UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UN SISTEMA DE DESARROLLO PARA EL MICROCONTROLADOR 80535

T E S I S

Que para obtener el título de:

INGENIERO MECANICO ELECTRICISTA

P r e s e n t a :

MIGUEL ANGEL BAÑUELOS SAUCEDO

Director de Tesis: Ing. Francisco Rodríguez Ramírez

MEXICO, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

El desarrollo que vive la electrónica ha permitido que ésta extienda sus áreas de aplicación cada vez más, hasta alcanzar aspectos tan cotidianos como el transporte, el esparcimiento, la comunicación personal, e incluso la preparación de alimentos. Esta nueva revolución industrial es la que ha hecho posible que aparatos como los teléfonos celulares, los hornos de microondas o los reproductores de compact disk sean hoy tan comunes. Detrás de éstas y otras aplicaciones de la electrónica se encuentra un gigantesco trabajo de ingeniería, el cual es impulsado por un espíritu de creación, y está encauzado por un interés en satisfacer las necesidades de nuestra sociedad.

Dentro de muchos de los artículos electrónicos que encontramos en nuestra vida diaria, y en nuestro trabajo, podemos hallar un componente muy versátil, ejemplo sobresaliente de la tecnología: el microcontrolador.

Los microcontroladores son el resultado de la evolución de los microprocesadores, y son sistemas digitales que incluyen, en un sólo chip, una unidad central de proceso (CPU), memoria volátil (RAM), memoria permanente (ROM), funciones de entrada/salida, y en algunos casos: temporizadores, controladores

de interrupciones, moduladores de ancho de pulso, convertidores de señales analógicas a digitales (ADC), etc.

La principal ventaja de los microcontroladores estriba en que, al incluir diversas funciones en un sólo chip, proporcionan soluciones con menos componentes, lo cual permite reducir el tamaño de los aparatos y, aunado al empleo de tecnología CMOS en la fabricación de algunos microcontroladores, la realización de sistemas portátiles alimentados con baterías. Además, presentan todas las ventajas derivadas de un sistema digital como lo es la facilidad de modificación, ya que bastará con alterar el programa que lo controla, para poder realizar funciones distintas.

En la actualidad existe un gran número de fabricantes de dispositivos semiconductores que manejan líneas de microcontroladores entre los que podemos citar: Intel, Motorola, NEC, Zilog, Hitachi, Philips-Signetics, National y AMD entre otros. Se fabrican microcontroladores de 4, 8, 16 y 32 bits y su rango de aplicaciones es prácticamente ilimitado. Por ejemplo podemos citar: contestadoras telefónicas, teléfonos celulares, teléfonos públicos, controles de inyección de combustible en automóviles y controles antibloqueo de frenos, teclados de computadora, controladores de disco duro, impresoras, graficadores, fotocopiadoras, máquinas de escribir, sintonizadores de video por cable, hornos de microondas, sistemas de alarma, controladores industriales, etc.

Una característica notable del desarrollo de sistemas electrónicos, y en general de cualquier actividad, es el empleo de herramientas y dispositivos para facilitar el trabajo, y es así como se origina el proyecto que se presenta en esta tesis.

CAPITULO I.

DEFINICION DEL PROYECTO.

I.1 HERRAMIENTAS EN EL DISEÑO CON MICROCONTROLADORES.

El diseño de sistemas basados en microprocesadores o microcontroladores lo podemos dividir en dos: circuitería y programación. Para el desarrollo y prueba de la circuitería, es frecuente contar en un laboratorio con osciloscopio y punta lógica, los cuales nos permiten la puesta en marcha del circuito en gran parte de los casos; aunque a veces se requieren herramientas más sofisticadas, como por ejemplo un analizador de estados lógicos.

La depuración de la programación, está generalmente menos apoyada y en el caso más simple consta únicamente de un ensamblador y un grabador de memorias UVEPROM. En el caso de que el microprocesador del sistema a desarrollar sea diferente del procesador del sistema en el cual se ejecuta el programa ensamblador, por lo general una PC, se deberá emplear un ensamblador cruzado (crossassembler). De cualquier manera, la corrección de errores con estas herramientas es un proceso lento

y tedioso. Primero hay que generar un programa e incluir en él la escritura de datos clave a algún puerto del sistema, de tal manera que al verificar su estado, quizá mediante algunos LED, podamos tener una idea del grado de ejecución. Después tenemos que grabar la memoria y colocarla en la tarjeta. Si se localiza un error, deberá repetirse el proceso y además esperar a que un borrador de memorias deje a la UVEPROM lista para reescribirse. En el caso de contarse con memorias EEPROM y un grabador apropiado, el tiempo de borrado será más corto; sin embargo, el procedimiento no deja de ser lento y poco eficiente.

Una primera alternativa la constituyen los emuladores de memorias ROM. Estos consisten en un sistema de memoria RAM, que se comunican con una computadora central (PC) mediante un puerto serial, y se conectan, mediante un cable, al receptáculo de la memoria ROM en el prototipo. El programa, una vez ensamblado, se 'baja' por el puerto serial a una memoria RAM que hará las veces de la memoria ROM. Aquí se tiene la ventaja de que no es necesario quitar y poner la memoria cada vez que se modifica el programa, resultando en una opción más rápida que la anterior.

Otra opción son los emuladores en-circuito. Estos están formados por una circuitería que se comunica con una computadora central, ya sea por un puerto serial o mediante una conexión directa al bus, y tienen un cable conector especial que se enchufa en lugar del procesador del prototipo. Las funciones del

procesador serán emuladas, y controladas desde la computadora central. Se puede ejecutar el código que se encuentre en la memoria del prototipo, o bien, algún programa que se haya cargado en el emulador. También se puede leer y modificar el contenido de los registros internos del procesador, así como el de la memoria. En algunos casos, es posible llevar una historia de el resultado de la ejecución del programa para poder analizarlo mediante instrucciones especiales. En general, los emuladores en-circuito, representan una solución muy poderosa en la depuración de los programas; sin embargo, tienen el inconveniente de ser relativamente costosos, y de requerirse un emulador distinto para cada procesador.

