejan niveles TTL y requieren "drivers" antes de conectarse a la terminal, la cual maneja niveles según la norma RS232.

El reloj para generar la velocidad de transmisión y recepción se obtiene del reloj del sistema (el que alimenta al 8088). Esta señal de reloj entra a un divisor programable dentro del 8256, el cual puede dividir entre 1, 2, 3 ó 5, para normalizar la frecuencia interna para el generador de baud rate a 1.024 Mhz, permitiendo, por tanto, frecuencias de 1.024 Mhz, 2.048 Mhz, 3.072 Mhz ó 5.12 Mhz. Si el reloj del sistema no es una de estas cuatro frecuencias, entonces la frecuencia del generador de baud rate no será una estándar; aunque funcionará en tanto cumpla las especificaciones de la hoja de datos.

Para que la velocidad de comunicación sea alguna una de las estándares, se elige usar una de las frecuencias anteriores: la mayor posible para tener un sistema más veloz. Por tanto, la frecuencia del reloj del sistema se elige a 5.12 Mhz, que es un poco superior a la de 5 Mhz especificada para el 8088; sin embargo existen tolerancias en dicho dato del fabricante, por lo que se prevee que el procesador funcionará adecuadamente sin necesidad de utilizar la versión de mayor velocidad. El circuito generador de reloj requerirá, por tanto, un cristal de 15.36 Mhz.

De esta manera, las velocidades posibles de comunicación entre el SSP y la terminal son 110, 300, 600, 1200, 2400, 4800 y 9600 bauds.

Dentro del 8256, los temporizadores integrados son útiles en el sistema para diferentes aplicaciones, tales como retardos de tiempo, interrupciones periódicas, conteo de eventos, generadores de frecuencia, etc. La señal de reloj que los alimenta es la misma que alimenta al generador de baud rate (señal de 1.024 Mhz).

Los dos puertos paralelos incorporados en el 8256 agregan 16 líneas más de entrada/salida al sistema, para tener un total de 40.

Los diversos dispositivos de entrada/salida trabajan con el CPU en un esquema de interrupciones para hacer más eficiente el uso del tiempo del procesador. Si mas de un dispositivo puede generar petición de interrupción, entonces se hace necesario un hardware que controle el

tráfico entre las diversas peticiones y atenciones de interrupción. El 8256 tiene varias fuentes que pueden generar petición de interrupción: el receptor del puerto serie, el transmisor del puerto serie, cualquiera de los 5 temporizadores, el "handshake" del puerto paralelo e incluso alguna interrupción externa. Por ello se hizo necesario incorporar en el chip un controlador de interrupciones encargado de manejar las prioridades entre los distintos circuitos que pueden generarla.

De esta manera, con un solo chip -el 8256- se están implementando casi la totalidad de los requerimientos de entrada/salida, bajo un esquema de interrupciones que hace eficiente al sistema. BASICM hace uso extensivo de esta estructura de interrupciones para implementar muchas de sus características importantes.

El SSP, como está enfocado principalmente a aplicaciones de control, requiere de un dispositivo de "vigilancia" o "watch-dog". Para la implementación de este dispositivo se usan dos circuitos monoestables, contenidos ambos en el chip 74LS123. El watch-dog es visto por el CPU como un puerto de salida. Si se le direcciona queda activado, y se debe seguir direccionando a una frecuencia mayor de 5 Hz (esta frecuencia depende de la implementación particular del circuito). Si por alguna causa se interrumpe ese direccionamiento, el dispositivo genera un pulso en su salida, la cual está conectada al RESET del sistema, que "despierta" de nuevo al microprocesador.

Este mecanismo permite que un programa que se ejecuta y cuyo control se pierda por algún ruido o interferencia en los circuitos, pueda recuperarse y no seguir "perdido", ya que esto podría resultar desastroso en programas que están manejando actuadores o cargas de potencia.

Resumiendo las características de hardware del SSP se tienen las siguientes:

* microprocesador 8088 trabajando a 5.12 Mhz

* un puerto serie RS232 con posibilidad de seleccionar entre 7 diferentes velocidades de comunicación (desde 110 hasta 9600 bauds) a través de "jumpers".

* 40 líneas de entrada/salida programables, divididas en 5 puertos de 8 bits cada uno.

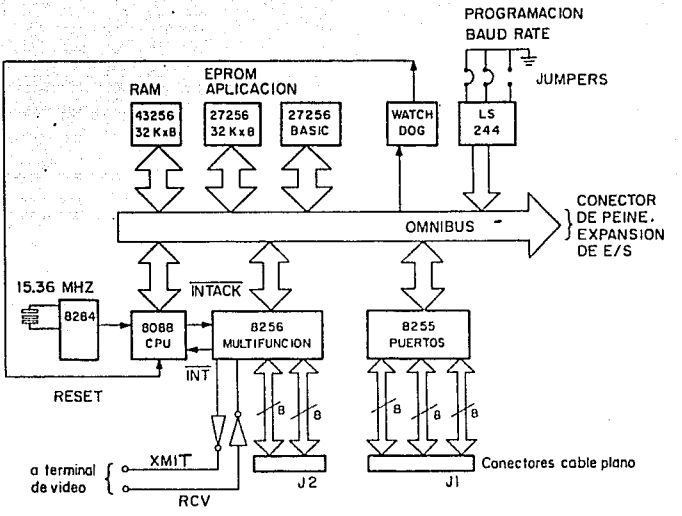
* 96 K bytes de memoria en la tarjeta: 32 K de ROM con el BASIC Multitarea, 32 K de RAM para programas de usuario y para parámetros del sistema, y un socket que permite la inserción de una memoria EPROM 27256 con algún programa de aplicación.

* 5 temporizadores programables, con posibilidad de implementación de un reloj en tiempo real.

* un controlador de interrupciones encargado del manejo de las interrupciones generadas por los temporizadores, el puerto serie y los puertos paralelos de 8256.

* un circuito de vigilancia (watch-dog) que reinicia al sistema cuando se pierde el control del programa.

* posibilidad de expansión a través del bús común OMNIBUS.



TOTAL: 40 Líneas de E/S

Fig 4 - Diagrama de bloques del Sistema SUPERPAT

4.2) CIRCUITO DEL PROCESADOR

4.2.1) SEÑALIZACION EN EL 8088

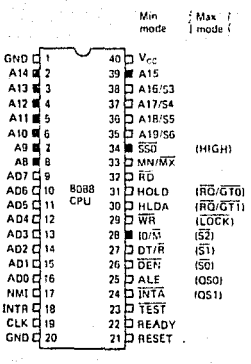
El 8088 viene encapsulado en un chip de 40 pines. Para su energización sólo requiere de una fuente de +5 volts, como es el estándar actual.

Tanto el 8088 como el 8086 pueden funcionar en dos modos distintos: modo MáXimo o modo MÍNimo.

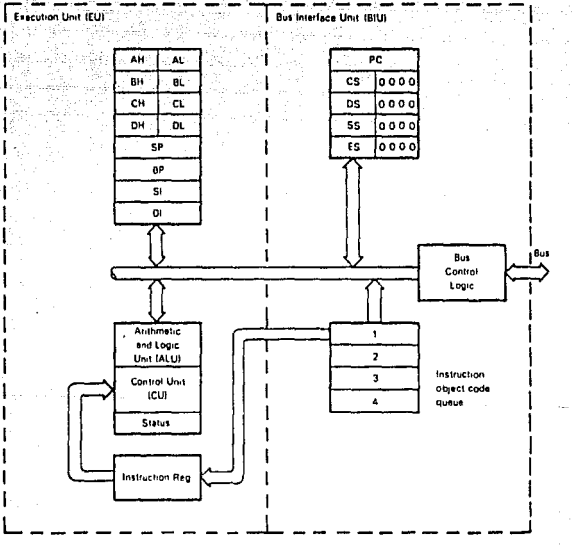
El procesador debe estar en modo máximo si se quiere trabajar en colaboración con el procesador de datos numéricos 8087 y el procesador de entrada/salida 8089. En modo máximo el 8088 depende de otros chips adicionales, como el controlador de bus 8288, para generar el conjunto completo de señales de control de los buses.

El 8088 se opera en modo mínimo en sistemas donde es el único procesador en los buses del sistema.

El modo de operación queda determinado por el nivel lógico aplicado al pin 33, que es la línea de entrada MN/$\overline{MX}$. Si este pin está en nivel alto, el 8088 funcionará en modo mínimo. Algunos pines tienen diferente función según el modo de operación. En la figura 5a se muestra la asignación de las señales a los pines del chip en los dos modos de operación.



a) asignación de pines



b) Arquitectura interna

Fig 5 - Microprocesador 8088 de Intel

Todas las señales del microprocesador están determinadas de acuerdo al reloj que alimenta al chip desde el generador de reloj externo. Un ciclo de reloj se denomina estado.

Para una operación básica del bus, tal como leer un byte de memoria o escribir una palabra a un puerto, se requiere de cierto número de estados. El conjunto de estados requeridos para una operación básica del bus se llama ciclo de máquina.

El 8088 está dividido internamente en dos subprocesadores: la Unidad de Ejecución y la Unidad de Interface con el Bus [referencia 7], como se observa en la figura 5b. El ciclo de máquina es un fenómeno de la unidad de interface con el bus, para la unidad de ejecución no existen estos ciclos, ya que está activa mientras ejecuta instrucciones, e inactiva mientras espera un código de instrucción o un dato que la unidad de interface con bus debe obtener. Por ello, algunos autores hablan de ciclos de bus en lugar de ciclos de máquina para este procesador; la diferencia es que los ciclos de bus sólo ocurren en demanda.

Un ciclo de bus dura 4 períodos de reloj -o estados- (T1, T2, T3 y T4), aunque se puede alargar -introduciendo estado de espera Tw- si así se requiere. Al igual que los ciclos de máquina comunes, cada ciclo de bus asigna periodos de reloj individuales a eventos específicos en el tiempo.

SEÑALES COMUNES EN AMBOS MODOS

NOTA: El estado en que se activa la señal se indica entre paréntesis

AD0-AD7 bidireccional, tristate. Bits inferiores del bus de direcciones (T1); bus de datos (T2 a T4)

A8-A15 salidas, tristate. Bits 8 a 15 del bus de direcciones (T1 a T4)

A16/S3-A19/S6 salidas, tristate. Bits 16 a 19 del bus de direcciones (T1), con valor igual a cero en operaciones de entrada/salida; información de status (T2 a T4)

$\overline{SSO}$ salida, tristate. Información de status

$\overline{RD}$ salida, tristate. Activación de lectura (T2 a T4)

READY entrada. Señal para alargar el ciclo de bus insertando ciclos de espera Tw; necesita sincronizarse por el 8284

$\overline{TEST}$ entrada. Entrada que se examina por medio de una instrucción "wait for test"

INTR entrada. Petición de interrupción mascarable

NMI entrada. Petición de interrupción no mascarable

RESET entrada. Reinicialización del microprocesador. Necesita sincronizarse por el 8284

MN/$\overline{MX}$ entrada. Modo de operación del procesador

Vcc,GND alimentación. Vcc= +5 volts ±10%

CLK reloj. Se obtiene del 8284

SEÑALES EN MODO MAXIMO

$\overline{S0}$,$\overline{S1}$,$\overline{S2}$ salidas, tristate. Status del ciclo de máquina. Se conecta al controlador de bus

$\overline{RQ}/\overline{GT0}$,$\overline{RQ}/\overline{GT1}$ bidireccional. Control de prioridad del bus

$\overline{LOCK}$ salida, tristate. Control de amarre (hold) del bus

SEÑALES EN MODO MINIMO

IO/$\overline{M}$ salida, tristate. Distingue el acceso a memoria o a entrada/salida; válida desde T4 previa al ciclo de bus actual

$\overline{INTA}$ salida. Activación de lectura para los ciclos de reconocimiento de interrupción (T2 a T4)

ALE salida. Señal para captura de la dirección en un "latch". Es un pulso activo en T1

DT/$\overline{R}$ salida, tristate. Se usa como control de la dirección que siguen los datos en un "tranceiver", si éste se requiere usar en el bus de datos

$\overline{DEN}$ salida, tristate. Habilitador para el "tranceiver" en el bus de datos (si se usa)

HOLD entrada. Indica que otro dispositivo maestro requiere usar el bus local. Requiere sincronización externa

HLDA salida. Respuesta a un HOLD. Al activar esta señal, el 8088 pone sus salidas en alta impedancia

Para la presente aplicación solo se requiere el uso de un procesador, así que el 8088 se trabaja en modo mínimo. A continuación se detalla la señalización en dicho modo.

4.2.2) CICLOS DE BUS EN EL MODO MINIMO

Para poder conectar el microprocesador en un sistema completo, es necesario conocer la señalización detallada de los buses. En la figura 6 se tiene un diagrama de tiempos de las señales del 8088 en modo mínimo para un ciclo de bus de lectura y para uno de escritura, tomados del manual del fabricante del chip [referencia 7]. A continuación se explican tales diagramas.

a) CICLO DE BUS DE LECTURA.

Durante el primer estado, T1, se activa la señal IO/$\overline{M}$. Esta señal se pone en nivel alto si se va a leer de un puerto y en nivel bajo si se va a leer desde memoria. En la figura 6 se muestran dos formas de onda para esta señal, porque puede estar activa en nivel alto o bajo para el ciclo de lectura. El punto donde las dos formas de onda se cruzan indica el instante en que la señal se hace válida.

Después de activar IO/$\overline{M}$, el 8088 envía un nivel alto en la señal ALE (Adress Latch Enable). Esta señal se conecta a la entrada de habilitación (STB) de los latches que harán la función de demultiplexar los bits de direcciones y los de datos en líneas diferentes.

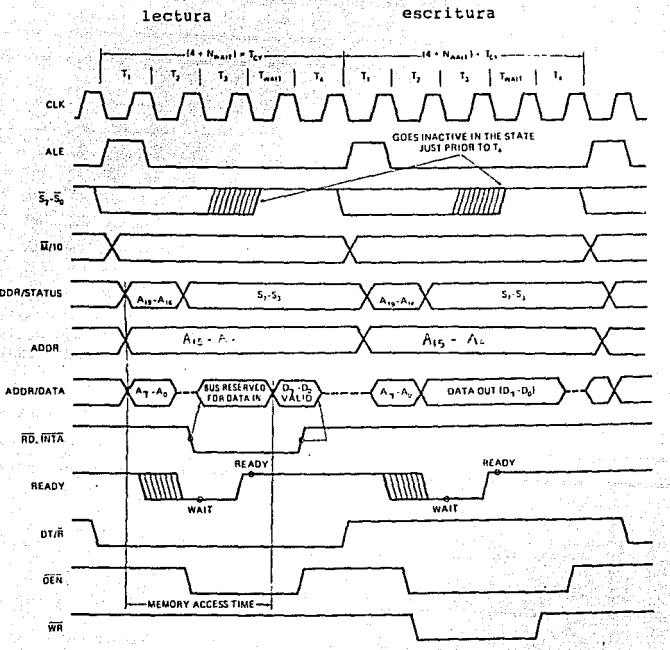


Fig 6 - Diagrama de tiempos de las señales del 8088 en modo mínimo

Después de activar ALE en alto, envía por las líneas ADDR/DATA (señales AD0-AD7) la dirección de la localidad de memoria que desea leer. Como los latches están habilitados por ALE, esta información pasa a través de ellos hasta sus salidas. Entonces el 8088 pone a ALE en nivel bajo en el flanco de subida de T1. Esto deshabilita los latches y provoca que capturen la información de la dirección. Las salidas del latch se pueden ahora utilizar para seleccionar la localidad de memoria o puerto.

Lo mismo se hace con las líneas ADDR/STATUS, que en T1 contienen los bits más significativos de la dirección (A16 a A19).

En las líneas ADDR (bits A8 a A15) se tienen direcciones válidas desde T1 hasta T4 (todo el ciclo de bus).

Como la información de la dirección se tiene después de T1 en los latches, el 8088 no necesita seguir enviándola. Por tanto, como lo muestran las líneas punteadas en la figura 6, el 8088 flota las líneas ADDR/DATA para que puedan usarse como entrada de datos desde memoria o puertos, al mismo tiempo que remueve la información A16-A19 de la líneas ADDR/STATUS y envía por ellas información de status.

El 8088 está ahora listo para leer datos de la localidad de memoria o puerto direccionado, así que casi al final de T2 se activa la señal $\overline{RD}$ en nivel bajo. Con esta señal se habilita al dispositivo direccionado para que ponga un byte sobre el bus de datos. Esta relación causa-efecto se muestra en el diagrama con una flecha que va del flanco de bajada de RD a ls sección "data IN" de la forma de onda ADDR/DATA. (El

círculo de la flecha se pone siempre en la transición o nivel de la señal que causa alguna acción, y la punta de la flecha siempre indica la acción causada. Flechas de este tipo se usan solo para indicar el efecto de una señal de un dispositivo que tendrá sobre otro dispositivo).

En el diagrama se ha indicado un "tiempo de acceso a memoria" (memory access time). La localidad de memoria o puerto direccionado debe poner datos válidos en el bus de datos antes de que finalice ese periodo de tiempo indicado o el 8088 no obtendrá datos válidos. Si la memoria o dispositivo de entrada salida es muy lento respecto a ese tiempo, una solución es alargar el ciclo de bus insertando estados de espera, Tw.

Las dos señales restantes que intervienen en el ciclo de bus son $\overline{DEN}$ y DT/$\overline{R}$.

La señal $\overline{DEN}$ se usa para habilitar buffers bidireccionales en el bus de datos. Para un sistema pequeño, estos buffers no se requieren, pero conforme se agregan dispositivos al sistema, se hacen necesarios.

Esto se debe a que los dispositivos tales como ROMs y RAMs usados alrededor de los microprocesadores tienen entradas MOS, así que en una polarización DC no requieren mucha corriente (aproximadamente 20 uA); pero cada entrada o salida agregada al bus de datos del sistema actúa como un capacitor de pocos picofarads (alrededor de 20 pF) conectado a tierra. Si se agregan demasiados dispositivos a las líneas del bus del datos, las salidas del 8088 no pueden suministrar suficiente corriente para cargar y descargar la capacitancia del

circuito rápidamente; y entonces se necesita agregar un buffer que maneje alta corriente para hacer el trabajo.

Las salidas de ese buffer deben ser tristate para que no interfieran con otras actividades es esas líneas (por ejemplo, cuando sobre esas líneas hay direcciones). La señal $\overline{DEN}$ del 8088 se usa para habilitar los buffers sobre el bus de datos.

Los buffers usados en el bus de datos deben ser bidireccionales, porque el procesador envía datos en operaciones de escritura y los recibe en operaciones de lectura. La señal DT/$\overline{R}$ del 8088 se usa para especificar la dirección en la cual los buffers están habilitados. Cuando DT/$\overline{R}$ esté en nivel alto, el buffer transmitirá datos -si está habilitado por $\overline{DEN}$- del 8088 hacia ROM, RAM o puertos. Si DT/$\overline{R}$ está en nivel bajo, entonces los datos vienen hacia el 8088.

Durante T1 del ciclo de bus del 8088 se activa DT/$\overline{R}$ en nivel bajo para poner al buffer de datos en el modo receptor. Luego, después de que el bus envió los bits de direcciones, se activa $\overline{DEN}$ en nivel bajo para habilitar los buffers sobre el bus de datos. Así, los datos puestos por el puerto o memoria direccionada serán capaces de llegar al 8088.

Al finalizar el diseño del hardware se hará un análisis estático de corrientes de carga para justificar el uso o no de un buffer bidireccional en el bus de datos.

b) CICLO DE BUS DE ESCRITURA

Las señales para un ciclo de bus de escritura son muy semejantes a las del ciclo de lectura.

Durante T1 se activa IO/$\overline{M}$ en nivel bajo si la escritura va a ser en memoria, y en nivel alto si va a ser en un puerto. Casi al mismo tiempo ALE toma un nivel alto para activar los latches de direcciones. El 8088 entonces saca la dirección por ADDR/DATA (A0-A7), ADDR/STATUS (A16-A19) y ADDR (A8-A15). Cuando se lee o escribe a un puerto, las líneas A16-A19 tienen siempre nivel bajo, porque el 8088 solo envía dirección de 16 bits para los puertos.

Después de que esta dirección han tenido tiempo de pasar a través de los latches, el 8088 desactiva ALE poniéndolo en nivel bajo otra vez, en el flanco de subida de T1, para capturar la información de la dirección en las salidas de los latches. Además de mantener la dirección, estos latches también funcionan como buffers para las líneas de direcciones.