Una posibilidad de depuración de la programación más económica la constituyen los sistemas de desarrollo. Estos son tarjetas que se controlan desde una computadora central vía un puerto serial, y están formados por un sistema básico de memoria y dispositivos periféricos, a partir de los cuales se puede generar y probar una arquitectura más específica, ya que las líneas de datos, direcciones y control se encuentran disponibles al usuario mediante algún tipo de conector. Al igual que con el emulador en-circuito, se pueden monitorear todas las operaciones del procesador desde la computadora central. En ocasiones se les conoce como sistemas de evaluación, y bien podría considerárseles la mejor alternativa, en razón costo-beneficio, para la

depuración de programas de microcontroladores y microprocesadores.

Es así como nace la idea de diseñar un sistema de desarrollo para un microcontrolador. En este caso, además de su uso como herramienta en el desarrollo de proyectos, representa también un apoyo didáctico en la comprensión del funcionamiento de sistemas basados en microcontrolador.

I.2 EL MICROCONTROLADOR 8031.

Actualmente existen en el mercado toda una variedad de microcontroladores. Resulta fácil percatarse de que los fabricantes de dispositivos semiconductores más importantes incluyen por lo general una línea de microcontroladores en su línea de productos. Como ya mencionamos, podemos encontrar microcontroladores de 4, 8, 16, y 32 bits; sin embargo, con una longitud de palabra de 8 bits podemos resolver muchos problemas de aplicación, además de que su dominio nos permite comprender mejor a aquellos microcontroladores que manejan un mayor número de bits por palabra.

De entre esa variedad de microcontroladores, escoger el más adecuado para una aplicación específica, puede volverse una tarea algo compleja, debido principalmente a que un mismo fabricante

puede producir diversas variantes de un mismo microcontrolador. Este hecho multiplica las alternativas, de tal manera que es muy factible encontrar varias opciones para el mismo problema. En este caso deberán entrar en juego factores como la disponibilidad, el costo, el respaldo por parte del fabricante, y si ya se cuenta con alguna herramienta de apoyo (p.ej: manuales, ensamblador, emulador, etc.). No obstante, los microcontroladores son dispositivos tan versátiles, que a menudo cualquiera puede resolver una tarea, aunque se tengan que implementar de manera distinta.

Uno de los microcontroladores más baratos, que se pueden conseguir en México, es el 8031 que diseñó originalmente Intel, a principios de la década de los ochenta. Éste, se ha convertido en un estándar de la industria electrónica, y en la actualidad es fabricado también por otras empresas como Advanced Micro Devices, Siemens y Signetics, además de que se han generado versiones ampliadas.

El microcontrolador 8031 forma parte de una familia denominada por Intel como MCS-51, y posee las siguientes características:

-Unidad central de proceso (CPU) de 8 bits.

-Procesador booleano.

-4 puertos de E/S de 8 bits.

-128 a 256 bytes de memoria RAM en el chip.

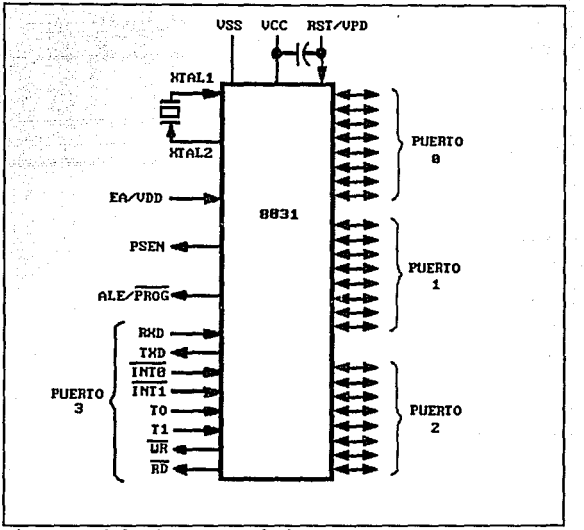


Diagrama del microcontrolador 8031.

-4 u 8 Kbytes de memoria ROM de código interna.

-2 ó 3 contadores temporizadores de 16 bits.

-Puerto serial full-duplex.

-5 ó 6 fuentes de interrupción con 2 niveles de prioridad.

-Oscilador integrado.

-Direccionamiento para 64 Kbytes de código.

-Direccionamiento para 64 Kbytes de datos independientes.

I.3 EL MICROCONTROLADOR 80535.

Actualmente muchos microcontroladores cuentan con algunas características adicionales tales como convertidores analógico/digitales, supervisores (watchdog), y otros. Tal es el caso del 80535, que es un microcontrolador derivado del 8031 y que es producido por Advanced Micro Devices (AMD) y Siemens.

Al tener la posibilidad de trabajar con este microcontrolador, se consideró como una buena alternativa el generar un sistema de desarrollo basándose en él, ya que se estarían satisfaciendo dos aspectos: primero, al ser totalmente compatible con el original 8031 se aprovecharían sus ventajas de costo y facilidad de adquisición; segundo, al ser una versión que incluye muchas de las cualidades de los microcontroladores más modernos, se cubriría también una mayor gama de aplicaciones y se tendría un sistema competitivo con los desarrollos más recientes en el área.