Después de que la ha capturado en el latch, el 8088 quita la información de la dirección del bus multiplexado direcciones/datos (ADDR/DATA) y pone los datos deseados en dicho bus. Entonces se activa la señal $\overline{WR}$ en nivel bajo. Esta señal se usa para activar la memoria o puerto donde los datos se escribirán.

Después de que la memoria o puerto direccionado ha tenido tiempo para aceptar los datos del bus de datos, el 8088 pone en nivel alto la señal $\overline{WR}$ y flota el bus de datos.

Si la entrada READY se pone en nivel bajo por algún hardware externo antes o durante T2 del ciclo de bus, el 8088 insertará estados de espera después de T3.

Si el sistema es demasiado grande que necesita buffers en el bus de datos, entonces DT/$\overline{R}$ se conectará a los buffers.

Esta señal estará en nivel alto para poner a los buffers en modo transmisor. Cuando $\overline{DEN}$ se active, habilitará a los buffers y los datos pasarán a través de los buffers al puerto o memoria direccionada.

4.2.3) OBTENCION DE LOS BUSES DEL SISTEMA

De acuerdo a la temporización de las señales producidas por el 8088, se requieren latches para capturar la información de la dirección, enviada por el microprocesador sólo en T1, para tenerla presente en los siguientes estados (T2 a T4) del ciclo de bus.

Intel creó para ello el circuito 8282, un latch de 8 bits. Un circuito de función similar exite en la línea de productos TTL: el 74LS373, 8 latches con salidas tristate.

La información de los datos no necesita capturarse en un latch, se toma del bus multiplexado direcciones/datos, ya que las señales que se usan para indicar el momento de la transferencia de datos ($\overline{RD}$ y $\overline{WR}$), son activas en el momento en que el bus está listo para transportar datos (después de T1).

De esta manera, la conexión del procesador para obtener sus buses se muestra en la figura 7.

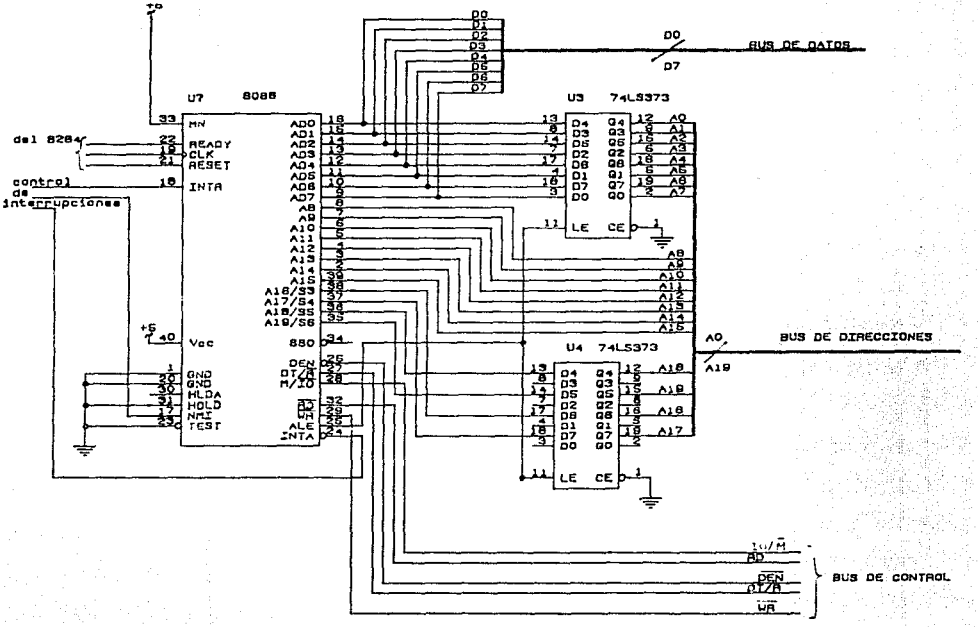


Fig 7 - Circuito para el 8088 en modo mínimo

4.3) CIRCUITO GENERADAOR DE RELOJ

El circuito 8284 fué diseñado especialmente para generar la señal de reloj del 8088, así que no hay que preocuparse por los parámetros de esta señal: está garantizado el correcto periodo de reloj, su ciclo de trabajo, etc. El 8284A se puede usar en sistemas de hasta 8 Mhz, el 8284A-1 debe usarse si se requiere un reloj de 10 Mhz. Para el SSP es suficiente el primero de éstos, ya que su fecuencia de reloj es de 5.12 Mhz.

El 8284 contiene un oscilador controlado por cristal, un contador divisor entre 3 y sincronización completa para READY y para la lógica de RESET [referencia 8].

Las señales que salen del 8284 hacia el procesador son CLK (señal de reloj), RESET (señal de reinicialización del procesador) y READY (señal para sincronizar al procesador con dispositivos externos más lentos). Tanto RESET como READY son señales sincronizadas con CLK.

Otras señales de salida del 8284 son PCLK (señal que funciona como reloj para periféricos, tiene un ciclo de trabajo de un medio y una frecuencia a la mitad de CLK) y OSC (una señal de reloj con niveles TTL con frecuencia igual a la del cristal). Estas dos salidas se han dejado sin uso en el SSP.

Las diferentes señales de entrada al 8284 controlan su modo de operación [referencia 8].

La lógica de reset provee una entrada llamada $\overline{RES}$; la cual se sincroniza con el flanco de bajada de CLK. Con una red RC simple conectada a esta entrada, se puede proveer de reset automático al encender el sistema. Además, si se conecta la salida del watch-dog a esta línea, se tiene control en el sistema para generar el reset cuando el programa se pierda.

Otros pines de entrada son X1 y X2, los cuales son los puntos de conexión para el cristal oscilador con frecuencia igual al triple de la frecuencia de reloj requerida para el CPU. En este caso el cristal oscila a 15.12 Mhz.

El generador de reloj consiste de un contador divisor entre tres, con una entrada especial de "clear" que inhibe el conteo, llamada CSYNC, la cual se utiliza para sincronizar el reloj de salida con un evento externo. En la actual aplicación se deja inactiva (en nivel bajo).

La entrada $F/\overline{C}$ es un pin que selecciona ya sea el cristal oscilador o la entrada EFI como el reloj de alimentación al 8284. Si se selecciona EFI, la salida se toma de OSC. Como se va a utilizar cristal, $F/\overline{C}$ debe quedar en nivel bajo. EFI no se utiliza y se puede conectar a nivel bajo.

El 8284 contiene dos entradas para READY, que son RDY1 y RDY2, provistas para acomodar dos sistemas que pueden accesar los buses. Cuando RDY está en nivel alto, es la indicación de que el dispositivo localizado en el bus de datos del sistema ha recibido o está disponible para aceptar el dato.

Cada entrada RDY tiene un habilitador, $\overline{AEN1}$ y $\overline{AEN2}$, para sus respectivas señales RDY. Si se tiene una configuración donde solo un procesador controla los buses del sistema (es el caso del SSP), ambas entradas $\overline{AEN}$ se conectan a nivel bajo, RDY2 se conecta a nivel bajo, y RDY1 es la entrada que se usa para generar la señal READY.

De acuerdo a este funcionamiento del 8284 se obtiene el circuito generador de reloj para el SSP, el cual se muestra en la figura 8.

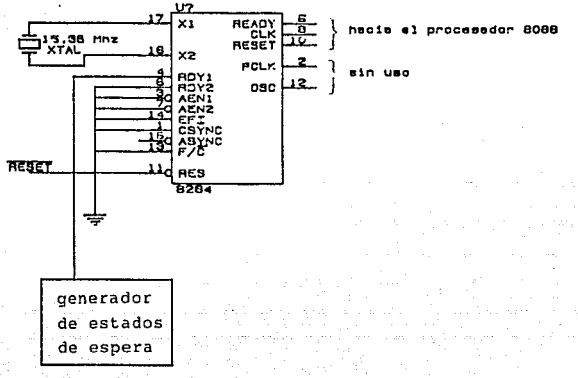


Fig 8 - Circuito generador de reloj para el 8088 en el Sistema SUPERPAT

4.4) CIRCUITO DE MEMORIA

Para la conexión de cada memoria dentro del espacio de 1 Mega Byte se siguen algunas consideraciones:

a) El primer KB del espacio total de memoria lo utiliza la estructura de interrupciones del 8088 para almacenar los 256 vectores de interrupción posibles. Para esta área es conveniente usar RAM para tener flexibilidad en el uso las interrupcines; además, como BASICM requiere que en los siguientes 7 KB haya RAM, entonces los primeros 8 KB ,al menos, deben ser RAM. Por ello, el chip de memoria RAM se elige poner a partir de la dirección cero.

b) Al encender el sistema, éste se reseteará automáticamante si se le conecta una red RC para ello. Cuando esto ocurre, entre otras cosas, el registro de segmento CS se pone en FFFFH (*), los registros de segmento DS, SS y ES, y el contador de programa IP se ponen en ceros, y comienza la ejecución del programa. Como CS contiene FFFFH e IP contiene 0, la primera instrucción ejecutada se toma de la localidad de memoria FFFF0H. En esta dirección debe haber ROM con el programa que inicialice el hardware periférico y que tome el control del sistema.

Por esta razón, la ROM con el software del sistema (el BASICM) debe ir en los 32 KB finales del espacio de memoria. Esto conduce a que la ROM inicia en la dirección F8000H.

* NOTA: Se usará la notación de poner H al final de un número para indicar que está expresado en base hexadecimal.

c) Para la EPROM con programas de usuario no existe ninguna restricción y se puede colocar en cualquier espacio disponible de los restante. BASICM la considera a partir de la dirección 10000H.

De acuerdo a las consideraciones anteriores, se puede obtener el mapa de memoria para el SSP como sigue:

DIRECCION	MEMORIA
00000H a 07FFFH (32k)	RAM 43256
10000H a 17FFFH (32K)	EPROM 27256
F8000H a FFFFFH (32K)	ROM BASICM

El circuito decodificador de memoria para implementar este mapa se muestra en la figura 9.

De acuerdo a ese circuito, el mapeo se repite en algunas direcciones, debido a que no se utilizó A17 y A18 en la decodificación, para hacerla con menos chips. El usuario del sistema debe ver el mapa de memoria mostrado en la siguiente tabla:

DIRECCION	MEMORIA	CAPACIDAD
00000H-07FFFH	RAM	(32K)
08000H-0FFFFH	nada	(32K)
10000H-17FFFH	EPROM	(32K)
18000H-1FFFFH	nada	(32k)
20000H-F7FFFH	sin uso	(864K)
F8000H-FFFFFH	ROM	(32K)

TABLA 1 - Mapa de memoria del SSP

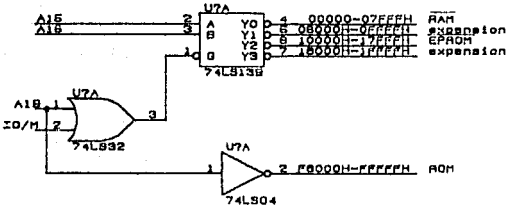


Fig 9 - Circuito de decodificación de memoria para el Sistema SUPERPAT

4.5) CIRCUITO PARA PUERTOS DE ENTRADA/SALIDA

4.5.1) CIRCUITO DE DECODIFICACION

Los diferentes dispositivos para implementar la entrada/salida en el SSP se mapean en el espacio total de 64 KB de la siguiente manera:

DIRECCION	CIRCUITO
0000H	8255 (4 puertos)
4000H	Watch-dog (1 puerto)
8000H	disponible para expansión
C000H	8256 (16 puertos)
E000H	datos del baud rate (1 puerto)

El circuito para implementar esta decodificación del espacio de entrada/salida se muestra en la figura 10.

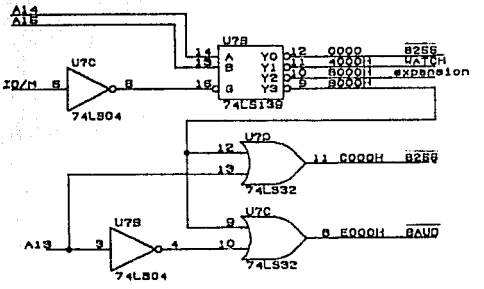


Fig 10 - Circuito de decodificación de entrada/salida para el Sistema SUPERPAT

4.5.2) CONEXION DEL 8255

De acuerdo a la información del 8255 proporcionada por el fabricante [referencia 5], los pines de control del chip para interconectarlo al microprocesador son:

$\overline{CS}$ Señal activa baja, habilita la comunicación entre el 8255 y el CPU

$\overline{RD}$ habilita al 8255 enviar información de datos o de status hacia el CPU por el bus de datos.

$\overline{WR}$ habilita al 8255 para que el CPU escriba en él datos o palabras de control.

A0 y A1 selección de puerto 0 ó 1. Estas señales, junto con RD y WR, controlan la selección de uno de los tres puertos o de los registros de control. Se conectan a los bits menos significativos del bus de direcciones.

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	OPERACION
0	0	0	1	0	leer dato del puerto A
0	1	0	1	0	leer dato del puerto B
1	0	0	1	0	leer dato del puerto C
0	0	1	0	0	escribir dato al puerto A
0	1	1	0	0	escribir dato al puerto B
1	0	1	0	0	escribir dato al puerto C
1	1	0	1	0	escribir comandos al 8255
x	x	x	x	1	pines de salida de datos en tristate

TABLA 2 - Operación básica del 8255

4.5.3) CONEXION DEL 8256

El diseño versátil del 8256 permite conectarlo directamente al 8085, 8086, 8088, 80186, 80188 y a la familia de microcomputadoras en un chip MCS-48 y MCS-51 de Intel [referencia 6].

Para proveer todas las funciones de la MUART en 40 pines y mantener direccionamiento directo de registros, se usa un bus multiplexado de direcciones/datos. La MUART contiene 16 registros de lectura/escritura internos directamente direccionables. Cuatro de las ocho líneas direcciones/datos se usan para generar la dirección. Para sistemas de 16 bits (como es el caso del SSP), AD1-AD4 se usan para generar la dirección para los registros internos de datos, y A0 se usa como una segunda activación del chip en bajo.

El 8256 contiene una sección de hardware que le permite comunicarse con el microprocesador. La interface de bus del 8256 usa las señales de control de bus estándares, las cuales son compatibles con todos los periféricos y microprocesadores de Intel.

Las señal de activación de chip ($\overline{CS}$) proviene del decodificador de direcciones, y es capturado en un latch junto con las direcciones en el flanco de bajada de ALE. Por tanto, CS no tiene que permanecer baja durante todo el ciclo de bus.

Los buffers que conectan al bus de datos permanecen en tristate a menos que una señal $\overline{RD}$ o $\overline{WR}$ se active mientras $\overline{CS}$ haya sido activada.

La señal INT se usa para interrumpir al CPU e $\overline{INTA}$ para

recibir el reconocimiento del CPU de que la petición de interrución será atendida. El MUART puede vectorizar al CPU a la rutina de servicio apropiada dependiendo de la fuente de interrupción.

La tabla 3 muestra los registros internos del 8256 y su dirección de acceso en el SSP de acuerdo a la conexión del chip dentro del sistema (NOTA: "int." se usa como abreviatura de "interrupción" o "interrupciones").

DIRECCION	REG.DE LECTURA	REG.DE ESCRITURA
C000H	comando 1	comando 1
C002H	comando 2	comando 2
C004H	comando 3	comando 3
C006H	modo	modo
C008H	control puerto 1	control puerto 1
C00AH	int. habilitadas	poner interrupciones
C00CH	dirección de int.	quitar interrupciones
C00EH	buffer receptor	buffer transmisor
C010H	puerto 1	puerto 1
C012H	puerto 2	puerto 2
C014H	timer 1	timer 1
C016H	timer 2	timer 2
C018H	timer 3	timer 3
C01AH	timer 4	timer 4
C01CH	timer 5	timer 5
C01EH	status	modificación

TABLA 3 - Registros internos del 8256 en el SSP

4.5.4) CIRCUITO SELECTOR DE VELOCIDAD

Para seleccionar la velocidad a la cual se comunica el SSP y la terminal de video se usan unos "jumpers". El software del sistema lee la información de esos "jumpers" y hace la programación necesaria en el 8256 para que la UART funcione a la velocidad elegida.

El circuito usado como puerto de entrada es el 74LS244, un buffer tristate de 8 bits con una línea de activación. El chip se habilita, y por tanto pasa la información programada en los "jumpers" hacia el bus de datos, cuando se hace una operación de lectura de puerto en la dirección E000H. Este puerto es sólo de lectura.

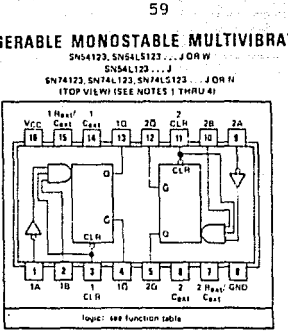
4.5.5.) CIRCUITO WATCH-DOG

Se basa en el chip 74LS123, el cual contiene dos multivibradores monoestables redisparables, que pueden generar un pulso de un ancho controlado.

El circuito se dispara con un flanco de subida en su entrada B, o uno de bajada en su entrada A, como lo muestra la tabla funcional del dispositivo [referencia 9], mostrado en la figura 11a.

RETRIGGERABLE MONOSTABLE MULTIVIBRATORS

SN54123, SN54LS123 . . . J OR W
SN54L123 . . . J
SN74123, SN74L123, SN74LS123 . . . J OR N
(TOP VIEW) (SEE NOTES 1 THRU 4)

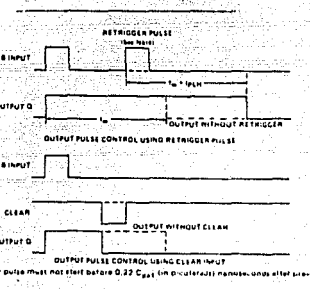


'123, 'L123, 'LS123
FUNCTION TABLE

INPUTS			OUTPUTS	
CLEAR	A	B	Q	Q̄
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⎍	⎎
H	↓	H	⎍	⎎
↑	L	H	⎍	⎎

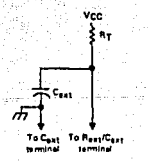
a) asignación de pines y tabla funcional

description (continued)



NOTE: Retrigger pulse must not start before 0.32 C_{ext} (in picofarads) nanoseconds after previous trigger pulse.

b) control del pulso de salida



TIMING COMPONENT CONNECTIONS

c) conección de componentes externos

Fig 11 - Funcionamiento del circuito 74LS123

Una vez disparado, el ancho del pulso básico puede extenderse redisparando la compuerta en la entrada A o en la B (con flancos descendentes o ascendentes respectivamente), o bien reducirse con el uso de una entrada "clear"(CLR), como se observa en la figura 11b.

El tiempo de duración del pulso está programado por la selección de valores de capacitancias y resistencias externas, las cuales se conectan entre los pines Cext y Rext/Cext del chip, como se indica en la figura 11c.

De acuerdo a la hoja de datos, el ancho del pulso de salida para el LS123, si Cext > 1000 pF, se obtiene así:

$$tw=(0.45)\ (Rt)\ (Cext)$$

donde Rt está en kilo-ohms, Cext en pF y tw en ns.

Aplicando esta fórmula, si para el primer monoestable se eligen valores de Cext=47 microfarads y Rt=47 kilo-ohms, se obtiene un pulso de duración tw= 994 milisegundos, aproximadamente 1 segundo.

Para implementar el watch dog, la señal $\overline{WATCH}$ (generada por la decodificación de entrada/salida) se conecta a la entrada B de un primer monoestable. Cuando $\overline{WATCH}$ se activa, se genera una transición de nivel bajo a nivel alto en dicha entrada B y se dispara por tanto el monoestable. Su salida $\overline{Q}$ pasa por tanto de "1" a "0" en un pulso negativo de aproximadamente 1 segundo de duración.

Esta salida se conecta a la entrada B del segundo monoestable, el cual no se dispara porque se requiere una transición de "0" a "1" en este pin para dispararlo.

Como el primer monoestable es redisparable, si la fre-

cuencia con que se habilite $\overline{WATCH}$ es mayor de 1 Hz (esto es, antes de que la duración de tw=1 seg termine), su salida $\overline{Q}$ permanecerá en estado bajo, y por tanto no se disparará el segundo monoestable.

Sin embargo, si en algún momento deja de activarse $\overline{WATCH}$, entonces $\overline{Q}$ del primer monoestable pasará de "0" a "1" (termina el pulso), disparando al segundo monoestable, en cuya salida $\overline{Q}$ aparece entonces un pulso negativo. Esta salida se conecta a la entrada $\overline{RES}$ del 8284 para generar un RESET al 8088.

Para el segundo monoestable, los valores elegidos para Cext es 10 microfarads y Rt de 33 kilo-ohms, lo que dá un ancho de pulso de 13.5 milisegundo, suficiente para activar $\overline{RES}$ en el 8284.

En la figura 12 se muestra la implementación del circuito watch dog para el SSP.

4.5.6) MAPA COMPLETO DE ENTRADA/SALIDA

De acuerdo a las conexiones de los distintos circuitos que implementan la entrada/salida en el SSP, se obtiene el mapa completo mostrado en la tabla 4.

La notación usada en dicha tabla es: R=sólo lectura; W=sólo escritura; R/W=lectura o escritura; int.=interrupción.

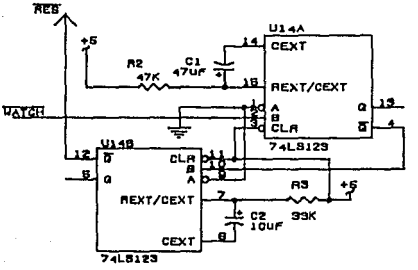


Fig 12 - Circuito del Watch-dog

DIRECCION	PUERTO	TIPO	CIRCUITO
0000H	dato puerto A	R/W	
0001H	dato puerto B	R/W	8255
0002H	dato puerto C	R/W	
0003H	comando	W	
4000H	watch dog	W	74LS123
8000H	disponible expansión		
C000H	comando 1	R/W	
C002H	comando 2	R/W	
C004H	comando 3	R/W	
C006H	modo	R/W	
C008H	control puerto 1	R/W	
C00AH	int. habilitadas	R	
	poner int.	W	
C00CH	dirección de int.	R	
	quitar int.	W	
C00EH	buffer receptor	R	8256
	buffer transmisor	W	
C010H	puerto 1	R/W	
C012H	puerto 2	R/W	
C014H	timer 1	R/W	
C016H	timer 2	R/W	
C018H	timer 3	R/W	
C01AH	timer 4	R/W	
C01CH	timer 5	R/W	
C01EH	status	R	
	modificación	W	

TABLA 4 - Mapa completo de entrada/salida del SSP

4.6) ANALISIS ESTATICO DEL SSP

El objetivo de este análisis es determinar si las conexiones entre los circuitos funcionará apropiadamente desde el puento de vista estático (análisis en DC). Para ello se requiere que:

a) los niveles de voltaje de salidas y entradas sean compatibles, lo cual ya se ha estandarizado a "niveles TTL" y se cumple en todos los chips usados en el SSP.

b) que las corrientes manejada por cada salida no excedan los valores máximos especificados por el fabricante. Este parámetro es el que requiere analizarse para determinar si es apropiado o no.

En el caso de que la carga en las salidas de un chip exceda su valor nominal, se requiere el uso de buffers. Los buffers son circuitos que incrementan la capacidad de corriente en las líneas de salida de un chip. Si se requieren buffers en el bus de datos, éstos deben ser bidireccionales, debido a que existen operaciones de lectura y escritura desde el CPU; el tipo de operación determina la dirección del buffer.

Los datos necesarios para realizar este análisis de corrientes son:

1) la carga que representan los circuitos TTL y MOS conectados a la salida de cada chip; y

2) las máximas corrientes de carga entregadas por cada chip en su salida.

Para el primer punto se sabe que los circuitos MOS

tienen una impedancia de entrada muy grande, por lo que la corriente que demandan en su entrada es muy pequeña (generalmente 10 uA).

Existen chips (tanto TTL como MOS) con salidas tristate. Estos circuitos, aunque se encuentran virtualmente desconectados cuando están deshabilitados, sí representan una pequeña carga, ya que existe una pequeña corriente de fuga. Su valor es variable dependiendo del circuito, aunque la mayoría de los fabricantes lo han estandarizado a 20 uA.

Para el segundo punto no existen valores estandarizados y cada fabricante proporciona sus propios valores.

La tabla 5 muestra los datos de los chips utilizados en el SSP necesarios para realizar el análisis estático del sistema.

Ahora el problema se reduce a calcular, por la ley de Kirchoff, la carga máxima en la salida del chip que se esté examinando y compararla con el valor máximo especificado, tanto para el nivel alto como para el nivel bajo, y hacerla para todos los chips.

Esta tarea puede simplificarse probando sólo las salidas más susceptibles de presentar problemas, esto es, en salidas de los chips MOS que conectan a varias cargas TTL.

Las líneas con mayor carga en el SSP son las del bus de datos. En la figura 13 se realiza un análisis estático de esta línea. En algunos circuitos que tienen líneas bidireccionales se debe considerar la carga que representan su entrada y su salida.

	LS373	LS244	27256	43256	8256	8255	8088	
I_{IH}	20	20	10	10	10	10	10	uA
I_{IL}	-400	-200	-10	-10	-10	-10	-10	uA
I_{OH}	-2.6	-15					-2.5	mA
I_{OL}	24	24					2	mA
I_{OZL}	-20	-20	-20	-20	-20	-20		uA
I_{OZH}	20	20	20	20	20	20		uA

NOTAS:

Se toma como positiva la corriente que entra al chip

I_{IH} Corriente de carga para una salida en estado alto

I_{IL} Corriente de carga para una salida en estado bajo

I_{OH} Corriente de entrada en estado alto

I_{OL} Corriente de entrada en estado bajo

I_{OZL} Corriente de salida en estado de alta impedancia al aplicarle un voltaje bajo

I_{OZH} Corriente de salida en estado de alta impedancia al aplicarle un voltaje alto

TABLA 5 - Datos de los chip utilizados en el SSP necesarios para el análisis estático.

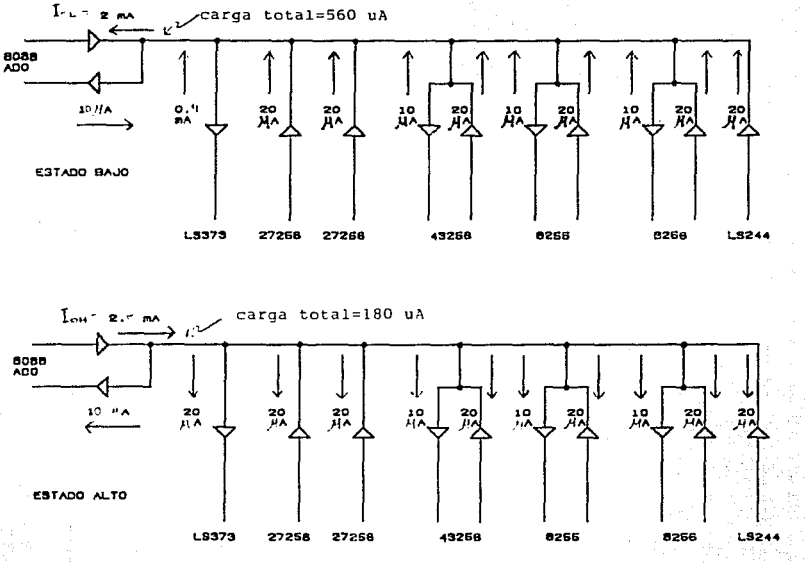


Fig 13 - Análisis estático del bus de datos del Sistema SUPERPAT

En la figura 13 se obtienen las máximas corrientes de carga que demandan todos los circuitos conectados a las líneas direcciones/datos del 8088.

Para el estado bajo, esta corriente es de 560 uA, muy inferior a 2 mA que es la máxima corriente que puede suministrar el 8088. Para el estado alto, la carga total es de 180 uA, también inferior a 2.5 mA que puede suministrar el 8088 en este estado.

Por tanto no hay necesidad del uso de buffers en el bus de datos.

Para las otras líneas del procesador las cargas son menores, por lo que funcionarán apropiadamente.

La conclusión final de este análisis es que el diseño electrónico del SSP funciona correctamente desde el punto de vista estático.

4.7) ANALISIS DINAMICO DEL SSP

Los diagramas de tiempos anteriormente usados (figura 6) bastan para mostrar la secuencia de actividades en los buses del 8088. Sin embargo no están totalmente detallados para poder determinar si un dispositivo (memoria o puerto) es lo suficientemente rápido para trabajar en un sistema 8088 a una velocidad dada.

Para poder hacer tales cálculos de precisión en el tiempo, el fabricante proporciona hojas de datos con diagramas de tiempos detallados. En la figura 14a se muestra uno de tales diagramas para el 8088 en modo mínimo, tomado del manual de Intel [referencia 7].

No es necesario preocuparse de muchos de los parámetros indicados en esa hoja de datos. En muchos sistemas no hay que preocuparse, por ejemplo, acerca de los parámetros de la señal de reloj, porque se usará el 8284 con el cristal para producir tal señal.

El principal parámetro que se necesita considerar para determinar si una memoria o dispositivo de entrada/salida trabajará adecuadamente a la velocidad del sistema, es el tiempo de acceso.

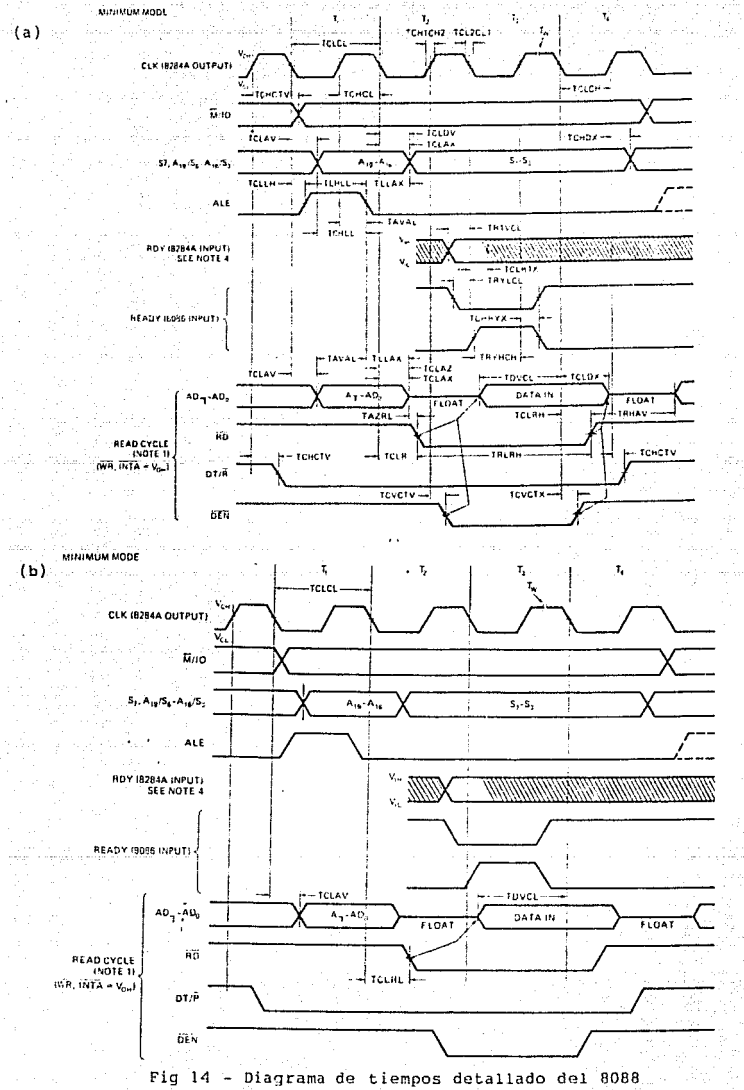


Fig 14 - Diagrama de tiempos detallado del 8088 en modo mínimo

El tiempo de acceso (Tacc) para una memoria se define como el tiempo máximo en el que el dispositivo pondrá datos válidos en sus salidas, si el chip y sus buffers de salida están habilitados, después de que una dirección se aplica a sus líneas de dirección.

Para determinar si este parámetro es correcto, se requiere examinar los parámetros de tiempo y cómo se habilita la memoria en el sistema. Para ello, se muestra en la figura 14b un diagrama simplificado del diagrama de tiempos suministrado por el fabricante.

En este diagrama simplificado se observa que las direcciones son válidas en T1, un tiempo etiquetado TCLAV después del flanco de bajada de T1. TCLAV (Time from Clock Low to Adress Valid) es el tiempo de retardo para tener una dirección válida en el bus. De acuerdo a la hoja de datos [referencia 7], el valor máximo para este tiempo es de 110 ns.

Después, en el mismo bus se nota que los datos válidos deben llegar al 8088 desde memoria un tiempo TDVCL antes del flanco de bajada del reloj en T3. TDVCL (Time Data must be Valid before Clock goes Low) tiene un valor de 30 ns según la misma hoja de datos.

El tiempo entre el final del intervalo TCLAV y el inicio del intervalo TDVCL es el tiempo disponible para poner las direcciones en la memoria y accesarla. Este tiempo se puede determinar restando TCLAV y TDVCL del tiempo de 3 periodos de reloj.

Con un reloj de 5.12 Mhz, cada ciclo de reloj es de 195 ns. Tres ciclos de reloj dan un total de 585 ns. Restando 110

ns de TCLAV y 30 ns de TDVCL, resulta en 445 ns disponibles para poner la dirección en la memoria y para su Tacc.

Las señales AO-A7 y A16-A19 van del 8088 a través de los latches 74LS373 para llegar a las memorias. El tiempo de propagación del LS373 también debe restarse de la cantidad obtenida para determinar cuánto tiempo queda disponible para el Tacc. El máximo retardo para el 74LS373 es de 18 ns, que restado de 445 ns dá finalmente un tiempo de acceso Tacc=427 ns.

La figura 15 muestra el resumen del cálculo del tiempo de acceso a memoria en el SSP.

Según los manuales del fabricante de las memorias, el máximo tiempo de acceso para la EPROM 27256 es de 250 ns, y para la RAM 43256 es de 120 ns, ambos muy inferiores de 427 ns.

Por tanto, el tiempo de acceso de las memorias utilizadas es apropiado para el 8088 operando a 5.12 Mhz, y no se requiere el uso de circuito generador de estados de espera.

La conclusión final de este análisis es que el diseño electrónico del SSP funciona correctamente también desde el punto de vista dinámico.

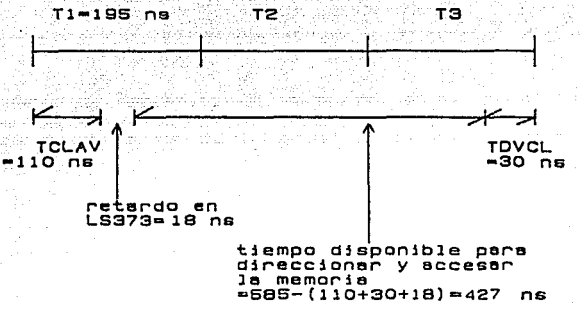


Fig 15 - Resumen del cálculo del tiempo de acceso a memoria en el Sistema SUPERPAT

4.8) IMPLEMENTACION FINAL

De acuerdo al diseño electrónico del sistema SUPERPAT realizado, después de hacer el análisis estático y el dinámico, se concluyó que el sistema opera apropiadamente, ya que todos los parámetros están dentro de los rangos que especifica el fabricante de los diferentes chips.

De esta manera se puede obtener el diagrama electrónico final del SSP, el cual se muestra en la figura 16. En la tabla 6 se muestra la tabla de todos los componentes necesarios.

En la figura 17 se muestra el "layout" de la tarjeta SUPERPAT.

La implementación física se realizó con apoyo del II-UNAM. La tarjeta se encuentra en la Coordinación de Automatización de dicha dependencia.

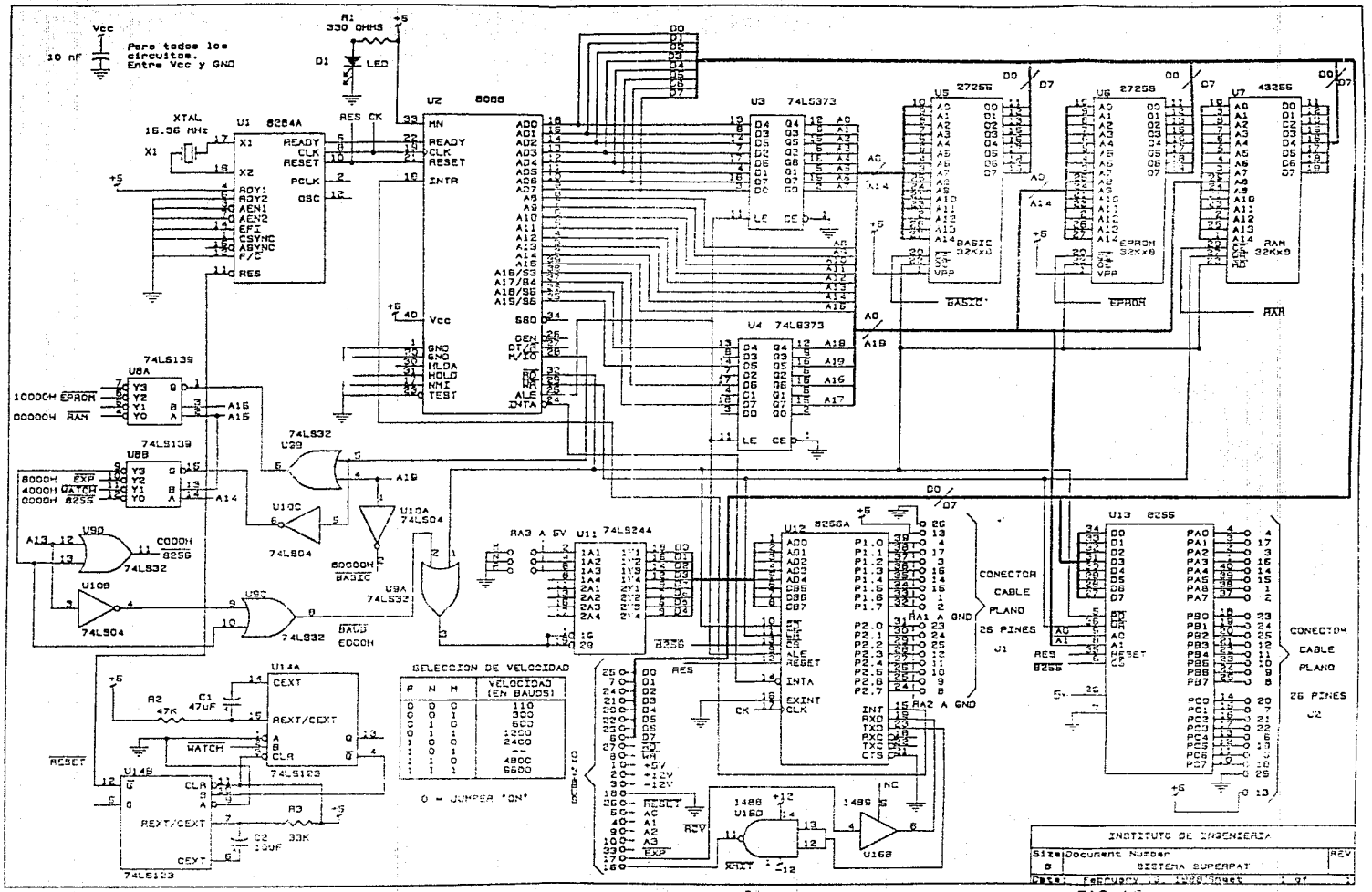

Vcc
10 nF
Para todos los circuitos. Entre Vcc y GND
R1 330 OHMS
+5
D1 LED
XTAL 16.36 MHz
U1 8284A
U2 8086
U3 74LS373
U4 74LS373
U5 27256
U6 27256
U7 43256
BASIC 32Kx8
EPROM 32Kx8
RAM 32Kx8
74LS139
74LS32
74LS04
74LS244
8255
8256A
74LS123
CONECTOR CABLE PLANO 26 PINES
GELECCION DE VELOCIDAD
VELOCIDAD (EN BAUDS)
0 = JUMPER "ON"
INSTITUTO DE INGENIERIA
SISTEMA SUPERPAT

FIG 16

NUMERO	CANTIDAD	REFERENCIA	PARTE
1	2	U5,U6	27256
2	1	U9	74LS32
3	1	U8	74LS139
4	1	C2	47uF
5	1	R3	33K
6	1	R2	47K
7	1	U7	43256
8	10	C	10nF
9	1	U1	8284A
10	1	U2	8088
11	1	U10	74LS04
12	1	U11	74LS244
13	1	U13	8255
14	1	U15	1488
15	1	U16	1489
16	3	M,N,P	JUMPER
17	1	X1	XTAL
18	1	U12	8256A
19	1	D1	LED
20	1	R1	330 OHMS
21	1	U14	74LS123
22	1	RA1	10K
23	1	RA2	10K
24	1	RA3	10K
25	2	U3,U4	74LS373
26	2	J1,J2	CONECTOR CABLE PLANO 26 PINES

TABLA 6 - Lista de componentes del Sistema SUPERPAT

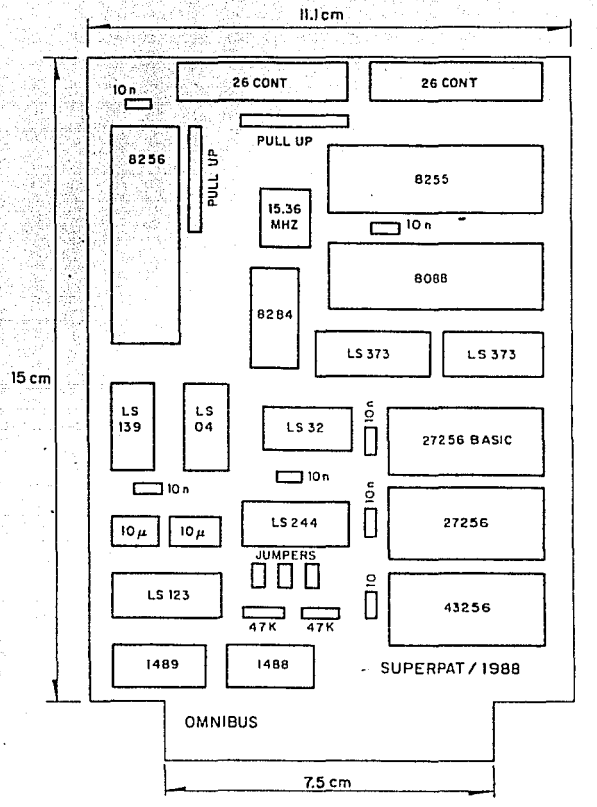


Fig 17 SUPERPAT lay out

5) SSP, DESCRIPCION DEL SOFTWARE

5.1) CARACTERISTICAS GENERALES

El software del sistema consiste de un lenguaje Basic (BASICM) de ejecución rápida, que permite programación de multitareas y está enfocado a aplicaciones de control de procesos en tiempo real.