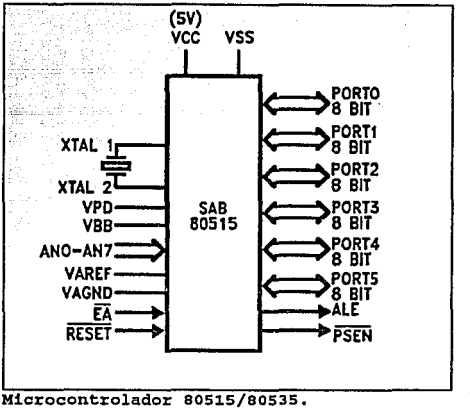
El microcontrolador 80535 cuenta con las siguientes características:

-256 bytes de RAM integrada.

-8 Kbytes de ROM (80515).

-6 puertos de E/S de 8 bits.

-3 contadores/temporizadores de eventos de 16 bits.



Microcontrolador 80515/80535.

-Capacidad de recarga, captura y comparación en el Temporizador (timer) número 2.

-Puerto serial full-duplex.

-12 fuentes de interrupción con 4 niveles de prioridad.

-Convertidor A/D de 8 bits y 8 canales seleccionables.

-Supervisor/temporizador de 8 bits.

-Función de espera (standby).

-Direccionamiento para 64 Kbytes de código.

-Direccionamiento para 64 Kbytes de datos.

-El código y funciones del 8031 son compatibles.

I.4 EL SISTEMA DE DESARROLLO SD535.

Tomando en cuenta las características del microcontrolador 80535, se procedió a definir cuales serían los alcances del proyecto; es decir, cuales eran las funciones que pretenderíamos implementar en nuestro diseño.

Como características del sistema de desarrollo se propusieron las siguientes:

- -Control de las funciones desde una PC mediante comunicación serial empleando menús y ventanas.
- -Manejo independiente de los mapas de memoria de código y datos.
- -Suficiente memoria de código para ejecutar programas largos como los generados mediante un compilador.
- -Suficiente memoria de datos para incluso poder funcionar como un pequeño sistema de adquisición de datos.
- -Ampliación del número de líneas de entrada/salida (E/S) mediante una interfase paralela programable (PPI).
- -Líneas de datos, direcciones y control disponibles para expansión o modificación del sistema.
- -Capacidad de generación de señales mediante un convertidor digital-analógico.
- -Capacidad de grabación y ejecución de un programa almacenado en EEPROM.

-Manejo de archivos de datos y programa almacenados en disco.

-Despliegue del contenido de registros y memoria en pantalla.

-Posibilidad de inserción de pausas en el código del programa para facilitar su depuración.

-Ejecución de programas paso a paso.

-Un ambiente integrado que permita la edición del programa, su ensamblado y ejecución en el mismo entorno operativo.

Una vez planteados los objetivos que debía cumplir el sistema de desarrollo, se procedió a detallar su diseño. Primero se consideró la parte electrónica y posteriormente se desarrolló la parte de programación.

CAPITULO II.

DISEÑO DE LA CIRCUITERIA.

II.1 DISEÑO DEL MAPA DE MEMORIA.

Una vez que se tuvieron definidos los alcances de la circuitería se procedió a establecer el mapa de memoria del sistema, ya que éste es el punto de partida para diseñar la lógica de decodificación y para poder especificar cuales son los componentes que se requieren. Para poder establecerlo debemos considerar algunos aspectos del funcionamiento del microcontrolador. Para empezar, es importante considerar la rutina de Reset del micro, ya que ésta define la ubicación de la primera instrucción, a partir de la cual comenzará la operación del sistema.

Después de dar un Reset al microcontrolador, el apuntador de programa (Program Counter) se carga con 0000H, y se ejecuta la instrucción localizada en esa dirección. Ésto nos obliga a que en las primeras direcciones de memoria tengamos una o varias instrucciones en memoria ROM, ya que de otra forma no podríamos asegurar cual es el flujo de instrucciones una vez que se ha

reinicializado al microcontrolador. Es por ello que el programa monitor del sistema de desarrollo podría ubicarse en las primeras localidades del mapa de memoria. Sin embargo, para el manejo de las interrupciones, el microcontrolador cuenta con una tabla de vectores de interrupción, que se encuentra ubicada en las primeras localidades del mapa de memoria, a partir de la dirección 0003H. Si implementamos con memoria ROM esas localidades, entonces no tendremos oportunidad de alterar las instrucciones contenidas en esa tabla. Por lo tanto, una posible solución es implementar las tres primeras localidades del mapa de memoria (0000H-0002H) con ROM y a partir de la dirección 0003H implementarlas con RAM, para permitir la alteración de la información. Esas primeras tres localidades deberán contener entonces un salto a la dirección donde se ubique el programa monitor. La implementación de una lógica de decodificación de esta naturaleza no resulta sencilla. Si para ella se emplean compuertas lógicas, se necesitarán varios chips para implementar la función requerida, debido a que todas

FUENTE	DIR DEL VECTOR
RESET	0000H
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H
TF2+EXF2	002BH
IADC	0043H
IEX2	004BH
IEX3	0053H
IEX4	005BH
IEX5	0063H
IEX6	006BH

Mapa de vectores de interrupción.

las líneas de direccionamiento se vuelven significativas. Aún y cuando se empleasen decodificadores, necesitaremos unos cinco chips TTL. Otra alternativa es el empleo de PALs, aunque dado el número de entradas que requiere la función, se necesitaría por lo menos todo un PAL para implementarla.

El microcontrolador maneja mapas de memoria independientes entre si para código y datos. Como resulta lógico imaginarse, para el mapa de memoria de código sólo se permiten los accesos de lectura. Para nuestro sistema de desarrollo resulta esencial que el contenido de la memoria de código pueda ser alterada (reescrita), y por eso se empleará una memoria RAM. La única línea del microcontrolador que permite una escritura en memoria externa al microcontrolador es -WR, la cual es activada con las instrucciones que hacen escritura en memoria de datos. Por lo tanto, la única posibilidad de poder escribir en el área de código, es haciéndole creer al microcontrolador que está actuando en el área de datos. Para lograr esto, se implementó una lógica que traslapa el área de código encima del área de datos. Dado que ambos mapas de memoria manejan las mismas líneas de direccionamiento, la única manera de producir el traslape es generando una línea de control adicional. Para la implementación de esta línea se puede utilizar un biestable (flip-flop), sin embargo se prefirió emplear un Latch por aportar más salidas en un sólo chip que después se podrían aprovechar. Un bit de este Latch servirá para determinar si se traslapan los mapas o no, es

decir, si las instrucciones de escritura se llevarán a cabo sobre la memoria de datos o sobre la memoria de código. Como en un estado inicial los mapas no deben encontrarse traslapados, se escogió un Latch que tuviera un terminal de borrado (clear), de tal manera que si conectamos esa terminal con la línea de Reset del sistema, podamos asegurar que al reinicializar al sistema los mapas no se encuentren traslapados.

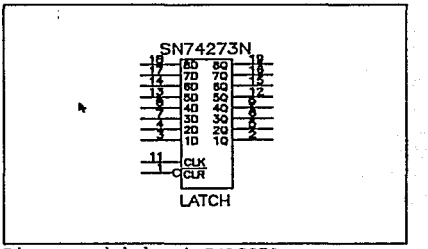
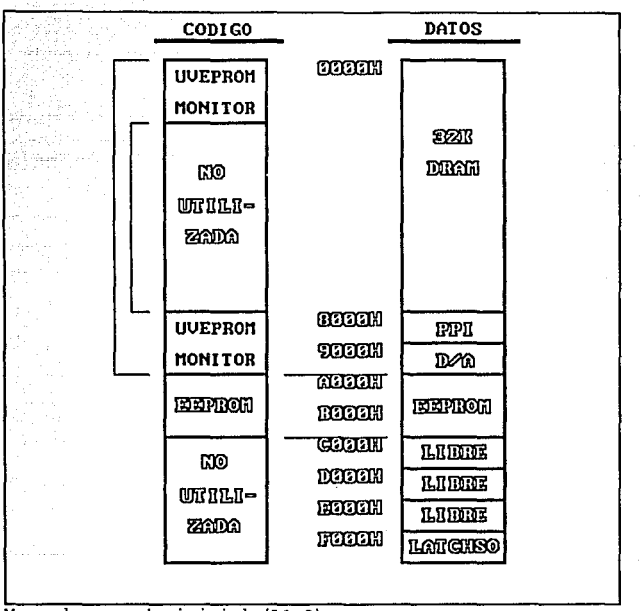


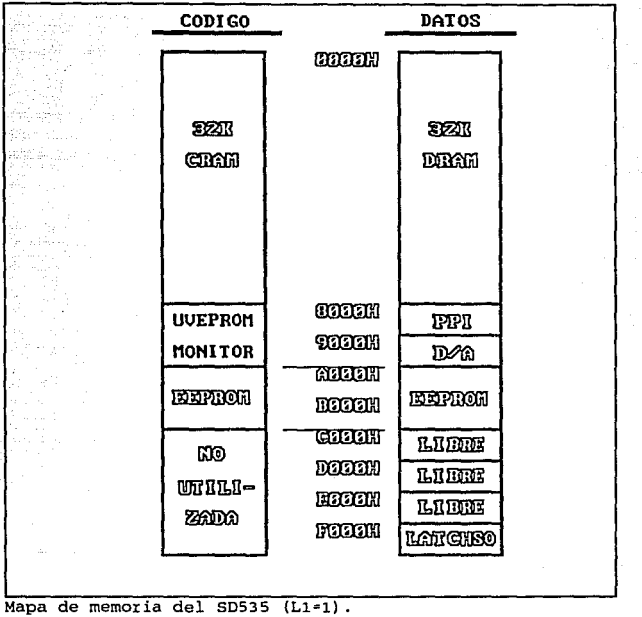
Diagrama del latch 74LS273.

El mismo Latch empleado en la superposición de los mapas de código y datos se aprovechó para implementar una lógica de decodificación muy simple y versátil para la ubicación del programa monitor en ROM. Como este Latch es utilizado exclusivamente por el programa monitor, en adelante nos referiremos a él como Latch SO. Se diseñó una arquitectura que permite traslapar una memoria ROM con una RAM en las primeras



Mapa de memoria inicial (L1=0).

localidades de el mapa de código. El monitor se encuentra en una memoria UVEPROM 2764 de 8 Kbytes, la cual después de un Reset se direcciona en las localidades 0000H-1FFFH y simultáneamente en las localidades 8000H-9FFFH; es decir, que si el microcontrolador direcciona las localidades 0000H y 8000H, físicamente se estará direccionando la misma localidad dentro de la UVEPROM.



Mapa de memoria del SD535 (L1=1).

Una vez que se inicia la ejecución del programa monitor, se transfiere el control al segundo grupo de localidades (8000H-9FFFH) y se modifica el bit de control del Latch SO (se hace L1=1), esto provoca que ahora en las localidades 0000H-1FFFH del mapa de memoria de código se active un chip con memoria RAM (denominado CRAM), en lugar de duplicarse la activación de la UVEPROM. Como se puede ver en las ilustraciones, el chip CRAM se

activa para los primeros 32 Kbytes del mapa de memoria de código (0000H-7FFFH).

Otras tres líneas del Latch se usaron para la conexión de tres LEDs. Uno verde, que indica que el sistema se encuentra listo para recibir un comando, uno rojo para señalar que el sistema se encuentra ocupado ejecutando un comando, y otro amarillo que se enciende cuando se ha transferido el control de la tarjeta a un programa del usuario.