BASICM es semejante al Basic de las computadoras personales, aunque con algunas mejoras para una ejecución más rápida y con un conjunto de instrucciones ampliado; pero no contiene nada respecto a manejo de archivos en medios de almacenamiento secundario.

BASICM, en combinación con su sistema de hardware complementario, está pensado para resolver problemas de control hallados en el ambiente industrial. Así, el diseñador del sistema raramente requerirá apoyarse en programas escritos en lenguaje ensamblador, lo que simplifica considerablemente el tiempo de desarrollo normalmente empleado para la integración de sistemas de control.

Los programas en BASICM se desarrollan tal como en otro Basic, por lo que el programador no nota diferencia. Sin embargo, internamente BASICM compila las líneas del programa tecleado a un lenguaje intermedio. Los programas en BASICM se ejecutan más rápido que en los lenguajes interpretados, debido al mejoramiento de la arquitectura y a la incorporación de este compilador "transparente". El lenguaje intermedio se ejecuta más rápido, y el proceso de traslación es

totalmente transparente para el programador.

BASICM siempre es interactivo. El programa siempre puede desplegarse y editarse. No tiene etapa de compilación separada, sólo se escribe y corre. La compilación "transparente" se hace conforme cada línea se teclea.

La mayoría de los intérpretes Basic permiten operaciones matemáticas tanto en números enteros como en punto flotante. Los cálculos matemáticos en punto flotante son más lentos que para los enteros. El control de procesos está involucrado principalmente con entrada y salida con variables externas, más que con extensivo procesamiento de números. Como BASICM fué diseñado para necesidades halladas en aplicaciones de control industrial, esto es, entrada/salida extensiva y velocidad, se implementó en él exclusivamente el manejo de números enteros.

BASICM usa una aritmética "complemento a 2" en 32 bits, lo que dá un rango de -2,147,483,648 a +2,147,483,647. También acepta longitudes de datos de 8 y 16 bits. Asume que todos los números son decimales, aunque se puede especificar un número en binario o hexadecimal. Contiene además instrucciones para conversión entre estas diferentes bases.

BASICM tiene un total de 120 comandos.

A un primer nivel, BASICM contiene 51 comandos estándares, hallados en la mayoría de los lenguajes Basic.

El siguiente nivel tiene una extensión de 56 comandos que fueron específicamente desarrollados para aplicaciones de control.

En el nivel final se encuentran 13 comandos de programa-

ción avanzada, que incluyen multitareas e interrupciones.

Un comando especial de BASICM ejecuta un programa MONITOR, el cual permite fácilmente el control de puertos, memoria, registros de CPU, así como la ejecución de programas en código máquina del procesador 8088 con definición de "breakpoints".

Una lista de los comandos de BASICM se encuentra en el apéndice 1, con una descripción breve de los mismos. El manual de BASICM [referencia 10] contiene información más detallada.

5.2) CARACTERISTICAS AVANZADAS

Una de las más importantes características de BASICM para aplicaciones en control y medición es la capacidad de realizar multitareas. En este esquema, cada tarea se ejecuta casi como si fuera un programa independiente.

BASICM es capaz de desarrollar hasta 16 tareas concurrentemente. Así por ejemplo, en una aplicación de control de manipuladores, una tarea de fondo puede monitorear switches de información, mientras el programa principal está controlando la herramienta de corte.

El control se transfiere de una a otra entre las tareas definidas, a la velocidad del microprocesador, así que muchas tareas parecen ejecutarse simultáneamente. Esta transferencia de tareas se desarrolla bajo control del programa de usuario, así que la frecuencia de ejecución de la tarea (o su "prioridad") queda determinada por el propio usuario.

La estructura de interrupción de usuario está soportada en paralelo con la estructura de multitareas. Las peticiones de interrupción de usuario se manejan al completar cada declarativa de BASICM, y tienen mayor prioridad que la petición de servicio de multitarea.

Antes de usar estas opciones, el programador debe tener pleno conocimiento del manejo de estas facilidades. El manual correspondiente [referencia 10] detalla el uso de los comandos para el manejo de multitareas y de interrupciones de usuario, en su sección de programación avanzada.

5.3) FORMATO DE ALMACENAMIENTO DE BASICM

BASICM tiene una estructura interna compleja. Conforme se desarrolla un programa, BASICM agrega información adicional que acelera la ejecución en el momento de la corrida. En otros lenguajes, las declarativas GOTO y GOSUB hacen lenta la ejecución por el tiempo considerable que se gasta buscando el número de línea destino. BASICM tiene un algoritmo de localización rápida basado en su formato de almacenamiento.

Toda línea del programa debe empezar con un número de línea (entre 1 y 65535). El texto del programa se almacena en RAM, exactamente en la forma en que se introduzca del teclado. Las líneas se almacenan en orden ascendente, según el número de línea y no según el orden en que se teclearon.

Un programa BASICM se almacena en RAM a partir de la dirección 0000:1800H, con el siguiente formato:

* el byte 1 y 2 contienen el número de la línea con la cual inicia el programa (para almacenamiento de datos de 2 bytes, siempre se almacena en el primer byte el valor menos significativo).
* el byte 3 y 4 contienen el valor de desplazamiento, en número de bytes, para llegar a la siguiente línea, contado a partir del inicio de la presente línea, es decir, a partir del byte 1.
* el byte 5 y 6 contienen el número de caracteres ASCII tecleados en dicha línea.

* los siguientes bytes contienen el código ASCII de los caracteres tecleados; la línea termina con un "Carriage Return" (código ODH), el "Line Feed" se ignora. Después le sigue el código intermedio creado por BASICM en el proceso de compilación, el cual le sirve para la ejecución de la línea.
* los tres siguientes bytes son ceros, indicando fin de línea.
* nuevamente aparecen 2 bytes que contienen el número de la línea (la siguiente) y se repite toda la secuencia
* al final de la última línea se agregan 6 ceros que indican fin de programa

La figura 18 muestra un ejemplo de este formato de almacenamiento para un programa BASICM.

```
 5 DATO=RND(-13)
10 DATO=RND(1)
15 DATO=DATO MOD 8888H
20 PR "EL DATO ES: ";:PR DATO
30 DELAY 150H
40 GOTO 10
```

direccion	codigo	interpretacion ASCII
0000:1800	05 00 26 00 0E 00 44 41 54 4F 3D 52 4E 44 28 2D	..&...DATO=RND(-
0000:1810	31 33 29 0D B0 1D 10 12 B6 0E 00 1B 0D 15 2B 26	13).....+&
0000:1820	15 25 29 00 00 00 0A 00 21 00 0C 00 44 41 54 4F	.%).....!...DATO
0000:1830	3D 52 4E 44 28 31 29 0D B0 1D 0D 12 B6 0E 00 1B	=RND(1).....
0000:1840	01 15 25 29 00 00 00 0F 00 2D 00 14 00 44 41 54	..%).....-...DAT
0000:1850	4F 3D 44 41 54 4F 20 4D 4F 44 20 38 38 38 38 48	O=DATO MOD 8888H
0000:1860	0D B0 1D 11 12 B6 0E 00 12 B6 0E 1A 88 88 15 4C	L
0000:1870	25 00 00 00 14 00 40 00 1B 00 50 52 20 22 45 4C	%.....@...PR "EL
0000:1880	20 44 41 54 4F 20 45 53 3A 20 22 3B 3A 50 52 20	DATO ES: ";:PR
0000:1890	44 41 54 4F 0D 30 18 14 13 0D 45 4C 20 44 41 54	DATO.0....EL DAT
0000:18A0	4F 20 45 53 3A 20 03 00 00 30 18 09 12 B6 0E 00	O ES: ...0.....
0000:18B0	18 00 00 00 1E 00 1A 00 0B 00 44 45 4C 41 59 20	.....DELAY
0000:18C0	31 35 30 48 0D 2E 21 07 1A 50 01 00 00 00 28 00	150H..!..P....(.
0000:18D0	16 00 08 10 47 4F 54 4F 20 31 30 0D 07 20 06 1A	....GOTO 10.. ..
0000:18E0	0A 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF	.....
0000:18F0	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	.....

FIG 18 - Ejemplo del formato de almacenamiento en RAM de un programa BASICM

6) LA COMPUTADORA PC COMO SOPORTE DEL SSP

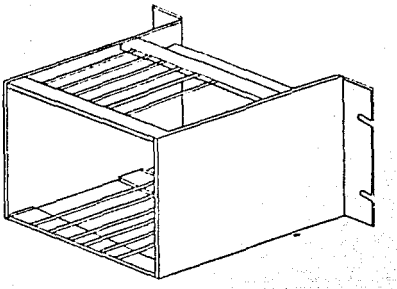
6.1) PUESTA EN OPERACION DEL SSP

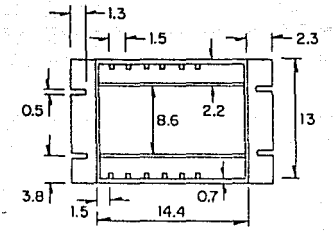
La tarjeta del módulo SUPERPAT se inserta en un portatarjetas como el mostrado en la figura 19, junto con una fuente que genera los voltajes de +5 volts a 1 A y +/- 12 volts a 200 mA.

En su parte posterior el portatarjetas tiene acoplado un conector DB25 para el puerto serie RS232. La asignación de pines para este conector es:

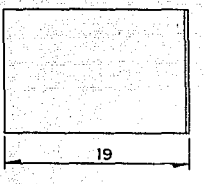
2	RCV	- línea de recepción de datos
3	XMIT	- línea de transmisión de datos
7	GND	- línea de tierra

A este conector debe acloplarse el cable que irá hacia la terminal, para implementar la conexión tal como se muestra en la figura 20.

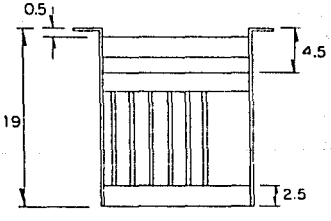




FRENTE



PERFIL



PLANTA

Acotaciones en cm

Fig 19 Portatarjetas PAT

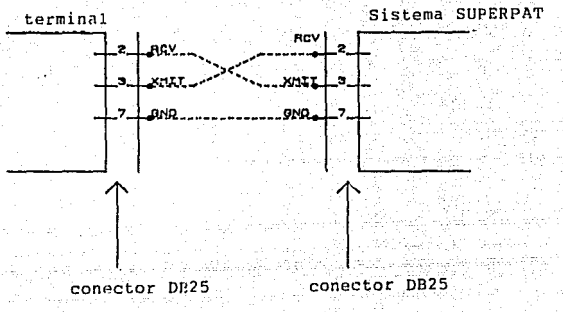


Fig 20 - Conexión terminal - Sistema SUPERPAT

El SSP puede operar a una de entre 7 diferentes velocidades de comunicación con la terminal. Esta velocidad se elige con los "jumpers" que se encuentran en la tarjeta SUPERPAT, según lo indica la tabla 7.

P N M	VELOCIDAD (EN BAUDS)
0 0 0	110
0 0 1	300
0 1 0	600
0 1 1	1200
1 0 0	2400
1 0 1	--
1 1 0	4800
1 1 1	9600

0=JUMPER PUESTO

TABLA 7 - Elección de la velocidad de comunicación del SSP

La terminal debe estar configurada conforme a los parámetros programados en el SSP para la comunicación serie. Estos parámetros se han puesto a valores comunes, y son:

1) un bit de inicio
2) 7 bits de datos paridad SPACE (fija a "0" ó a "1"), u 8 bits de datos sin paridad
3) dos bits de paro
4) conexión full-dúplex
5) velocidad de comunicación a 2400 bauds
6) poner terminal en línea (ON-LINE)

Teniendo la terminal configurada de esta manera, y conectada al SSP por su puerto serie, se puede entonces encender el sistema (primero la terminal y después el SSP).

Al encender el SSP, se ejecutará automáticamente el programa previamente grabado en la EPROM de usuario -si es que existe la EPROM insertada en el socket 1- ya que tiene la característica de AUTORUN. Si no se desea que esto ocurra, se debe oprimir la tecla ESC en la terminal dentro de los 2 segundo a continuación de haber encendido el SSP. Con esto, BASICM tomará el control del sistema, enviando los siguientes letreros y su prompt '>' :

```
SISTEMA SUPERPAT
BASIC MULTITAREA v1.1,1988
>
```

A partir de ese momento, el SSP está listo para aceptar cualquier comando de BASICM.

En lugar de una terminal común, se puede utilizar una computadora PC, lo cual incrementa considerablemente la facilidad de desarrollo de programas en Basic o en lenguaje ensamblador.

La computadora PC puede emplearse:

a) como terminal estándar RS232 para la comunicación y programación del SSP

b) para almacenamiento y edición de programas BASICM para el SSP

c) para desarrollo de programas en lenguaje ensamblador 8088 que se ejecutarán en el SSP

d) como programador de EPROMs para grabar tanto programas desarrollados en BASICM como en lenguaje ensamblador.

Todas estas funciones amplían las facilidades proporcionadas por el SSP, haciendo del binomio PC-SSP un sistema de desarrollo completo.

A continuación se describen detalladamente cada una de las posibilidades. El software necesario se encuentra en los discos flexibles etiquetados "SUPERPAT" y "MACROENSAMBLADOR", según se indique en cada caso.

6.2) EMULACION DE LA PC COMO TERMINAL DE VIDEO.

La PC se puede utilizar como terminal y comunicarse así con el SSP.

La máxima velocidad a la que puede comunicarse el SSP es de 9600 bits/segundo. Sin embargo no es recomendable usar esta velocidad máxima cuando se usa la PC como terminal debido a las limitaciones del puerto serie de la PC. Se recomienda usar 2400 bits/segundo si se requiere hacer transferencia desde o hacia disco de la PC, como más adelante se indica.

Existen en el mercado muchos paquetes de software de comunicaciones para la PC. Entre ellos, el que se utiliza aquí es el llamado VTERM, el cual permite emular una terminal VT100 o una VT52, que recibe y transmite datos a través del puerto serie de la PC, permitiendo definir los parámetros de la comunicación (velocidad, número de bits, etc).

Además dicho software permite facilidades tales como la transmisión de archivos tomados desde disco, o bien, almacenar en un archivo en disco una secuencia de caracteres recibidos. Estas capacidades de VTERM apoyan enormemente el uso de la PC como soporte para el SSP.

El procedimiento para usar la PC como terminal para el SSP es:

1.- Conectar el SSP al puerto serie de la PC (señales RCV, XMIT y GND), según la conexión indicada en la figura 20.

2.- Colocar el disco etiquetado "SUPERPAT" en la unidad

de discos A de la PC.

3.- Reinicializar la PC (encenderla o darle Reset). El disco contiene un archivos de comandos autoejecutable, el cual pide la fecha y hora actuales y después ejecuta VTERM. Con esto, la PC queda automáticamente funcionando en modo terminal. Estando en modo terminal, algunas teclas tienen alguna función especial (*): la tecla [F6] sirve para invocar ayuda sobre el manejo de VTERM; la tecla [F5] permite ver y cambiar los parámetros de la comunicación. Estos parámetros se cargan inicialmente desde un archivo llamado VTERM.SET, el cual contiene los valores de default indicados anteriormente (la velocidad se ha puesto a 2400 bits/segundo, la cual debe de coincidir con la velocidad programada en la tarjeta SUPERPAT; los demás parámetros no deben cambiarse). Se puede regresar a MS-DOS tecleando [Shift][Shift], y con esta misma secuencia se regresa al modo terminal si se está en MS-DOS.

4.- Encender el SSP. Se debe presionar la tecla [ESC] si no se desea ejecutar el programa de la EPROM de AUTO-RUN. Con ello, en el video de la PC -funcionando en modo terminal- debe aparecer el letrero inicial y el prompt de BASICM, con lo cual está lista para recibir comandos.

*NOTA: Para referenciar una tecla que debe ser presionada se usa la notación de poner el nombre de dicha tecla encerrado entre corchetes.

6.3) DESARROLLO DE PROGRAMAS EN BASIC

La capacidad de VTERM de transferir archivos permite desarrollar en la PC programas BASICM que se ejecutarán en el SSP. El procedimiento se describe a continuación:

1) Salir del modo terminal, tecleando [Shift][Shift], para regresar a MS DOS.
2) Usando cualquier editor de texto de los disponibles para PC, editar el programa Basic de acuerdo a la sintaxis de BASICM del SSP [referencia 10] y grabarlo como un archivo ASCII en disco.
3) Regresar al modo terminal tecleando [Shift][Shitf]
4) Poner los parámetros de transmisión correctos para la transferencia. La tecla [F5] permite ver y cambiar dichos parámetros, a través de varias pantallas. La primer pantalla se refiere a los parámetros básicos para la comunicación, puestos ya a un default. Tecleando nuevamente [F5] aparece una segunda pantalla, con los parámetros que se refieren a la transferencia de archivos. En esta pantalla, el primer campo pregunta por el nombre del archivo a transferir; ahí se debe teclear el nombre con el que se grabó el archivo Basic en disco. El segundo campo pregunta la dirección de transferencia, oprimiendo la tecla "+" se puede cambiar esta dirección, la cual debe indicar "SEND" para que el archivo sea enviado desde la PC hacia el SSP. Los demás parámetros ya están definidos y no deben cambiarse. Teclando [F5] o [ESC] se sale de

las pantallas de parámetros.

5) Realizar la transferencia. Una vez puestos los parámetros sólo se necesita teclear [ALT][T] y el programa se transferirá a la memoria del SSP, tal como si se hubiera tecleado desde terminal, por tanto, se irá desplegando en el video. Una vez transferido, el programa puede aceptar cualquier comando de BASICM, como edición de línea, ejecución, etc.

NOTA: Antes de transferir un programa se debe asegurar que no haya otro presente (tecleando el comando NEW) o ambos se encimarán.

El caso inverso de transferencia, desde SSP hacia un archivo en disco también es posible dentro de las facilidades de VTERM. El procedimiento es:

1) Usando la PC como terminal, teclear el programa BASICM en el SSP.

2) Poner los parámetros en la pantalla de transferencia de archivos. Nuevamente el primer campo es el nombre del archivo, pero ahora con el que se grabará en disco. Si el nombre del archivo especificado en este campo ya existe, se agregarán al final de él todos los datos transmitidos. Ahora el segundo campo (el de dirección) debe indicar "RECEIVE", porque la PC va a recibir ese archivo con datos desde el SSP. Los demás parámetros deben dejarse sin modificaciones.