II.2 IMPLEMENTACION DE LA LOGICA DE DECODIFICACION.

Para la implementación de la lógica de decodificación se utilizó un PAL (Arreglo Lógico Programable) número 16L8. Este PAL consiste en un arreglo de compuertas AND, cuyas conexiones son programables, conectadas en cascada con un arreglo de compuertas OR e Inversores de salida Tres Estados no-programables. Mediante el empleo de esta tecnología se reduce el número de componentes y se logra una arquitectura más flexible.

Como se puede apreciar en el diagrama, las líneas de direcciones A12-A15, así como las líneas de lectura (RD), escritura (WR) y cambio de mapa de la ROM del sistema monitor, se usan como entradas del PAL. Adicionalmente se emplea la entrada I9 del PAL como una habilitación del chip y se encuentra conectada a tierra (se verifica baja).

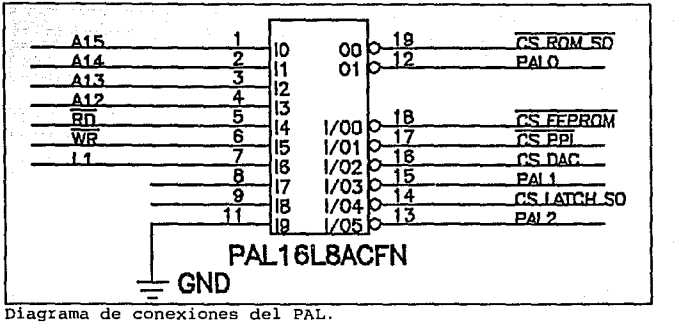


Diagrama de conexiones del PAL.

La programación interna del PAL genera, a partir de las entradas, las líneas de habilitación para la ROM del sistema monitor, la EEPROM, el PPI (puerto paralelo), el DAC (convertidor analógico-digital), y el Latch SO. Para hacer más flexible el sistema de desarrollo, se dejaron libres para el usuario tres salidas del PAL (pines 12, 13 y 15), que están disponibles mediante un conector, para posibles expansiones del sistema. Cabe aclarar que la decodificación de estas salidas aún no está determinada, dejándose también la libertad de programarlas al usuario.

Para la implementación de la memoria de datos y de la

memoria de código se decidió utilizar memorias RAM de 32 Kbytes, ya que al ser de gran capacidad proporcionan gran flexibilidad al sistema. Ocupan el mismo espacio en la tarjeta que una memoria 8 Kbytes, por lo que se reduce el número de componentes, y además resulta inferior el precio por Kbyte de memoria, abaratando el costo del sistema.

Las líneas de selección de los chips de memoria RAM de código (CRAM) y memoria RAM de datos (DRAM) se implementó con el auxilio de un chip de compuertas NAND y un chip de inversores. El uso de estos componentes se hubiera podido evitar si hubiésemos utilizado la totalidad de las líneas de entrada/salida de nuestro PAL; sin embargo, era importante dejar algunas líneas de decodificación libres para expansión del sistema. Se emplearon compuertas lógicas debido a que la decodificación que se requería se podía implementar con pocas de ellas.

Para el control de escritura en el mapa de código se usa como ya se mencionó anteriormente una lógica de decodificación que permite traslapar los mapas de memoria de código y datos. Es decir, en el caso de que deseemos alterar el programa a ejecutarse (memoria de código), la lógica de decodificación permite desactivar la memoria de datos (DRAM) y activar en su lugar a la memoria de código (CRAM), de tal manera que todas las instrucciones de lectura y escritura a memoria de datos externa al microprocesador, desemboquen en el chip CRAM en lugar del chip

DRAM. Por ello utilizamos un bit (B6) del LATCH SO, y que denominamos L2, para indicarle a la lógica de decodificación cuando debe traslapar los mapas de memoria. El chip DRAM sólo recibe la señal de habilitación (Chip Select) cuando se selecciona a la mitad inferior del mapa de memoria (direcciones 0000H-7FFFH, es decir A15=0), y la línea de traslape de memoria está inactiva (L2=0). Por otro lado, la línea de habilitación de escritura para la CRAM (-WR CRAM) sólo se verifica cuando el microcontrolador manda la señal de escritura (-WR=0) y la línea de habilitación de cambio de mapa está activa (L2=1).

II.3 MEMORIA EEPROM.

Para ampliar el rango de aplicaciones del SD535 como herramienta auxiliar en el desarrollo de sistemas basados en microcontrolador, se decidió incluir la posibilidad de programar memorias EPROM. La manera más simple de conseguirlo fue la inclusión de un receptáculo en el que se pudiera colocar una memoria EEPROM, ya que la programación y el borrado de este tipo de memorias se puede realizar eléctricamente con la misma fuente de alimentación (Vcc=5V) que se emplea para el sistema.

Si bien por el momento este tipo de memorias resulta costosa en comparación con las UVEPROM, se compensa por el hecho de ser más rápida la reprogramación, y como mencionamos, es más fácil de implementarse en la tarjeta.

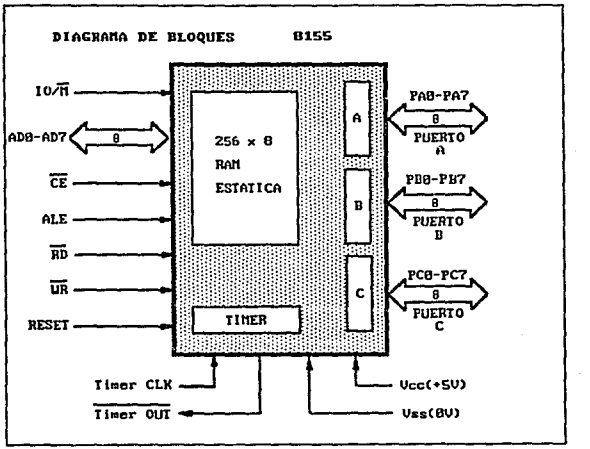
Considerando las áreas de memoria disponibles en nuestro mapa de memoria, el costo del chip, y su disponibilidad en el mercado, la memoria EEPROM que se seleccionó fue una 2864. Esta tiene una capacidad de 8 Kbytes, la cual es suficiente para almacenar programas de cierta complejidad.