3) Iniciar la transferencia. Para ello teclear:

LIST [ALT][T] [ENTER]

Con esto el programa se lista y todos los caracteres ASCII se van almacenando en la PC. Antes de iniciar la transferencia, VTERM checa si el archivo existe, y si es así, pregunta si se desean agregar al final de tal archivo los nuevos datos (en caso negativo se tendría que regresar al paso 2 a poner un nuevo nombre). Los datos recibidos se van almacenando en un buffer. Si se transmiten demasiados datos, el buffer se llena y automáticamente se graba su información en disco. Esta es la razón por la cual el SSP se programó a 2400 bauds, ya que al requerirse acceso a disco, velocidades mayores provocarían pérdidas de información.

4) Después que se ha listado el programa completo, se termina la transmisión. Para ello se teclea

[ALT][K]

Esto causa que los datos que aún se encuentren en el buffer se graben en disco y se cierre el archivo. De esta manera se tiene ahora un archivo accesable desde MS DOS.

6.4)DESARROLLO DE PROGRAMAS EN LENGUAJE ENSAMBLADOR 8088

La PC estándar utiliza precisamente el microprocesador 8088 como CPU. Como se ha desarrollado una gran cantidad de software para PC que facilita el desarrollo de programas en lenguaje ensamblador, se tiene entonces disponible una sólida infraestructura para tal efecto.

El proceso se puede dividir en los siguientes pasos:

1.- Edición del programa fuente

2.- Generación del código máquina

3.- Depuración y pruebas

4.- Carga y ejecución

Todo el software requerido en este proceso se encuentra en el disco etiquetado "MACROENSAMBLADOR", por lo que se requiere tener este disco presente para emplear los procedimientos que se describen abajo. En los ejemplos usados en esta descripción se considera que tal disco se encuentra en la unidad de discos B.

1) Edición.

El programa fuente en lenguaje ensamblador 8088 debe grabarse en un archivo, utilizando cualquier editor de texto de los disponibles para PC. En el disco se ha incluído el editor TURBO. La elaboración de este programa requiere de conocimientos previos del lenguaje máquina del 8088 y además del Macroensamblador que se utiliza para los siguientes pasos [referencia 11]. En el apéndice 2 se muestra un programa de ejemplo con el

formato apropiado con que debe crearse este archivo; también se incluye en el disco de trabajo, su nombre es SUPPAT.ASM

2.- Generación del código máquina.

Para generar el código máquina, el programa fuente creado se ensambla. Para ello se utiliza el Macroensamblador de MicroSoft [referencia 11], por eso el archivo fuente sigue cierta sintaxis. El comando para ensamblar es:

MASM archivo,,;

donde "archivo" es el nombre del archivo que contiene el programa fuente, el cual debe tener extensión .ASM Con este proceso se genera un archivo con el código objeto (extensión .OBJ) y uno con el listado del ensamblado (extensión .LST)

Ejemplo: B> MASM SUPPAT,,;

ensambla SUPPAT.ASM, genera SUPPAT.OBJ y SUPPAT.LST

A continuación se necesita ligar el archivo de código objeto para obtner un archivo de código ejecutable. Para ello se utiliza el ligador LINK de MS DOS. El comando es:

LINK archivo,,/MAP;

donde "archivo" es el nombre del archivo con el código objeto (.OBJ). Se genera un archivo ejecutable bajo MS DOS (extensión .EXE) y uno que contiene el mapa de los símbolos utilizados en el programa fuente (extensión .MAP).

Ejemplo: B> LINK SUPPAT,,/MAP;

liga SUPPAT.OBJ, genera SUPPAT.EXE y SUPPAT.MAP

3.- Depuración.

Para la depuración o etapa de prueba del programa se utiliza la utilería llamada SYMDEB, un "debug" o depurador simbólico [referencia 12]. El término "simbólico" significa que permite referenciar las direcciones e instrucciones por nombre, en lugar de por su valor numérico. Si se quiere utilizar esta característica se necesita:

a) haber declarado con "PUBLIC" los símbolos en el programa fuente (ver SUPPAT.ASM en apéndice 2).

b) generar un archivo de símbolos. Para esto el comando es:

MAPSYM archivo

donde "archivo" es el nombre del archivo de mapa de símbolos (.MAP), del cual se genera un nuevo archivo con la información de los símbolos (.SYM)

Ejemplo: B> MAPSYM SUPPAT

usa SUPPAT.MAP para generar SUPPAT.SYM

Para entrar al debug, el comando es:

SYMDEB arch.SYM arch.EXE

Ejemplo: B>SYMDEB SUPPAT.SYM SUPPAT.EXE

Descripción adicional de las opciones y de los comandos de SYMDEB se encuentran en el manual correspondiente [referencia 12].

4.- Carga y ejecución.

El código máquina del programa resultante de los procedimientos anteriores se puede cargar en RAM del SSP utilizando las facilidades del MONITOR de BASICM. Este código se puede obtener imprimiendo el desensamblado del programa a través del uso del debug. Otra opción sería grabar en EPROM dicho código, ya que el SSP tiene un socket disponible para una 27256.

6.5) PROGRAMACION DE EPROMS PARA PROGRAMAS DESARROLLADOS EN LENGUAJE ENSAMBLADOR

El código máquina correspondiente a un programa desarrollado en lenguaje ensamblador se puede grabar permanentemente en una EPROM. Para ello se utiliza un paquete comercial llamado EPROM WRITER EW-901.

Este paquete cuenta con una tarjeta de hardware diseñada para insertarse en un slot de expansión de la PC, y un software que permite fácil manejo de las diversas funciones con las que cuenta. Se pueden grabar varios tipos de EPROMs (2716, 2732, 2764, 27128, 27256 y 27512) sin necesidad de mover ningún switch; todo el control es vía software a través de menús, lo que resulta en gran facilidad de uso.

La instalación de la tarjeta y el uso del software se describen en el manual del programador, el cual se incluye en el apéndice 3. A la tarjeta insertada en el slot de expansión se le conecta externamente a la PC, a través de un cable plano, una base donde se inserta la EPROM.

De acuerdo al proceso descrito en la sección anterior, se obtuvo un archivo ejecutable bajo MS DOS (.EXE), el cual contiene el código máquina como resultado de ensamblar el programa fuente original (.ASM). Sin embargo, este archivo ejecutable contiene además algunos bytes iniciales de "encabezado de archivo" que usa MS DOS como información para cargarlo apropiadamente en memoria.

Para obtener un archivo que contenga únicamente el código máquina del programa fuente, se usa la utilierfa

EXE2BIN, de MS DOS. Pero para poder utilizarla se requiere que el programa fuente cumpla con ciertas características [referencia 13]. El programa de ejemplo SUPPAT.ASM cumple con estas condiciones.

El formato para ejecutar este comando es:

EXE2BIN archivo.EXE

el cual procesa un archivo ejecutable (de extensión .EXE) para generar un archivo binario (extensión .BIN).

Ejemplo: B>EXE2BIN SUPPAT.EXE

procesa SUPPAT.EXE para generar SUPPAT.BIN.

Si el archivo fuente no cumple con las condiciones para utilizar EXE2BIN, como puede ser, por ejemplo, que tenga un origen diferente de 0 ó 100H, entonces otra opción para generar el archivo binario es utilizar las facilidades de SYMDEB a través de su comando W [referencia 12]: se carga el archivo .EXE en el debug y se graba desde ahí como un archivo .BIN.

El archivo de extensión .BIN es el que se usará como entrada para el programador de EPROMs.

Una vez que se tiene listo este archivo, se ejecuta el software del programador, el cual se encuentra en el disco "SUPERPAT", tecleando:

A>UPP512

Con ello aparece el menú de opciones para realizar el grabado de la memoria. Para la descripción del uso de este software, consultar el manual en el apéndice 3.

A continuación se muestra a manera de ejemplo el proceso seguido en particular para grabar una EPROM 27256 con el archivo SUPPAT.BIN (lo que se subraya es lo que debe teclear

el usuario):

a) A> UPP512 (se inicia desde MS DOS)

b) E (EPROM TYPE)

c) 9 (27256/12.5 V)

d) L (LOAD DISK)

STARTING ADDRESS=0000

KEY IN FILE NAME?SUPPAT.BIN

e) C (COPY)

f) Q (QUIT)

A> (se regresa a MS DOS)

La memoria una vez programada puede insertarse en el socket 1 del SSP (socket de la EPROM de usuario). Cabe aclarar que el código que lee de ahí el software del sistema, es interpretado como código BASICM si se deja ejecutar en modo AUTORUN (este código no debe dejarse ejecutar en AUTORUN, ya que se interpretará y ejecutará diferente a lo que se programó, porque no es código BASICM).

La EPROM programada también puede insertarse en el socket 0 del SSP (socket del software del sistema). En este caso el programa de usuario grabado tomará el control total del sistema si es que se quita el software BASICM. Si se desea combinar el software de BASICM con alguna rutina de usuario (en el chip de memoria hay espacio para ello porque no está ocupada por BASICM toda el área de 32k), se usan las facilidades del programador de EPROMs y del debug para modificar el contenido de la ROM.

6.6) PROGRAMACION DE EPROMS PARA PROGRAMAS BASICM

Para el proceso de grabación de programas BASICM en EPROM se utiliza también el programador disponible en la PC.

Tal como se describió en la sección 5.3, el SSP tiene un formato propio para almacenamiento de los programas Basic en RAM. También usa un formato preestablecido para almacenar un programa Basic en EPROM, ya que tiene que almacenar, además del texto propio del programa (lo que teclea el usuario) y su código intermedio generado, cierta información de control e información sobre las variables del programa, indispensables para poder inicializar correctamente su área de RAM donde se manipulan los valores de dichas variables.

En el apéndice 4 se describe detalladamente el formato de almacenamiento de datos en EPROM para programas en BASICM.

Tal como se explica en ese apéndice, la información requerida para grabarse en EPROM se encuentra en RAM, en distintas localidades. El proceso de grabación involucra primero la obtención de tal información. Para ello se despliega en terminal, en su valor ASCII para tener un despliegue entendible, toda la información que debe grabarse en EPROM, tomada directamente del área de RAM donde se encuentra. Ese despliege se recibe en un archivo en disco en la PC, aprovechando las facilidades de VTERM.

El segundo paso consiste en procesar dicho archivo en la PC para obtener la información en su valor binario original, en otro archivo.

Este último archivo binario obtenido es el que final-

mente se utiliza como entrada para el programador de EPROMs de la PC.

Para realizar el primer paso se implementó un programa que despliega en terminal el valor ASCII hexadecimal de los datos necesarios para grabarse en EPROM. Esta rutina se programó en lenguaje ensamblador y se grabó en un espacio de la ROM de BASICM, ya que no todo el espacio de memoria del chip está ocupado. La dirección dentro de la ROM a partir de la cual se grabó esta rutina es la 6000H, por tanto su dirección en el espacio total de memoria en el SSP es F800H:6000H, en la dirección física FE000H. El proceso para desarrollar y grabar esta rutina es exactamente el descrito en las secciones 6.4 y 6.5 de este mismo capítulo.

En el apéndice 5 se muestra el programa desarrollado para implementar este despliegue en la terminal.

Para el paso de procesar el archivo con datos ASCII para obtener un archivo con datos binarios, se desarrolló un programa en PC (bajo MS DOS). Este programa se podía implementar en cualquier lenguaje de alto nivel; se implementó en lenguaje "C". En el apéndice 6 se muestra el listado de dicho programa, llamado ASCBIN (convertidor de ASCii a BINario).

De acuerdo a este soporte realizado, el procedimiento para grabar programas desarrollados en BASICM es:

1) Usando la PC en modo terminal, cargar en el SSP el programa BASICM que se desea grabar, ya sea tecleándolo directamente o transfiriéndolo desde un archivo en disco de la PC.

2) preparar los parámetros de transferencia de archivos para recibir datos en un archivo (dirección="RECEIVE") tal como se describió en la sección 6.3.

3) Teclear el comando de BASICM:

   LINK 0F8006000H [ALT][T] [RETURN]

   ya que en la dirección F800:6000H se encuentra la rutina que raliza el despliegue ASCII en terminal de los datos y [ALT][T] inicia en la PC la recepción.

4) Al finalizar el desplegado de los datos, terminar la transferencia de información hacia el archivo en PC, tecleando:

   [ALT][K]

   Con ello, se cierra el archivo en PC que contiene toda la información desplegada.

5) Para obtener el valor binario de esos datos, en el disco etiquetado "SUPERPAT" se encuentra el programa ASCBIN. Para ejecutarlo se teclea:

   ASCBIN archivo

   donde "archivo" es el nombre del archivo en el cual se recibieron los datos. ASCBIN crea un archivo con el mismo nombre, pero con extensión .BIN, el cual contiene exactamente el código a grabarse en la EPROM.

6) Una vez que se tiene listo el archivo a grabar, se ejecuta el software del programador, tecleando

A>UPP512

y se procede a grabar la EPROM con los datos del archivo de extensión .BIN recién creado, usando los menús apropiados [apéndice 3].

La memoria programada debe insertarse en el socket 1 del SSP. Al encender el sistema se ejecutará automáticamente el programa BASICM ahí grabado.

A continuación se muestra un ejemplo de este procedimiento, hasta la obtención del archivo binario listo para grabarse en EPROM (el desplegado de este archivo se muestra con la ayuda del debug, ya que es un archivo binario).

```
>LIST
    5 DATO=RND(-13)
   10 DATO=RND(1)
   15 DATO=DATO MOD 8888H
   20 PR "EL DATO ES: ";:PR DATO
   30 DELAY 150H
   40 GOTO 10


>LINK 0F8006000H

       * Codigo objeto para la EPROM : *
 FF 0C 00 FF 7F E4 13 FF 7F FF FF FF FF FF FF
 FF FF 04 44 41 54 4F FF FE FF 9C 73 00 00 05
 00 26 00 0E 00 44 41 54 4F 3D 52 4E 44 28 2D
 31 33 29 0D B0 1D 10 12 B6 0E 00 1B 0D 15 2B
 26 15 25 29 00 00 00 0A 00 21 00 0C 00 44 41
 54 4F 3D 52 4E 44 28 31 29 0D B0 1D 0D 12 B6
 0E 00 1B 01 15 25 29 00 00 00 0F 00 2D 00 14
 00 44 41 54 4F 3D 44 41 54 4F 20 4D 4F 44 20
 38 38 38 38 48 0D B0 1D 11 12 B6 0E 00 12 B6
 0E 1A 88 88 15 4C 25 00 00 00 14 00 40 00 1B
 00 50 52 20 22 45 4C 20 44 41 54 4F 20 45 53
 3A 20 22 3B 3A 50 52 20 44 41 54 4F 0D 30 18
 14 13 0D 45 4C 20 44 41 54 4F 20 45 53 3A 20
 03 00 00 30 18 09 12 B6 0E 00 18 00 00 00 1E
 00 1A 00 0B 00 44 45 4C 41 59 20 31 35 30 48
 0D 2E 21 07 1A 50 01 00 00 00 28 00 16 00 08
 10 47 4F 54 4F 20 31 30 0D 07 20 06 1A 0A 00
 00 00 00 00 00 00 00 00
```

Este desplegado que se obtiene al ejecutar el comando LINK se recibe en un archivo, que en este caso se llamó PRUE.COD
El proceso continua con el siguiente comando:

```
A>ascbin prue.cod

Convertidor de codigo ASCii a BINario              version 1.0, 1988
Programo':  Alberto  Aduna  Grajeda


El archivo a procesar es prue.cod
.................
Proceso termina. Se genero el archivo prue.bin
```

```
debug prue.bin
-r
AX=0000  BX=0000  CX=0107  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B1C  ES=0B1C  SS=0B1C  CS=0B1C  IP=0100   NV UP DI PL NZ NA PO NC
0B1C:0100 FF0C          DEC     WORD PTR [SI]                    DS:0000=20CD
-d 100 2cf
0B1C:0100  FF 0C 00 FF 7F E4 18 FF-7F FF FF FF FF FF FF FF   .....d..........
0B1C:0110  FF 04 44 41 54 4F FF FE-FF 9C 73 00 00 05 00 26   ..DATO.~...s....&
0B1C:0120  00 0E 00 44 41 54 4F 3D-52 4E 44 28 2D 31 33 29   ...DATO=RND(-13)
0B1C:0130  0D B0 1D 10 12 B6 0E 00-1B 0D 15 2B 26 15 25 29   .0...6.....+&.%)
0B1C:0140  00 00 00 0A 00 21 00 0C-00 44 41 54 4F 3D 52 4E   .....!...DATO=RN
0B1C:0150  44 28 31 29 0D B0 1D 0D-12 B6 0E 00 1B 01 15 25   D(1).0...6.....%
0B1C:0160  29 00 00 00 0F 00 2D 00-14 00 44 41 54 4F 3D 44   ).....-...DATO=D
0B1C:0170  41 54 4F 20 4D 4F 44 20-38 38 38 48 0D B0 1D      ATO MOD 8888H.0.
0B1C:0180  11 12 B6 0E 00 12 B6 0E-1A 88 88 15 4C 25 00 00   ..6...6.....L%..
0B1C:0190  00 14 00 40 00 1B 00 50-52 20 22 45 4C 20 44 41   ...@...PR "EL DA
0B1C:01A0  54 4F 20 45 53 3A 20 22-3B 3A 50 52 20 44 41 54   TO ES: ";:PR DAT
0B1C:01B0  4F 0D 30 18 14 13 0D 45-4C 20 44 41 54 4F 20 45   O.0....EL DATO E
0B1C:01C0  53 3A 20 03 00 00 30 18-09 12 B6 0E 00 18 00 00   S: ...0...6.....
0B1C:01D0  00 1E 00 1A 00 0B 00 44-45 4C 41 59 20 31 35 30   .......DELAY 150
0B1C:01E0  48 0D 2E 21 07 1A 50 01-00 00 00 26 00 16 00 08   H..!..P....&....
0B1C:01F0  10 47 4F 54 4F 20 31 30-0D 07 20 06 1A 0A 00 00   .GOTO 10.. .....
0B1C:0200  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-q

A:\>
```

6.7) SSP-PC, UN SISTEMA DE DESARROLLO COMPLETO

El SSP es una microcomputadora programable que permite el desarrollo de software para las diferentes aplicaciones. El uso de la PC aumenta las posibilidades del SSP al incorporar: a) el uso de editores para crear programas y de medios de almacenamiento masivo para guardarlos; b) soporte para programación del SSP en lenguaje ensamblador; y c) soporte para grabación de programas en EPROM.

De esta manera, el binomio Sistema SUPERPAT-Computadora PC (SSP-PC) se consolida como una firme herramienta para el desarrollo de sistemas de control de procesos industriales, de bajo costo y fácil programación.

La figura 21 muestra el esquema del SSP-PC como un sistema de desarrollo completo.

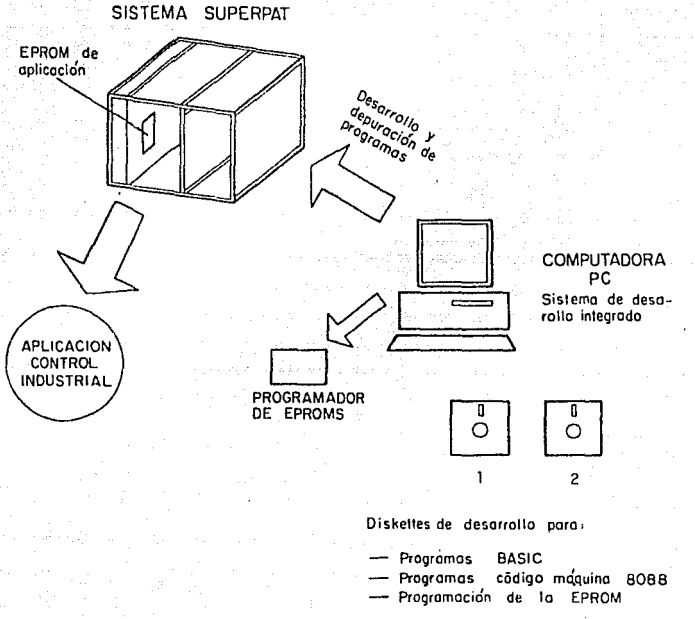


Fig 21 - Sistema de desarrollo para aplicaciones de control industrial

7) MODULOS DE PRUEBA Y EXPANSIONES FUTURAS

7.1) PRUEBAS GENERALES DEL SISTEMA SSP-PC

Para probar el correcto funcionamiento del SSP, tanto del software (BASICM) como del hardware (a través de probar los puertos de entrada/salida), se desarrolló un módulo de despliegues BCD y programas de prueba en BASICM para manejarlo. Además, estos programas sirvieron para probar el soporte de desarrollo creado utilizando la computadora PC, desde cargar el programa en la RAM del SSP hasta grabarlo en EPROM y tenerlo funcionando en modo AUTORUN.