En el mapa de memoria se observa que la EEPROM está referida dos veces, como código y como datos, esto es porque la operación de escritura se efectúa como si fuera datos y la lectura como si fuera código. Todo ello para facilitar su empleo e implementación.

II.4 PUERTOS DEL SISTEMA.

El microcontrolador cuenta, como se mencionó anteriormente, con seis puertos paralelos bidireccionales de 8 bits; sin embargo, dos de estos puertos se ocupan como buses de direcciones y datos multiplexados para el manejo de la memoria externa al microcontrolador. El puerto P0 se multiplexa para manejar el bus de datos y la parte menos significativa del bus de direcciones (A0-A7). A su vez, el puerto P2 maneja la parte más significativa

del bus de direcciones (A8-A15). Además de P0 y P2, se sacrifican dos líneas del puerto P3 (bits 6 y 7) para generar las líneas de escritura -WR y lectura -RD para la memoria de datos externa. Otras dos líneas del puerto P3 (bits 0 y 1) se emplean para la comunicación serial con la computadora central (PC). Es así como se utilizan dos y medio puertos para funciones propias del sistema, dejando para el usuario tres y medio puertos libres. No obstante, se pensó en incluir un puerto paralelo programable adicional para compensar esta situación. Entre las



posibilidades que se contemplaron estuvo la utilización del chip 8155 de Intel, el cual tiene 2 puertos paralelos bidireccionales

de 8 bits, 1 puerto de 6 bits de iguales características, 256 bytes de RAM estática, y un contador/temporizador de 14 bits.

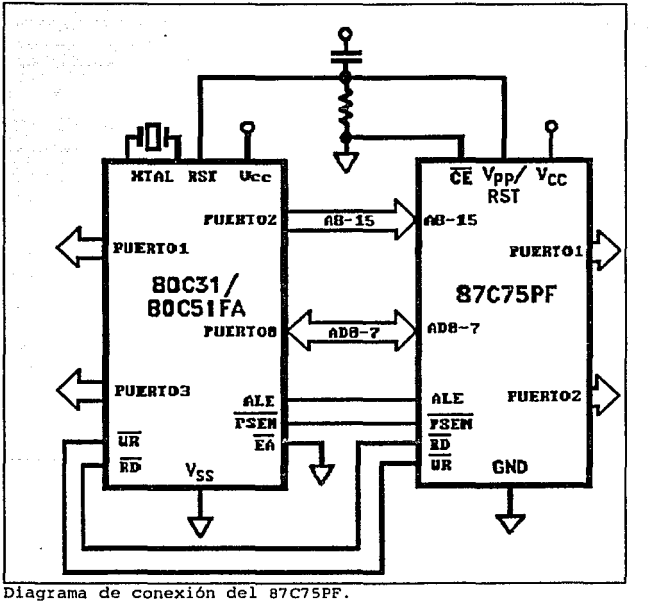


Diagrama de conexión del 87C75PF.

Otro chip que se consideró, fue el expansor de bus 87C75PF, el cual consiste de dos puertos de E/S de 8 bits configurables como Open Drain, bidireccionales y CMOS, además cuenta con 32

Kbytes de UVEPROM, y no necesita lógica de interfase. Sin embargo, los dos puertos que se añaden mediante este chip, deben ser direccionados como si fueran una localidad de memoria de datos externa. Por lo tanto, no se alcanza la potencia que presentan otros chips expansores de bus como el 68HC24, el cual es capaz de recuperar completamente los puertos perdidos al conectar memoria externa en un sistema basado en el microcontrolador 68HC11 de Motorola, permitiendo emular perfectamente las características de un microcontrolador con memoria interna. Desgraciadamente, no se encontró un circuito equivalente para el 80535.

Después de estudiar estas posibilidades, finalmente se optó por emplear un chip más popular como lo es el 8255. Éste es una interfase periférica programable (PPI) que cuenta con tres puertos paralelos bidireccionales de 8 bits, y tiene un uso muy extendido en el diseño de sistemas digitales. Además si consideramos al sistema de desarrollo desde el punto de vista didáctico, es preferible emplear componentes cuyo uso es más generalizado.

El microcontrolador 80535 cuenta, como ya mencionamos, con un convertidor analógico/digital de 8 bits y de 8 canales, lo cual le da la posibilidad de monitorear señales analógicas y efectuar tareas de control. Para extender la capacidad de control, se decidió incorporar al sistema de desarrollo una

salida analógica mediante un convertidor digital/analógico de 8 bits. Este tipo de convertidores se consigue fácilmente en el mercado nacional, sin embargo nos encontramos con una limitante. Durante la elaboración del diseño, se consideró la posibilidad de que toda la tarjeta del sistema de desarrollo funcionara con una sóla fuente de alimentación de 5 Volts. Un diseño con elementos 'tradicionales' requeriría de suministros adicionales de +12 V y - 12 V para los manejadores de la interfase serial RS-232C y para la etapa de conversión digital/analógica.