El módulo de despliegues BCD desarrollado consta de 4 despliegues de 7 segmentos, manejados por 4 drivers codificadores binario a BCD, 7447, según se muestra en la figura 22.

Para su interfaz con el SSP, el módulo tiene un conector de 26 pines que se comunica, a través de cable plano de 26 líneas, directamente al conector J1 ó J2 del SSP. De conectarse a J1, el módulo BCD se maneja a través de los puertos A y B del circuito 8255. Si se conecta a J2, el módulo se maneja a través de los 2 puertos paralelos (P1 y P2) del circuito 8256.

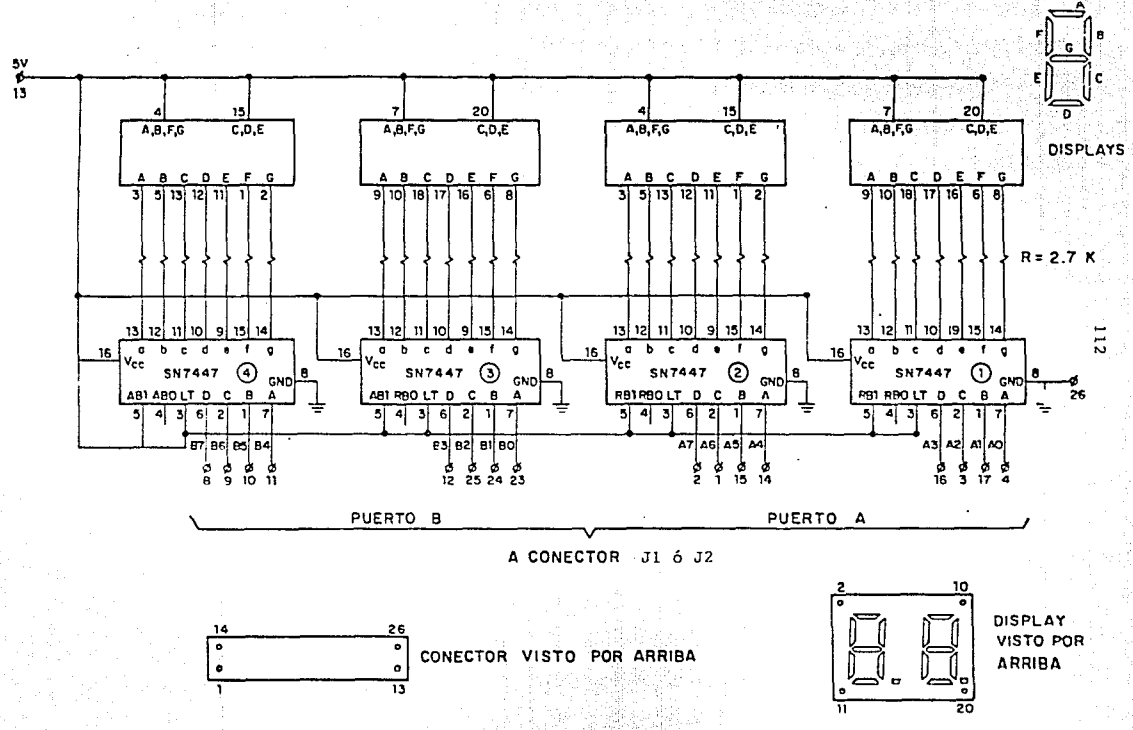


Fig 22 Modulo para despliegue BCD

De esta forma se tiene un circuito de prueba rápido para la verificación del funcionamiento de los puertos del SSP.

Dentro de un sistema de control industrial, este módulo podría emplearse para mostrar el valor de alguna variable, del set-point, para la cuenta de eventos, programación de opciones, etc; constituyendo un módulo de interface con el operador (figura 2).

El apéndice 7 muestra el programa desarrollado para probar el módulo de despliegues, usando los puertos paralelos del 8256, llamado PRO.BAS . Los registros del 8256 (tabla 3) se programan para configurar los puertos en modo salida [referencia 6].

El programa se desarrolló en el editor TURBO de PC, y se transfirió a la RAM del SSP con el procedimiento indicado en 6.3

Teniéndolo presente en la memoria del SSP, se procedió a grabarlo en EPROM, de acuerdo al procedimiento descrito en 6.6, de la siguiente manera:

a) se define en los parámetros de transferencia de archivo de VTERM el modo recepción ("receive") en un archivo que en este caso se llamó PRO.COD

b) estando la PC en modo terminal se teclea:

   LINK 0F8006000H [ALT][T] [ENTER]

   con lo cual se lista y se transfiere hacia la PC el código ASCII de los datos que se requieren grabar en la EPROM. Al terminar el desplegado se teclea [ALT][K] con lo que se cierra el archivo PRO.COD en disco.

c) Se genera el archivo binario. Para ello se teclea, bajo MS DOS:

A> ASCBIN pro.cod

con esto se genera el archivo PRO.BIN . Este archivo contiene el código para grabarse en la EPROM.

d) se coloca un chip 27256 en el socket del programador de la PC y se ejecuta el software UPP512, con las opciones para grabar los datos del archivo PRO.BIN

e) se coloca la EPROM grabada en el socket 1 del SSP.

Después de realizar este proceso, al encender el SSP o teclear el comando AUTORUN, el programa se ejecuta automáticamente.

El archivo PRO.COD y PRO.BIN también se muestran en el apéndice 7.

De esta manera se probó tanto el software y hardware del SSP, como el soporte de desarrollo en PC. Los resultados fueron positivos. Por tanto, el sistema completo SSP-PC funciona correctamente.

7.2) EXPANSIONES FUTURAS

Tal como se partió originalmete en el diseño, la microcomputadora SUPERPAT es sólo un módulo dentro de la implementación completa de un sistema de control digital. Para utilizarlo en alguna aplicación específica, requiere de los módulos de interface necesarios para dicha aplicación. Estos módulos pueden ser, por ejemplo, un módulo de adquisición de datos, un módulo para comunicaciones, un módulo para manejar cargas de potencia, etc.

De esta manera se pueden implementar de manera flexible y fácil diversos sistemas para la automatización industrial.

8) CONCLUSIONES

1.- La electrónica se caracteriza por su progreso técnico acelerado, por lo que se requiere una contínua tarea de actualización. En este proyecto se buscó la actualización de los controladores industriales que se han venido desarrollando en la Coordinación de Automatización del Instituto de Ingeniería de la UNAM.

Para ello se desarrolló la microcomputadora SUPERPAT (SSP), programable en un ambiente multitasking y orientada a aplicaciones de control industrial. Con el sólido soporte de desarrollo -utilizando la computadora PC- que se incorporó al SSP, se logró un sistema de desarrollo completo para la creación de sistemas de control industrial.

Este sistema se probó (capítulo 7), obteniéndose resultados positivos. De acuerdo a estos resultados, se puede concluir que los objetivos planteados inicialmente fueron alcanzados totalmente.

2.- La utilización principal de las microcomputadoras es como computadoras personales. En esta área existe una dependencia tecnológica y económica muy grande de nuestro país hacia el extranjero: las grandes compañías transnacionales tienen el control absoluto del mercado y de los desarrollos tecnológicos, a tal grado que en nuestro país no hay desarrollo de computadoras.

La segunda utilización en importancia de las microcomputadoras es como controladores industriales. En este campo sí existe la posibilidad de desarrollar nuestros propios equipos adaptados a nuestras necesidades; aunque actualmente existe también una gran dependencia en esta área: se importa del extranjero la mayoría del equipo electrónico de control que requiere la industria.

Sin embargo, las investigaciones en el área electrónica en general, deben continuarse e intensificarse, partiendo de que la electrónica se relaciona con todas las áreas de desarrollo del país.

En este sentido, el SSP constituye un desarrollo tecnológico en el área particular de los controladores industriales. El SSP puede utilizarse para resolver problemas concretos de automatización en la industria nacional. Sus características técnicas, aunado a su bajo costo (comparado con los equivalentes comerciales) hacen factible su incorporación al mercado. La interacción UNAM-Industria permitirá que en un futuro surgan los proyectos para hacer esto posible. Se pretende a un mediano plazo la transferencia de tecnología hacia empresas nacionales receptoras.

3.- El SSP es útil para implementar diversos sistemas digitales, no sólo de control de procesos, sino también otros, tales como sistemas de transmisión de datos, de adquisición y procesamiento de señales, de robótica, etc.

El enfoque modular con que se creó lo hace un sistema versátil y por tanto útil en muchas aplicaciones.

De esta manera, el sistema SSP-PC es un sistema de desarrollo completo que permite implementar diversos sistemas electrónicos basados en el microprocesador, de bajo costo y fácil programación.

4.- Como los cambios tecnológicos se suceden aceleradamente, algunos equipos se hacen obsoletos en poco tiempo. No es el caso del SSP. Sus características técnicas, como el ser un sistema multifunciones que soporta una gran variedad de aplicaciones y su enfoque modular pensado para readaptarse a diferentes necesidades, permiten afirmar que es un equipo que será válido durante varios años.

9) REFERENCIAS

1.- Noyce R. N. y Hoff M.C., "A history of Microprocessors Development at Intel", IEEE Micro, Vol. 1, Núm. 1, Febrero 1981, págs. 8-21

2.- Andrews G.R. y Schneider F.B., "Concepts and Notations for Concurrent Programming", Computing Surveys, Vol. 15, Núm. 1, Marzo 1983, págs. 3-43

3.- Freedman A., Glosario de Computación, McGraw Hill, México, 1984

4.- Hartmann C.A. y Fehr S., "A VLSI Architecture for Software Structure: The Intel 8086", IEEE Micro, Mayo 1981, págs. 57-69

5.- Intel, "8255A/8255A-5 Programmable Peripheral Interface", en Microsystem Component Handbook - Microprocessor and Peripheral, Vol. II, Intel Corporation, Sta. Clara California, U.S.A., 1985, págs. 5-273 a 5-293

6.- Intel, "Aplication Note AP-153: Designing with the 8256", op.cit., págs. 5-353 a 5-405

7.- Intel, "8088/8088-2 8 bit HMOS Microprocessor", op.cit., Vol. I, págs. 3-79 a 3-105

8.- Intel, "8284A/8284A-1 Clock Generator and Driver for 8086,8088 Microprocessor", op.cit., Vol I, págs. 3-237 a 3-244

9.- Texas Instruments, The TTL Data Book, segunda edición, Texas Instruments Incorporated, Texas, U.S.A., 1981

10.- Coordinación de Automatización II-UNAM, Manual BASICM del Sistema SUPERPAT, 1988

11.- MicroSoft, MacroAssembler Reference Manual, MicroSoft Corporation, U.S.A., 1985

12.- MicroSoft, "SYMDEB: A Symbolic Debug Utility", en MacroAssembler User's Guide, MicroSoft Corporation, U.S.A., 1985, págs. 73-163

13.- Jourdain R., "Convert programs from .EXE to .BIN type", en Programmer's Problem Solver for IBM PC, XT & AT, Brady Communications Company Inc., U.S.A., 1986, págs. 41-44

A P E N D I C E S

APENDICE 1: LISTA DE COMANDOS DE BASICM

1) COMANDOS ESTANDARES

ABS	Regresa el valor absoluto de un número
ASC	Regresa el código ASCII del primer caracter en una cadena de caracteres o string (*).
CHR$	Regresa un caracter, dado un valor numérico
CLEAR	Pone en ceros todas las variables
CONT	Continúa la ejecución del programa después de un BREAK
DATA	Permite definir datos en un programa
DELETE	Borra líneas del programa
DIM	Declara el tamaño máximo de un arreglo o un string
EDIT	Va al editor de línea
END	Detiene la ejecución de un programa
ERL	Regresa el número de línea en la cual ocurrió un error
ERR	Regresa el código de error del último error que ocurrió
ERROR	Causa una condición de error
FOR/NEXT	Estructura para programar un ciclo iterativo
FREE	Despliega la cantidad de memoria disponible

* NOTA: se usará el término "string" para denominar a una cadena de caracteres.

GOSUB/RETURN Declara una llamada a una subrutina y su regreso de ella

GOTO Declara un salto incondicional en el programa

IF/THEN/ELSE Declara un salto condicional en el programa

INKEY$ Regresa un caracter del puerto serie

INP Regresa un dato de un byte del puerto direccionado

INPUT Regresa un dato del puerto serie

LEFT$ Regresa, a partir de un string, una subcadena de caracteres justificada a la izquierda.

LEN Regresa la longitud de un string

LIST Lista el programa por el puerto serie

MID$ Regresa una subcadena de un string

MOD Regresa el residuo de una división

NEW Reinicializa el espacio del programa

NULL Escribe códigos ASCII NULL al puerto serie

ONERR Habilita el seguimiento de errores

ON/GOSUB/GOTO Salto condicional indexado por una variable

OUT Escribe un dato de un byte al puerto direccionado

PEEK Regresa un dato de un byte desde una dirección de memoria

POKE Escribe un dato de un byte hacia una dirección de memoria

POS Regresa la posición de una subcadena dentro de un string

PRINT Escribe un dato al puerto serie

PWR Eleva una constante a una potencia

READ Asigna el dato de una declarativa DATA a una variable

REM	Inserta comentarios en un programa
RENUM	Reenumera las líneas de un programa
RESTORE	Reestablece el apuntador de DATA
RIGTH$	Regresa un string justificado a la derecha
RND	Regresa un número pseudo-aleatorio
RUN	Causa que el programa se ejecute
SGN	Regresa el signo de un número
SQR	Regresa la raíz cuadrada entera de un número
STOP	Detiene la ejecución del programa
STR$	Regresa la representación en string de un número
TRACE	Ejecución del programa "paso a paso"
VAL	Regresa la representación numérica de un string con caracteres ASCII de números

2.- COMANDOS EXTENDIDOS

AIN	Regresa el resultado de una conversión A/D
AOT	Escribe dato a un conversor D/A
AUTORUN	Ejecuta el programa grabado en la EPROM de AUTORUN
BCD	Regresa la representación BCD de un número
BIN	Regresa el valor binario de un número BCD
BIN$	Regresa un string binario (*) de un dato tipo byte

* NOTA: se usará el término "string binario" para referirse a un string que representa el valor de un número usando caracteres ASCII del "1" y del "0". De manera semejante se usará el término "string hexadecimal".

BREAK	Inserta un "break" en un programa y despliega variables
CIN$	Regresa un string desde el puerto serie
CLEARPORT	Pone en ceros cualquiera o todos los puertos de E/S
DBIN$	Regresa el string binario de un dato tipo word (16 bits)
DEC	Decremento rápido de una variable
DELAY	Crea un intervalo de retardo en incrementos de milisegundos
DHEX$	Regresa una cadena hexadecimal de un dato tipo word
DINP	Regresa un dato tipo word de 2 direcciones de puertos
DISPLAY$	Envía una cadena de caracteres a un puerto de E/S
DO/UNTIL	Programación de un ciclo iterativo estructurado
DOUT	Escribe un dato tipo word a 2 direcciones de puertos
DPEEK	Regresa un dato tipo word desde 2 direcciones de memoria
DPOKE	Escribe un dato tipo word hacia 2 direcciones de memoria
DUMP	Despliega tabla de variables
EXIT	Limpia el stack para salida prematura de una estructura iterativa
FREQ	Regresa la frecuencia de una señal periódica que se lee de un puerto de E/S
HEX$	Regresa una cadena hexadecimal de un dato tipo byte
INC	Incremento rápido de una variable

IN$	Convierte los datos de direcciones sucesivas de E/S en una cadena de caracteres
LINK	Ejecuta una rutina de código máquina suministrada por el usuario
MENU	Entra al Monitor residente y despliega su menú
MGET	Regresa un dato tipo byte de cualquiera de las 64K direcciones del espacio de E/S
MON	Llama al Monitor residente
MPUT	Escribe un dato tipo byte a cualquiera de las 64 k direcciones del espacio de E/S
OBJET	Escribe datos tipo byte de una declaración DATA hacia RAM
OUT$	Escribe un string a direcciones sucesivas de memoria
PACK	Elimina el código fuente del programa (queda sólo el código de la compilación)
PEEK$	Regresa un string desde una localidad absoluta de memoria
PLOAD	Transfiere un programa grabado en la EPROM de AUTO-RUN hacia RAM
POKE$	Escribe un string a partir de una localidad absoluta de memoria
PULSE	Regresa el ancho del pulso de una señal que se lee de un puerto de E/S
QBIN$	Regresa un string binario de un dato tipo doble word (32 bits)
QHEX$	Regresa un string hexadecimal de un dato tipo doble word

QINP	Regresa un dato tipo doble word desde 4 direcciones de puertos
QOUT	Escribe un dato tipo doble word hacia cuatro direcciones de puertos
QPEEK	Regresa un dato tipo doble word desde 4 direcciones de memoria
QPOKE	Escribe un dato tipo doble word hacia 4 direcciones de memoria
RBIT	Pone en cero un bit de un puerto especificado
RECOVER	Recupera los apuntadores de un programa BASICM
REMOVE	Remueve las entradas a la tabla de variables
RESET	Escribe ceros a bits individuales en una dirección de E/S
SBIT	Pone en "1" un bit en un puerto de E/S
SET	Escribe 1's en bits individuales en un puerto de E/S
SRAM	Almacena un programa autorun en RAM
SWAP	Intercambia los valores de 2 variables numéricas
TACH	Regresa el equivalente en RPM de una señal que se lee de un puerto de E/S
TBIT	Regresa el status de un bit de un puerto de E/S
TEST	Regresa el resultado de aplicar una máscara sobre el dato de un puerto de E/S
TIMER	Crea hasta 16 salidas temporizadas independientes
VARPTR	Regresa la dirección donde reside una variable

3.- COMANDOS AVANZADOS

RELOJ POR SOFTWARE

GDATE Asigna la fecha a una variable

GTIME Asigna lo hora actual a 3 variables

SDATE Escribe el dato de la fecha actual al reloj de tiempo real de BASICM

STIME Pone la hora en el reloj de tiempo real

TIME$ Regresa un string que representan el dato de la hora del reloj de tiempo real

MULTITASKING

ADDTASK Agrega una tarea a la cola de multitasking

CLEARTASK Remueve una tarea de la cola de multitasking

ENDTASK Provoca que una tarea finalice su ejecución

SKIPTASK Rota las tareas en la cola de multitasking

WAITASK Causa que la siguiente tarea en la cola se ejecute

INTERRUPCIONES

CLEARITR Limpia los saltos de interrupción declarados por ONITR

ONITR Declara un salto en la ocurrencia de una interrupción

IRET Regreso de una interrupción

APENDICE 2: LISTADO DEL PROGRAMA SUPPAT.ASM

```
;suppat.asm - Programa de ejemplo de la sintaxis de un programa fuente
;en lenguaje ensamblador para ejecutarse en el Sistema SUPERPAT

;                                   (c) Alberto Aduna G.          1988


; Aqui se describe el formato necesario para escribir un programa
;en lenguaje ensamblador utilizando el Macroensamblador MASM
;como el primer paso en el proceso de programacion del microprocesador
;8088 en el SSP.
; Para mayor información sobre este formato, consultar el manual
;"MicroSoft MacroAssembler Reference Manual"

        ;se pueden declarar al inicio las constantes a utilizar
        NDATOS      EQU     10

        ; Si se piensa usar el Debug simbolico en la depuracion poste-
        ;rior del programa, entonces se deben declarar con PUBLIC to-
        ;das las etiquetas del programa que se usaran en dicha depura-
        ;ción. Las etiquetas marcadas como SEGMENT no se declaran:

PUBLIC          INICIO,INICOD,PROCE1,PROCE2,SLOOP,NXTNUM

        ; Enseguida se declara el inicio de segmento, con
        ;cualquier nombre como etiqueta seguido del pseudo-op SEGMENT:

CODIGO          SEGMENT

        ; Para indicar al ensamblador que asuma este segmento como
        ;el segmento de codigo, se requiere la siguiente pseudo-op
        ;(en este segmento residen todas las partes del programa si
        ;se desea generar un archivo .BIN)

                ASSUME      CS:CODIGO, DS:CODIGO, SS:CODIGO


        ; a partir de aqui se teclean todas las instrucciones del pro-
        ;grama; se debe poner una etiqueta a la primera instruccion.
```

```
INICIO.
        ;generalmente los datos residen un segmento y el codigo
        ;en otro. Para poder generar un archivo .BIN o .COM se
        ;declaran los datos en el mismo segmento. Se acostumbra
        ;declararlos al inicio del segmento. Por tanto, la primera
        ;instruccion del programa es un salto a la direccion donde
        ;inicia el codigo

                JMP INICOD

        ;Aqui se definen los datos
         NUMSET  DB      1,2,-3,-4,5,6,-7,8,9,10
         RESUL   DB      ?,0FFH

         INICOD:         ;aqui inicia el codigo del programa

                 MOV AL,0
                 MOV SI,0
                 MOV CX,NDATOS
         SLOOP:  CMP NUMSET[SI],0
                 JL NXTNUM
                 ADD AL,NUMSET[SI]
         NXTNUM: INC SI
                 LOOP SLOOP
                 MOV RESUL,AL

                 CALL PROCE1

         RETF    DB 0CBH     ;el ultimo codigo del programa debe
                             ;ser un RETurnFar para que regrese el control
                             ;a BASICM


      ;Aqui se declaran los procedures que requiera el programa

PROCE1   PROC
         ;instrucciones del procedimiento 1
         RET
PROCE1   ENDP

PROCE2    PROC
          ;instrucciones del procedimiento 2
PROCE2    ENDP


CODIGO            ENDS    ;aqui termina el segmento;es necesario
                  ;cerrarlo poniendo la misma etiqueta con que se
                  ;abrio, seguida del pseudo-op ENDS

      ;al final, debe ir el pseudo-op END, debe ir seguido por la
      ;etiqueta en donde inicia la primera instruccion del programa
      ;y debe llevar un salto de linea o <Enter>

                  END         INICIO
```

APENDICE 3: MANUAL DEL PROGRAMADOR DE EPROMS PARA PC

1. INTRODUCTION

This EPROM Writer Card is specially designed for IBM PC. It can be run under MS-DOS system. The design purpose is for multi-function, easy-use and reliability.

The main features are as follows:

- Can program various EPROMS: 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A, 27512, 27512A. (refer to INTEL parts-number)
- The programming voltage Vpp is very stable, and should never damage the EPROM. The Vpp is set automatically according to the EPROM type, including 25V, 21V, 12.5V.
- Can program, blank - check, and verify 10 EPROMs simultaneously.
- Using intelligent programming, the speed is 8 times faster than normal.
- Very easy to use, no need to toggle any switch.
- Can load programmed data directly from

disk.

- Can save EPROM data on disk.
- Can display, print and modify EPROM data.

2. MEMORY BUFFER

Once you run EPROM WRITER software, the software will build up a 64K byte Memory Buffer for save Master EPROM contents or for load the disk file in the Memory Buffer.

The address area of Memory Buffer is 0000-FFFF.

3. INSTALLATION AND START-UP PROCEDURE

a. The EPROM Writer Card contains: (1) Interface Card, (2) TEXTOOL Card, (3) flat cable, and (4) EPROM software disk.

b. Turn off the power of IBM PC.

c. Insert the Interface Card into any slot, and leave the TEXTOOL Card outside the case of PC.

d. Turn on the power.

e. Insert EPROM software disk into the approprite disk driver, and run the EPROM softwater by key in

> UPP512

then the screen will shown as below.

IBM-PC MSDOS EPROM WRITER EW-901
V-4.6
****
SELECT FROM THE FOLLOWING: -
<E> : EPROM TYPE --- (2764/21V)
<T> : TEXTOOL QUANTITY --- (1)
<Q> : QUIT.
<L> : LOAD DISK --- IN ANY ADDRESS.
<S> : SAVE DISK --- FROM ANY ADDRESS
<D> : DEBUG. (DISPLAY, MODIFY, PRINT)
<B> : BLANK CHECK.
<R> : READ --- IN C000H.
<V> : VERIFY --- WITH 0000H.
<C> : COPY --- FROM 0000H.

132

<1> : READA --- IN ANY ADDRESS.
<2> : VERIFYA --- WITH ANY ADDRESS.
<3> : COPYA --- FROM ANY ADDRESS.
<4> : COPYB --- BLANK CHECK & COPY.
<5> : VERIFYS --- VERIFY & DISPLAY ERROR.

RESULT:

---

4. OPERATING PROCEDURE

a. <EPROM TYPE>

Key in E, then the screen will shown as below.

SELECT EPROM TYPE & PROGRAMMING Vpp:
<1> : 2716 /25V
<2> : 2732 /25V
<3> : 2732A /21V
<4> : 2764 /21V
<5> : 2764A /12.5V
<6> : 27128 /21V
<7> : 27128A /12.5V
<8> : 27256 /21V

<9> : 27256A /12.5V
<A> : 27512 /21V
<B> : 27512A /12.5V

WHICH ONE ?-

The user should select the EPROM type, if you want to program 27128, then you may key in 6. (The default EPROM type is 2764).

The EPROM type is refer to INTEL parts-number.

After selection of EPROM type, the programming voltage is set automaticaily.

b. <TEXTOOL QUANTITY>

Key in T, then the screen will shown as below.

SELECT TEXTOOL QUANTITY:
<1> : 1
<2> : 2
<3> : 3
<4> : 4

WHICH ONE ?-

The user should select the quantity of TEXTOOLS. If you want to program 4 EPROMs simultaneously, then you may input 4. (The default TEXTOOL quantity is 4 at power-up)

c. <QUIT>

Key in Q, then return to MS-DOS system.

d. <LOAD DISK>

Key in L, then the screen will shown as below.

STARTING ADDRESS = HEX

Now input the starting address of Memory Buffer, ie. 0000, then will shown as below.

KEY IN FILE NAME?

Now input the file name to be loaded, i.e., key in

B: TEST. OBJ

Then the file TEST.OBJ on disk driver B will be loaded to the EPROM Memory Buffer.

e. <SAVE DISK>

Key in S, then the screen will shown as below.

TOTAL BYTE = HEX

Now key in data length to be stored to the disk, in hex code; i.e; input 0800 to store 2K bytes of data. Refer following table:

store 1K then key in 0400
store 2K then key in 0800
store 3K then key in 0C00
store 4K then key in 1000
store 5K then key in 1400
store 8K then key in 2000
store 16K then key in 4000
store 32K then key in 8000
store 64K then key in 0000

Next, the screen will show as below

STARTING ADDRESS = HEX

Now input the starting address of Memory Buffer, ie. 0000.

Then screen will shown as below.

KEY IN FILE NAME?

Now key in the file name, i.e.;

A: TEST.OBJ

f. <DEBUG>

Key in D, then the screen will cleared. Now PC will accept the follow command:

Display command
Print command
Substitute command
Quit command

(1) Display command -- to display Memory Buffer.

D<start address>.<end address>

example:

```
-D0000.001F
0000: 34 54 76 87 32 34 54 09 98
      21 54 77 34 12 A3 BF
0010: AA CD 34 4D F5 5A AA FD
      CB 78 34 5F 23 22 AC CD
```

(2) Print command -- to print out Memory Buffer.

P<start address>.<end address>

example:

```
-P0000.001F
0000: AB 45 C7 D9 93 90 00 10 02
      44 D4 EE EF 6F AF 7B
0010: 45 F5 E6 E2 DA CB B9 00 33
```



```
AD 22 A1 1F 89 9B 54
```

(3) Substitute command -- to Modify Memory Buffer contents.

S<start address>

example:

```
-S0000
0000: 45-32
0001: 83-56
0002: 72-90
0003: 55-49
0004: 57-
      --- TYPE RETURN KEY TO STOP
```

Key in <RETURN> to stop Subsultude command

(4) Quit command -- return to main menu.

--Q

g. <BLANK CHECK>

Key in B to blank check the EPROMs on the TEXTOOL Card, and display the following result.

RESULT: BLANK 1 OK 1FFF

2 OK 1FFF
3 ERROR AT 1098
4 OK 1FFF

h. <READ>

Key in C will program the EPROMs with Memory buffer contents, starting from location 0000.

The relationship between EPROM type and Memory Buffer area is as shown:

ERROM TYPE	MEMOPY BUFFER
2716	0000-07FF
2732(A)	0000-0FFF
2764(A)	0000-1FFF
27128(A)	0000-3FFF
27256(A)	0000-7FFF
27512(A)	0000-FFFF

i. <VERIFY>

Key in V to verify the EPROM contents with the Memory Buffer, starting from location 0000 and display following result:

RESULT: VERIFY 1 OK 1FFF
2 OK 1FFF
3 ERROR AT 1FFF
4 OK 1FFF

j. <COPY>

Key in C will program the EPROMs with Memory buffer contents, starting from location 0000.

k. <READA>

Key in 1 then the PC will request to input the address as shown below.

STARTING ADDRESS = HEX

Now key in the start address, ex., 2000. Then PC will read the EPROM contents on TEXTOOL # 1 into Memory Buffer, starting from location 2000.

The relationship between the EPROM type and Memory Buffer area is as shown.

TYPE	START ADDRESS	END ADDRESS
2716	XXXX	XXXX + 07FF
2732(A)	XXXX	XXXX + 0FFF
2764(A)	XXXX	XXXX + 1FFF
27128(A)	XXXX	XXXX + 3FFF
27256(A)	XXXX	XXXX + 7FFF
27512(A)	XXXX	XXXX + FFFF

l. <VERIFYA>

Key in 2, then PC will request to input the start address as shown below.

STARTING ADDRESS = HEX

Now input the start address, ex., 2000. Then PC will verify the EPROM contents with Memory Buffer, starting from location 2000.

m. <COPYA>

Key in 3, then first blank check the EPROMs on the TEXTOOL Card, if they are blanked then PC will request to input the starting address as shown below.

STARTING ADDRESS = HEX

Now input the starting address, ex., 2000. Then program the EPROMs on the TEXTOOL Card with Memory Buffer contents, starting from location 2000.

n. <COPYB>

COPYB is the same as COPY but first blank check the EPROMs on the TEXTOOL Card. If they are blanked then

program these EPROMs with Memory Buffer contents, starting from location 0000.

o. <VERIFYB>

VERIFYB is the same as VERIFY but VERIFYB will display both the different data, as shown below.

EXAMPLE:

ADDRESS	EPROM	MEMORY
0012:	4C	(56)
0134:	83	(47)
0567:	55	(32)

5. EXAMPLES

a. Combine Data

EX., combine two 2764 EPROM into one 27128 EPROM.

1. Key in E
2. Key in 5 -- select 2764
3. Insert # 1 2764 EPROM into TEXTOOL # 1.
4. Key in R -- the PC will read in # 1 2764 EPROM, the memory area is 0000 - 1FFF.

5. Insert # 2 2764 EPROM into TEX-TOOL 1.
6. Key in 1 -- the screen will display as below.

   STARTING ADDRESS = HEX

   Now you key in 2000. PC will read in # 2 EPROM, the memory area is 2000 - 3FFF.
7. Key in E.
8. Key in 7 -- select 27128 /21V
9. Insert one blank 27128 EPROM into TEXTOOL # 1.
10. Key in C, then the Memory Buffer contents from 0000 to 3FFF will writer into 27128 EPROM on TEX-TOOL # 1.

b. SEPARATE DATA

EX. , seperate one 27256 EPROM into two 27128 EPROM

1. Key in E
2. Key in 8 -- select 27256.
3. Insert master 27256 EPROM into TEXTOOL # 1.
4. Key in R, then the PC will read in

27256 EPROM, the memory area is 0000-7FFF.
5. Key in E.
6. Key in 6 -- select 27128.
7. Insert # 1 27128 blank EPROM into TEXTOOL 1.
8. Key in C, then the Memory Buffer contents from 0000-to 3FFF will write into 27128 EPROM on TEX-TOOL # 1.
9. Insert # 2 27128 blank EPROM into TEXTOOL # 1.
10. Key in 3 -- the screen will shown as below.

    STARTING ADDRESS = HEX

Now you key in 4000, then the Memory Buffer contents from 4000 to 7FFF will write into 27128 EPROM on TEXTOOL # 1.

APENDICE 4: FORMATO DE ALMACENAMIENTO EN EPROM PARA PROGRAMAS BASICM.

BASICM tiene un comando (STORE) para realizar la grabación de un programa Basic en EPROM; pero requiere el uso de un módulo de hardware adicional que se acopla al SSP y queda mapeado en cierta área del espacio de entrada/salida.

En lugar de ello, se ha preferido utilizar el programador de EPROMs ya disponible en la PC que se utiliza también para la grabación de programas desarrollados en ensamblador.

Por tanto, en lugar de usar el comando STORE, se sigue otro procedimiento (descrito en 6.6), para el cual es necesario conocer el formato en el cual este comando almacena los datos en la EPROM.

Para ello, es necesario hacer un análisis de la rutina que se ejecuta al invocar el comando STORE. Esta rutina se encuentra a partir de la dirección 3AE7H dentro del segmento de código de BASICM (el cual reside en la dirección absoluta F8000H). A continuación se muestra el listado de esta rutina, sobre el cual se hace referencia para realizar su análisis.

```
1         ;STORE:                subrutina que realiza la programa-
2              ;cion de EPROMs para programas BASICM en el SSP
3
4         ;SEGMENTO EN 0F8000H    (CS=F800H)
5         ;SEGMENTO DE DATOS EN 0 (DS=0000H)
6
7         ORG  3AE7H
8
9         CLD
10        CALL init
11
12        MOV AX,[0469H]
13        SUB AX,0EB6H
14
15        MOV [0448H],AX
16        MOV SI,448H
17        MOV DI,1
18        MOV CX,2
19        CALL progra
20
21        MOV SI,461H
22        MOV CX,2
23        CALL progra
24
25        MOV SI,45DH
26        MOV CX,2
27        CALL progra
28
29        MOV SI,46DH
30        MOV CX,2
31        CALL progra
32
33        MOV SI,0EB6H
34        ADD DI,8
35        MOV CX,[448H]
36
37        JCXZ  novar
38        CALL progra
39
40 novar: MOV SI,1800H
41        MOV CX,[45DH]
42        ADD CX,6
43        SUB CX,SI
44        CALL progra
45
46        CALL rut1
47        MOV DX,3D33H
48        CALL rut2
49        MOV CX,00FFH
50        RET
51
52
53        ORG 3B46H
54 progra:                ;rutina que interactua con el hardware
55                        ;para hacer la grabacion de los CX datos
56                        ;tomados de DS:SI y almcenados en la
57                        ;direccion DI de la EPROM
```

La subrutina PROGRA (reside en el offset 3B46H del mismo segmento) es la que interactúa con el hardware particular para realizar la grabación de los datos. Utiliza el registro DI para indicar la dirección de la EPROM donde se grabará el dato actual, el registro SI como apuntador de la memoria (dentro del segmento de datos DS=0) de donde tomará el dato actual, y CX como el contador de número de datos a grabar, incrementando DI y SI en cada ocasión.

De acuerdo a esto, los datos que se graban en la EPROM son los apuntados por SI antes de llamar a la subrutina PROGRA, y son los siguientes:

NOTA: Las referencias hacia el listado de la rutina se ponen entre paréntesis, indicando el número de línea.

* el primer byte de la EPROM se deja sin grabar, ya que DI se inicializa en 1 y no 0 (línea 17). Por tanto queda con un valor FFH.
* la dirección donde empieza la información de las variables del programa es 0:EB6H. En la dirección 0:469H se encuentra la dirección donde termina dicha información. Por tanto, el número de bytes requeridos para la información de las variables, N, es:

  N=(contenido de la dirección 0469H)-0EB6H

  Este dato se calcula (líneas 12 y 13) y se almacena en 0:448H (línea 15). Este valor, que ocupa 2 bytes, se graba en la EPROM (líneas 16 a 19).
* los siguientes dos bytes que se graban en la EPROM son dos bytes de control, tomados a partir de la dirección 0:461H (líneas 21 a 23).

* en la dirección 0:45DH se encuentra la dirección donde termina el código del programa, y esta información constituye los dos siguientes bytes grabados en EPROM (líneas 25 a 27).
* continúan dos bytes de control, tomados a partir de 0:46DH (líneas 29 a 31).
* DI se incrementa en 8 (línea 34), por lo que los siguientes 8 bytes de la EPROM quedan sin grabar (permanecen con FFH)
* El registro SI se carga ahora con 0EB6H (línea 33), es decir, se apunta al área que contiene la información de las variables. El contador CX se carga con el dato almacenado en 0:448H (línea 35), donde se había guardado N, el número de bytes necesarios para esa información de las variables. Los siguientes N bytes en la EPROM almacenarán la información de las variables.
* Se conoce la dirección donde inicia el código del programa (0:1800H), y la dirección donde termina (valor almacenado en 0:45DH, grabado ya previamente), así que el número de bytes que ocupa el código del programa, M, es: M=1800H-(contenido de 045DH)

  En la rutina se obtiene este valor, más 6, y se carga en CX, el contador (líneas 40 a 43); el 6 se debe a que el fin de programa se indica con 6 ceros al final (como se indicó en 5.3). Como el registro SI se carga con 1800H, entonces los siguientes bytes que se graban en EPROM son los referentes al código del programa residente en RAM.

Después de realizar este análisis de la rutina, se puede resumir el formato con que se deben grabar los datos en EPROM para un programa BASICM, como sigue:

* el primer byte queda sin grabar (con FFH)
* los siguientes dos bytes contienen el valor de N, el número de bytes requeridos para la información de las variables (tomados desde [448H]).
* los siguientes dos bytes son de control (tomados desde [461H])
* continúan dos bytes que contienen la dirección de RAM donde termina el código del programa (tomados desde [45DH])
* los siguientes dos bytes son de control (tomados desde [46DH])
* los siguientes 8 bytes quedan sin grabar (con FFH)
* continúan todos los bytes necesarios para la información de las variables (tomados a partir de [EB6H])
* continúan todos los bytes referentes al código del programa (tomados a partir de [1800H])
* los últimos 6 bytes son ceros

APENDICE 5: LISTADO DEL PROGRAMA DESP.ASM

```
;desp.asm                                                    Enero/1988

;despliegue en terminal del ASCII de los datos requeridos para la
;grabacion en EPROM de un programa BASICM para el Sistama SUPERPAT

;Programo':
;                    Alberto Aduna Grajeda


STATUS      EQU 0C01EH      ;puerto de status del transmisor serie
TXBUF       EQU 0C00EH      ;puerto del buffer de transmision
CR          EQU 0DH
LF          EQU 0AH


CSEG    SEGMENT

        ORG 6000H       ;esta rutina se grabara en la RCM en donde es-
                        ;ta el Basic residente (CS=F800H) en el offset
                        ;6000H, en dicha direccion no hay codigo graba-
                        ;do (desde 5A10H hasta 7FEFH no hay codigo).


;------------------------ programa principal -----------------------
MAIN    PROC FAR
        ASSUME CS:CSEG

        MOV AX,CS
        MOV DS,AX                   ;DS=CS temporalmente
        MOV SI,OFFSET LETRERO       ;despliega letrero inicial
        MOV CX,efes-LETRERO
cont:   MOV AL,[SI]
        CALL TX
        INC SI
        LOOP cont

        MOV AL,' '
        CALL TX                     ;trasmite espacio en blanco

;inicia la tranzmision de los datos para la EPROM
        XOR BH,BH       ;inicializa contador de localidades por ren-
                        ;glon desplegadas

        MOV CX,1
        CALL DESPF      ;despliega CX efes iniciales y pone DS en 0

        MOV AX,DS:[0463H]
        SUB AX,02B6H        ;obtiene num. de bytes para datos de variables
        MOV DS:[0448H],AX   ;almacena dicho valor
        MOV SI,448H         ;y lo despliega en terminal
        MOV CX,2
        CALL DESPL
```

```
        MOV SI,461H             ;despliega los bytes de control
        MOV CX,2
        CALL DESPL

        MOV SI.45DH             ;despliega dir.en RAM donde termina el codigo
        MOV CX,2                ;del programa
        CALL DESPL

        MOV SI,46DH             ;despliega siguientes bytes de control
        MOV CX,2
        CALL DESPL

        ;transmite 8 'FF' que son las sig. 8 localidades que se quedan
        ;sin programar
        MOV CX,8
        CALL DESPF

        MOV SI,0EB6H        ;apunta a datos para variables
        MOV CX,DS:[448H]    ;obtiene num.de bytes para datos de variables
        JCXZ  novar         ;si es cero ese num., no hay variables
        CALL DESPL          ;de lo contrario desplegarlas

novar:  MOV SI,1800H        ;desplegar el codigo del programa
        MOV CX,DS:[45DH]
        ADD CX,6            ;incluyendo los 6 ceros de "fin de programa"
        SUB CX,SI
        CALL DESPL

        MOV AL,LF                   ;para terminar, saltar renglon
        CALL TX
        MOV AL,CR
        CALL TX

        RET                         ;regresa control a BASICM
MAIN    ENDP
;-------------------- FIN programa principal ----------------------


DESPL   PROC                    ;transmite a terminal los código ASCII
        ;hexa de los 2 nibbles que forman el byte apuntado por SI
CXDES:  MOV AL,[SI]
        MOV AH,AL
        ;transmite primer nibble
        SHR AL,1
        SHR AL,1
        SHR AL,1
        SHR AL,1
        CALL ASCII          ;convierte AL a su valor ASCII
        CALL TX             ;transmite dato en AL a la terminal

        ;transmite segundo nibble
        MOV AL,AH
        CALL ASCII
        CALL TX

        ;checar si es necesario saltar renglon. Cada 16 datos desple-
        ;gados en la terminal se pasa a una nueva linea
        INC BH
        CMP BH,0FH
        JNZ noCR
        XOR BH,BH
        MOV AL,CR       ;carry return
        CALL TX
        MOV AL,LF       ;line-feed
        CALL TX
```

```
noCR:  MOV AL,' '
       CALL TX

       INC SI
       LOOP CXDES     ;continua desplegando hasta CX datos
       RET
DESPL  ENDP

;-------------------------------------------------------------------
ASCII  PROC                        ;regresa en AL el código ASCII del
                                   ;numero contenido en AL
       AND AL,0FH
       CMP AL,0AH
       JL NUM
       ADD AL,37H
       JMP FASC
NUM:   ADD AL,30H
FASC:  RET
ASCII  ENDP

;-------------------------------------------------------------------
TX     PROC             ;transmite dato contenido en AL a terminal
       MOV BL,AL        ;almacenamiento temporal del dato a desplegar
LEE:   MOV DX,STATUS
       IN AL,DX
       AND AL,20H       ;00100000B
       CMP AL,20H       ;si bit 5 del valor del puerto de status=0
       JNZ LEE          ;entonces no se puede transmitir
       MOV AL,BL        ;de lo contrario
       MOV DX,TXBUF
       OUT DX,AL        ;enviar dato a puerto de transmision
       RET
TX     ENDP

;-------------------------------------------------------------------
DESPF  PROC                    ; despliega efes iniciales e inicia DS=0
       MOV AX,CS
       MOV DS,AX
       MOV SI,OFFSET efes
       CALL DESPL
       XOR AX,AX
       MOV DS,AX
       RET
DESPF  ENDP


;----------------------- Area de datos -----------------------------

LETRERO DB '       * Codigo objeto para la EPROM : *',LF,CR
efes    DB  0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh.0ffh

CSEG   ENDS

       END MAIN
```