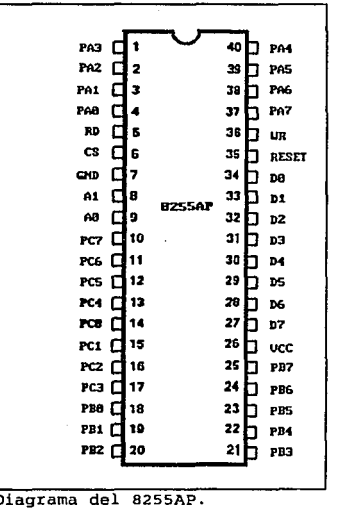


Diagrama del 8255AP.

El problema de la interfase serial se puede solucionar con el empleo del chip MAX232 que fabrica Maxim. Este chip se encarga de generar +10 V y -10 V a partir de una alimentaciónde +5 V, y está diseñado específicamente para producir los niveles de voltaje requeridos para una comunicación con el estándar RS-232C, partiendo de niveles lógicos TTL. El circuito es relativamente

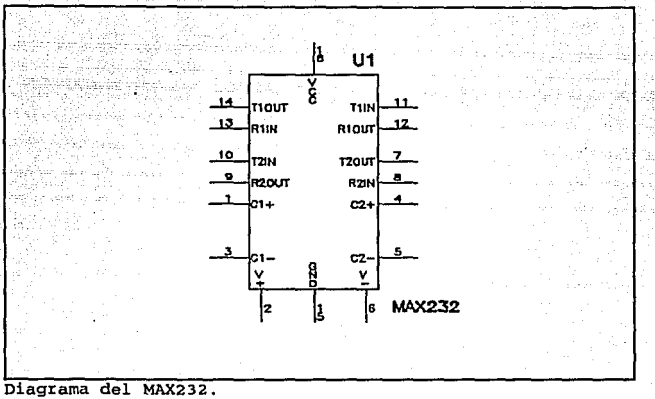


Diagrama del MAX232.

novedoso y está poco a poco extendiéndose su empleo. Para lograr su función emplea cuatro capacitores externos, un oscilador interno y unos interruptores analógicos. Los interruptores son abiertos y cerrados durante dos fases a una frecuencia de 16 KHz generada por el oscilador interno. Durante la primera fase se carga uno de los capacitores a +Vcc, para que en la segunda fase se sume el voltaje de este capacitor con el de alimentación y se carguen a +2Vcc los capacitores 2 y 3. Al repetirse la fase 1, se invierte la referencia a tierra de C3 y se usa para cargar a C4 con -2Vcc.

Con el empleo de este chip, sólo quedaba pendiente elegir un convertidor analógico/digital que pudiese funcionar con una sóla alimentación de 5 V.

Los convertidores analógico/digital más comunes como el MC1408,el DAC0800 y el DAC08 requieren una fuente de voltaje negativa, y además un amplificador operacional para transformar la señal de corriente que éstos entregan, en una señal de voltaje. Se buscaron alternativas que pudieran funcionar con las restricciones arriba descritas y una de las opciones fue el chip AD558 de Analog Devices. Éste es un convertidor analógico/digital de 8 bits compatible con microprocesador, es decir, cuenta con terminales de control que le permiten conectarse fácilmente al bus de un sistema con microprocesador. Se alimenta con una sóla fuente positiva de 5 a 15 V y cuenta con un amplificador operacional y una referencia de voltaje en el mismo chip, con lo cual se genera un sistema completo en un encapsulado de tan sólo 16 pines. Sin embargo, este circuito integrado no se consigue en el país, y aunque su fabricante lo etiqueta como un componente de bajo costo, al importarlo tenía un costo de $170,000. Fue por ello que se prefirió utilizar el DAC0830 que sí se consigue en el mercado nacional a un precio inferior ($32,300). Este convertidor, sin embargo, no tiene ni la referencia de voltaje ni un amplificador operacional integrado, por lo que deberán conectarse externamente. Otro inconveniente, es que aún y

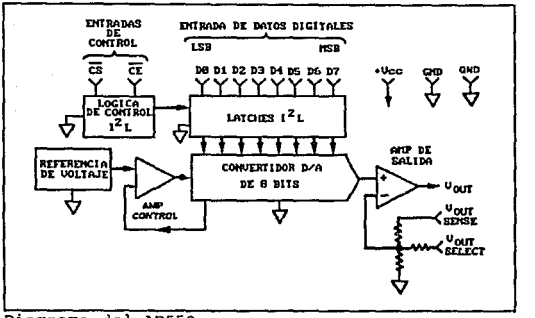


Diagrama del AD558.

cuando el fabricante (National) informa que su dispositivo es compatible con microprocesadores, después de analizar el diagrama de tiempos y su hoja de especificaciones, se descubre que su lógica es lenta comparada con la velocidad de transferencia de datos del microcontrolador. Cuando se alimenta con +5 V, la señal de -WR y los datos deben mantenerse estables por lo menos 900 ns, lo que representa más del doble de los 400 ns que proporciona el microcontrolador. Por lo tanto, y dado que el microcontrolador carece de la posibilidad de añadir estados de espera, además de los componentes mencionados, deberá incluirse un Latch para retener la información proveniente del microcontrolador y dar tiempo a que la reciba el convertidor. Para ello, se deberá configurar al DAC0830 en modo 'flow through'. En este modo, se vuelven 'transparentes' los dos latches internos que maneja el circuito integrado, de manera que el código de entrada se aplique

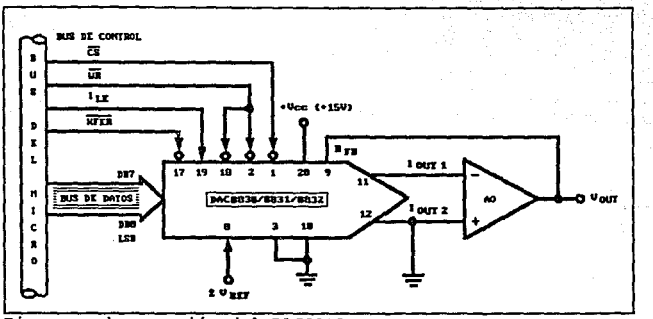


Diagrama de conexión del DAC0830.

directamente a la sección de conversión.