APENDICE 6: LISTADO DEL PROGRAMA ASCBIN.C

```
/*  ascbin.c -                                          version 1.1, 1988
        Programa que convierte los caracteres ASCII tomados
 del archivo de entrada y genera un archivo con el codigo binario
 correspondiente. Las cadenas entre asteriscos del archivo de entrada
 se consideran como comentario dentro del archivo y no se convierten.

                Programo':      Alberto Aduna Grajeda
*/

#include "stdio.h"
#define CR 0x0D
#define LF 0x0A

char name[20],name2[20],*ap1,*ap2;
FILE *f1,*f2;
char ch,n[3];
int cont,pto,i;

main(argc,argv)
int argc;
char *argv[];
{
        mensaje();
        if(argc==1){
                mensaje2();
                printf("Introducir el nombre del archivo a procesar: ");
                scanf("%s",name);
        }
        else
                strcpy(name,argv[1]);
        printf("\nEl archivo a procesar es %s\n",name);

        /* abrir archivo de lectura */
        if((f1=fopen(name,"rb")) == NULL){
                printf("el archivo %s no se puede accesar",name);
                exit();
        }

        ap1=name; ap2=name2;
        /*generar el nombre del archivo de salida */
        while(*ap1 != '.' && *ap1 != 0)
                *ap2++ = *ap1++;
        *ap2++='.';
        *ap2++='b';
        *ap2++='i';
        *ap2++='n';
        *ap2   = 0;

        /* abrir archivo de escritura */
        if((f2=fopen(name2,"wb")) == NULL){
                printf("no se puede crear el archivo de escritura %s",name2);
                exit();
        }
```

```
        pto=i=cont=0;   /* contador de puntos a impresion */
        ch=getc(f1);
        while(ch!=EOF){/* Inicia el proceso del arch. de entrada */
                while(ch==' ' || ch==LF || ch==CR || ch=='\t')
                        ch=getc(f1);
                if(ch == '*'){  /*saltar comentarios */
                        while((ch=getc(f1)) != '*')
                                ;
                        ch=getc(f1);
                }

                if(     ch >= '0' && ch <= '9') n[i++]=ch-'0';
                else if(ch >= 'A' && ch <= 'F') n[i++]=ch-'A'+ 0x0A;
                else if(ch >= 'a' && ch <= 'f') n[i++]=ch-'a'+ 0x0a;
                if(i==2){ /*se tienen los 2 bytes ASCII */
                        n[0]=n[0]<<4;
                        n[0]= n[0] | n[1];
                        cont++;
                        putc(n[0],f2);
                        i=0;

                }
                if(cont==16){
                        printf("."); /* imprime un '.' por cada 16 datos*/
                        pto++;
                        cont=0;
                }
                if(pto==64){
                        puts("\n"); /*salta renglon cada 64*16=1kb datos*/
                        pto=0;
                }
                if(ch != EOF) ch=getc(f1);
        }

        fclose(f1);
        fclose(f2);
        printf("\nProceso termina. Se genero el archivo %s\n",name2);
}

mensaje(){
        printf("\nConvertidor de codigo ASCii a BINario          ");
        printf("version 1.0, 1988\n");
        printf("Programo':  Alberto  Aduna  Grajeda\n\n");
}

mensaje2(){
        printf("\n Este programa requiere como entrada un archivo con\n");
        printf(  "los codigos ASCII de los caracteres hexa que  serán\n");
        printf(  "convertidos a binario.\n");
        printf(  "( toda  cadena  de  caracteres entre  asteriscos en\n");
        printf(  " el archivo de entrada se considera como comentario\n");
        printf(  " y no se convierten)\n\n");
}


                /* Desarrollado en el compilador Turbo-C v1.0; Enero 1988 */
```

APENDICE 7: PROGRAMA DE PRUEBA

```
  5 REM programa PRO.BAS
 10 MPUT 0C000H,2
 20 CONT1=0: CONT2=0
 30 AUX1=0: AUX2=0
 40 MPUT 0C006H,3
 50 MPUT 0C008H,0FFH
 60 MPUT 0C012H,CONT2
 70 MPUT 0C010H,CONT1
 80 DELAY 30
 90 INC AUX1:INC CONT1
100 IF AUX1<=9 GOTO 70
110 CONT1=CONT1+6: AUX1=0
120 IF CONT1 <= 99H GOTO 70
130 CONT1=0: INC CONT2: INC AUX2
140 IF AUX2<=9 GOTO 60
150 AUX2=0: CONT2=CONT2+6
160 IF CONT2 <= 99H GOTO 60
170 CONT2=0: GOTO 60
```

Descripcion del programa PRO.BAS

El programa está hecho para usarse junto con el modulo de despliegue de 7 segmentos, conectado a los puertos del 8256. Despliega continuamente una cuenta progresiva BCD en 4 dígitos.

```
10  Programa al 8256 en modo 8086 (bit 1)
    BITI=0 (bit 2), para poder usar bit 7 de puerto A (PA7) como
    salida.
20  Inicializa variables para dato actual a desplegar
30  Inicializa variables auxiliares para contar en BCD
40  Programa puerto B (digitos mas significativos en el despliegue)
    como puerto de salida
50  Programa puerto A (digitos menos significativos) como puerto de
    salida
60  Envia dato actual a puerto B
70  Envia dato actual de puerto A
80  Retardo de 30 mseg
90  Incrementa variables
100 Es un num. BCD valido? Si, continua desplegando digito inferior
110 No, ajusta
120 Se llego a la maxima cuenta en los digitos inferiores? No, continua
130 Si, Reinicializa cuenta de digitos inferiores, incrementa cuenta de
    digitos superiores
140 Es un num. BCD valido? Si, continua desplegado desde digito superior
150 No, Ajusta
160 Se llego a la maxima cuenta en los digitos superiores? No, continua
170 Si, reinicializa cuenta digitos superiores, continua desplegando
    desde digitos superiores.
```

Despliegue del archivo PRO.COD

```
        * Codigo objeto para la EPROM : *
FF 6C 00 FF 7F DB 1A FF 7F FF FF FF FF FF FF
FF FF 05 43 4F 4E 54 31 00 00 07 00 00 00 05
43 4F 4E 54 32 00 00 24 00 00 00 04 41 55 58
31 00 00 00 07 C0 0C 00 04 41 55 58 32 00 00
00 04 00 00 00 02 4C 54 00 00 00 00 00 00 00
00 00 03 55 58 30 00 00 00 00 00 00 00 00 03
43 4F 4E 00 00 00 00 00 00 00 00 03 55 58 31
00 00 00 00 00 00 00 00 05 45 44 49 32 30 00
00 00 00 00 00 0A 00 1F 00 0E 00 4D 50 55 54
20 30 43 30 30 30 48 2C 32 0D BA 1C 09 1A 00
C0 1B 02 00 00 00 14 00 2C 00 10 00 43 4F 4E
54 31 3D 30 3A 20 43 4F 4E 54 32 3D 30 B0 1D
0A 12 B6 0E 00 1B 00 00 B0 1D 0A 12 C2 0E 00
1B 00 00 00 00 1E 00 2B 00 0F 00 41 55 58 31
3D 30 3A 20 41 55 58 32 3D 30 0D B0 1D 0A 12
CE 0E 00 1E 00 00 B0 1D 0A 12 DA 0E 00 1B 00
00 00 00 28 00 1F 00 0E 00 4D 50 55 54 20 30
43 30 30 36 48 2C 33 0D BA 1C 09 1A 06 C0 1B
03 00 00 00 32 00 22 00 11 00 4D 50 55 54 20
30 43 30 30 38 48 2C 30 46 46 48 0D BA 1C 09
1A 08 C0 1B FF 00 00 00 3C 00 24 00 12 00 4D
50 55 54 20 30 43 30 31 32 48 2C 43 4F 4E 54
32 0D BA 1C 0A 1A 12 C0 12 C2 0E 00 00 00 46
00 24 00 12 00 4D 50 55 54 20 30 43 30 31 30
48 2C 43 4F 4E 54 31 0D BA 1C 0A 1A 10 C0 12
B6 0E 00 00 00 50 00 17 00 09 00 44 45 4C 41
59 20 33 30 0D 2E 21 06 1B 1E 00 00 00 5A 00
27 00 13 00 49 4E 43 20 41 55 58 31 3A 49 4E
43 20 43 4F 4E 54 31 0D F4 20 06 12 CE 0E F4
20 06 12 B6 0E 00 00 64 00 30 00 13 10 49 46
20 41 55 58 31 3C 3D 39 20 47 4F 54 4F 20 37
30 0D FB 21 15 12 CE 0E 1B 09 15 29 27 00 10
22 09 07 20 06 1A 46 00 00 00 6E 00 38 00 16
00 43 4F 4E 54 31 3D 43 4F 4E 54 31 2B 36 3A
20 41 55 58 31 3D 30 0D B0 1D 10 12 B6 0E 00
12 B6 0E 1B 06 15 C5 25 00 B0 1D 0A 12 CE 0E
00 1B 00 00 00 00 78 00 35 00 18 10 49 46 20
43 4F 4E 54 31 20 3C 3D 20 39 39 48 20 47 4F
54 4F 20 37 30 0D FB 21 15 12 B6 0E 1B 99 15
29 27 00 10 22 09 07 20 06 1A 46 00 00 00 82
00 3B 00 1D 00 43 4F 4E 54 31 3D 30 3A 20 49
4E 43 20 43 4F 4E 54 32 3A 20 49 4E 43 20 41
55 58 32 0D B0 1D 0A 12 B6 0E 00 1B 00 00 F4
20 06 12 C2 0E F4 20 06 12 DA 0E 00 00 8C 00
30 00 13 10 49 46 20 41 55 58 32 3C 3D 39 20
47 4F 54 4F 20 36 30 0D FB 21 15 12 DA 0E 1B
09 15 29 27 00 10 22 09 07 20 06 1A 3C 00 00
00 96 00 38 00 16 00 41 55 58 32 3D 30 3A 20
43 4F 4E 54 32 3D 43 4F 4E 54 32 2B 36 0D B0
1D 0A 12 DA 0E 00 1B 00 00 B0 1D 10 12 C2 0E
00 12 C2 0E 1B 06 15 C5 25 00 00 00 A0 00 35
00 18 10 49 46 20 43 4F 4E 54 32 20 3C 3D 20
39 39 48 20 47 4F 54 4F 20 36 30 0D FB 21 15
12 C2 0E 1B 99 15 29 27 00 10 22 09 07 20 06
1A 3C 00 00 00 AA C0 29 00 11 10 43 4F 4E 54
32 3D 30 3A 20 47 4F 54 4F 20 36 30 0D B0 1D
0A 12 C2 0E 00 1B 00 00 07 20 06 1A 3C 00 00
00 00 00 00 00 00 00
>
```

```
debug pro.bin
-r
AX=0000  BX=0000  CX=035E  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0D69  ES=0D69  SS=0D69  CS=0D69  IP=0100   NV UP EI PL NZ NA PO NC
0D69:0100 FF6C00        JMP     FAR [SI+00]                  DS:0000=20CD
-d 100 45f
0D69:0100  FF 6C 00 FF 7F DB 1A FF-7F FF FF FF FF FF FF FF  .l..............
0D69:0110  FF 05 43 4F 4E 54 31 00-00 07 00 00 00 05 43 4F  ..CONT1.......CO
0D69:0120  4E 54 32 00 00 24 00 00-00 04 41 55 58 31 00 00  NT2..$....AUX1..
0D69:0130  00 07 00 00 00 04 41 55-58 32 00 00 00 04 00 00  ......AUX2......
0D69:0140  00 02 4C 54 00 00 00 00-00 00 00 00 00 03 55 58  ..LT..........UX
0D69:0150  30 00 00 00 00 00 00 00-00 03 43 4F 4E 00 00 00  0.........CON...
0D69:0160  00 00 00 00 00 03 55 58-31 00 00 00 00 00 00 00  ......UX1.......
0D69:0170  00 05 45 44 49 32 30 00-00 00 00 00 00 0A 00 1F  ..EDI20.........
0D69:0180  00 0E 00 4D 50 55 54 20-30 43 30 30 30 48 2C 32  ...MPUT 0C000H,2
0D69:0190  0D BA 1C 09 1A 00 C0 1B-02 00 00 00 14 00 2C 00  ..............,.
0D69:01A0  10 00 43 4F 4E 54 31 3D-30 3A 20 43 4F 4E 54 32  ..CONT1=0: CONT2
0D69:01B0  3D 30 B0 1D 0A 12 B6 0E-00 1B 00 00 B0 1D 0A 12  =0..............
0D69:01C0  C2 0E 00 1B 00 00 00 00-1E 00 2B 00 0F 00 41 55  ..........+...AU
0D69:01D0  58 31 3D 30 3A 20 41 55-58 32 3D 30 0D B0 1D 0A  X1=0: AUX2=0....
0D69:01E0  12 CE 0E 00 1B 00 00 B0-1D 0A 12 DA 0E 00 1B 00  ................
0D69:01F0  00 00 00 28 00 1F 00 0E-00 4D 50 55 54 20 30 43  ...(.....MPUT 0C
0D69:0200  30 30 36 48 2C 33 0E BA-1C 09 1A 06 C0 1B 03 00  006H,3..........
0D69:0210  00 00 32 00 22 00 11 00-4D 50 55 54 20 30 43 30  ..2."...MPUT 0C0
0D69:0220  30 38 48 2C 30 46 46 48-0D BA 1C 09 1A 08 C0 1E  08H,0FFH........
0D69:0230  FF 00 00 00 3C 00 24 00-12 00 4D 50 55 54 20 30  ....<.$...MPUT 0
0D69:0240  43 30 31 32 48 2C 43 4F-4E 54 32 0D BA 1C 0A 1A  C012H,CONT2.....
0D69:0250  12 C0 12 C2 0E 00 00 00-46 00 24 00 12 00 4D 50  ........F.$...MP
0D69:0260  55 54 20 30 43 30 31 30-48 2C 43 4F 4E 54 31 0D  UT 0C010H,CONT1.
0D69:0270  BA 1C 0A 1A 10 C0 12 B6-0E 00 00 00 50 00 17 00  ............P...
0D69:0280  09 00 44 45 4C 41 59 20-33 30 0D 2E 21 06 1B 1E  ..DELAY 30..!...
0D69:0290  00 00 00 5A 00 27 00 13-00 49 4E 43 20 41 55 58  ...Z.'...INC AUX
0D69:02A0  31 3A 49 4E 43 20 43 4F-4E 54 31 0D F4 20 06 12  1:INC CONT1.. ..
0D69:02B0  CE 0E F4 20 06 12 B6 0E-00 00 64 00 30 00 13 10  ... ......d.0...
0D69:02C0  49 46 20 41 55 58 31 3C-3D 39 20 47 4F 54 4F 20  IF AUX1<=9 GOTO
0D69:02D0  37 30 0D FB 21 15 12 CE-0E 1B 09 15 29 27 00 10  70..!.......)'..
0D69:02E0  22 09 07 20 06 1A 46 00-00 00 6E 00 38 00 16 00  ".. ..F...n.8...
0D69:02F0  43 4F 4E 54 31 3D 43 4F-4E 54 31 2B 36 3A 20 41  CONT1=CONT1+6: A
0D69:0300  55 58 31 3D 30 0D B0 1D-10 12 B6 0E 00 12 B6 0E  UX1=0...........
0D69:0310  1B 06 15 C5 25 00 B0 1D-0A 12 CE 0E 00 1B 00 00  ....%...........
0D69:0320  00 00 78 00 35 00 18 10-49 46 20 43 4F 4E 54 31  ..x.5...IF CONT1
0D69:0330  20 3C 3D 20 39 39 48 20-47 4F 54 4F 20 37 30 0D  <= 99H GOTO 70.
0D69:0340  FB 21 15 12 B6 0E 1B 99-15 29 27 00 10 22 09 07  .!.......)'..".. 
0D69:0350  20 06 1A 46 00 00 00 82-06 3B 00 1D 00 43 4F 4E  ..F.....;...CON
0D69:0360  54 31 3D 30 3A 20 49 4E-43 20 43 4F 4E 54 32 3A  T1=0: INC CONT2:
0D69:0370  20 49 4E 43 20 41 55 58-32 0D B0 1D 0A 12 B6 0E  INC AUX2........
0D69:0380  00 1B 00 00 F4 20 06 12-C2 0E F4 20 06 12 DA 0E  ..... ..... ....
0D69:0390  00 00 8C 00 30 00 13 10-49 46 20 41 55 58 32 3C  ....0...IF AUX2<
0D69:03A0  3D 39 20 47 4F 54 4F 20-36 30 0D FB 21 15 12 DA  =9 GOTO 60..!...
0D69:03B0  0E 1B 09 15 29 27 00 10-22 09 07 20 06 1A 3C 00  ....)'..".. ..<.
0D69:03C0  00 00 96 00 58 00 16 00-41 55 58 32 3D 30 3A 20  ....8...AUX2=0: 
0D69:03D0  43 4F 4E 54 32 3D 43 4F-4E 54 32 2B 36 0D B0 1D  CONT2=CONT2+6...
0D69:03E0  0A 12 DA 0E 00 1B 00 00-B0 1D 10 12 C2 0E 00 12  ................
0D69:03F0  C2 0E 1B C6 15 C5 25 00-00 00 A0 00 35 00 1B 10  ......%.....5...
0D69:0400  49 46 20 43 4F 4E 54 32-20 3C 3D 20 39 39 43 20  IF CONT2 <= 99H 
0D69:0410  47 4F 54 4F 20 36 30 0D-FB 21 15 12 C2 0E 1B 99  GOTO 60..!......
0D69:0420  15 29 27 00 10 22 09 07-20 06 1A 3C 00 00 00 AA  .)'..".. ..<....
0D69:0430  00 29 00 11 10 43 4F 4E-54 32 3D 30 3A 20 47 4F  .)...CONT2=0: GO
0D69:0440  54 4F 20 36 30 0D B0 1D-0A 12 C2 0E 00 1B 00 00  TO 60..........
0D69:0450  07 20 06 1A 3C 00 00 00-00 00 00 00 00 00 E8 10  . ..<...........
-
```