El DAC830 tiene una red de resistencias internas que le permite producir en su salida una corriente que es proporcional al código binario de entrada. Las configuraciones básicas requieren de un amplificador operacional que convierta esa señal de corriente en una señal de voltaje. En las notas de aplicación del fabricante se emplea un convertidor de corriente a voltaje inversor; esto es, la salida será un voltaje negativo proporcional al código binario de entrada. Esta situación resulta desfavorable a los objetivos de nuestro sistema de desarrollo, ya que para que el amplificador operacional pueda generar un voltaje negativo, deberá polarizarse con una fuente de alimentación bipolar.

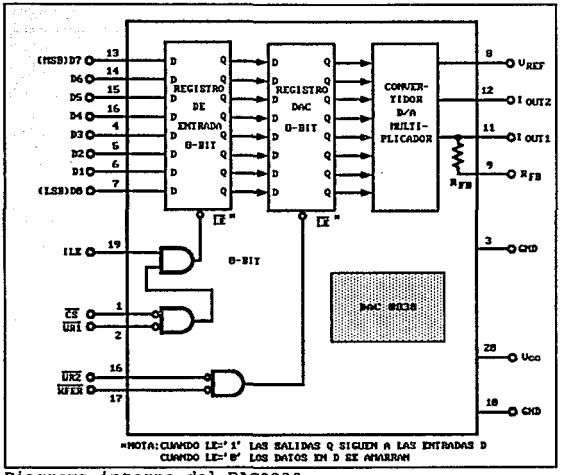


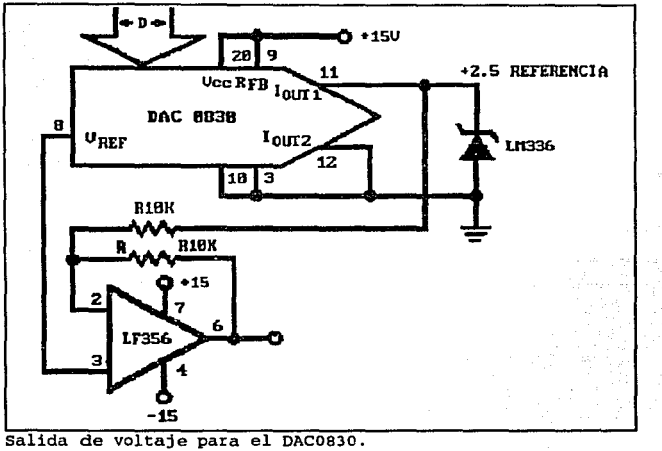
Diagrama interno del DAC0830.

Afortunadamente, la construcción interna del DAC0830 permite la implementación de una configuración en la que se produzca una salida de voltaje en lugar de una de corriente. Al emplear esta posibilidad junto con un amplificador operacional en configuración no-inversora, se puede generar una salida de voltaje positivo proporcional al código de entrada. En este caso, es posible utilizar un amplificador operacional polarizado con una sóla fuente, satisfaciendo nuestros requisitos de diseño.

Dado que el convertidor digital/analógico seleccionado requiere de una referencia de voltaje externa se analizaron las dos posibilidades más comerciales: el MC1403 fabricado por Motorola y el LM336-2.5 producido por National. Ambos componentes son referencias de voltaje integradas de 2.5 V con una tolerancia inicial de 1%. El MC1403 viene encapsulado en DIL (Dual In Line) de 8 pines y el LM336 en un empaquetado TO-98. Finalmente se optó por utilizar el integrado de National, debido a su encapsulado más pequeño y a que cuenta con una terminal de ajuste.

Para la selección del amplificador operacional apropiado estuvo delimitada por el requisito de que pudiese trabajar con una sóla fuente de alimentación. Además también es conveniente que la salida del amplificador pueda llegar a 0 V.

En los límites de excursión del voltaje de salida de la mayor parte de los amplificadores operacionales no se puede alcanzar el voltaje de alimentación. Por ejemplo, si polarizamos un μA741 con +-15 V, difícilmente la salida quedará fuera de +-14 V, ello debido a la arquitectura interna del circuito. Sin embargo, para nosotros resultaba de sumo interés que el voltaje de salida si pudiese bajar hasta los cero volts, aún y cuando coincidiera con el voltaje de polarización, ya que de esta forma un código binario de 00H podría estar representado por un voltaje de cero volts.



Salida de voltaje para el DAC0830.

Una de las opciones que se consideraron durante el diseño fue el amplificador operacional CA3140 de RCA con encapsulado DIP-8, el cual es un circuito BiMOS con etapa de entrada MOSFET que puede operar con una sóla fuente de 4 a 44 volts y su salida puede llegar a cero volts. Otra opción que se consideró fue el LM324 de National, el cual es un amplificador bipolar cuádruple que también satisface las condiciones de nuestro diseño. Además existe una versión con dos amplificadores en un DIP-8, el LM358.

De acuerdo con el Teorema del Muestreo, una señal se debe muestrear por lo menos al doble de su frecuencia máxima. El

convertidor analógico/digital que tiene ya integrado el microcontrolador 80535, efectúa una conversión en 15 µs, lo que implica un frecuencia máxima de muestreo de 66.666 Khz; es decir, la frecuencia máxima que se puede procesar es de 33 Hz. Si deseamos incluir una modesta capacidad de procesamiento de esa señal, digamos de unos 100 ciclos máquina, esta frecuencia debe bajar a los 10 KHz. Para impedir que se produzca el fenómeno conocido como traslape (aliasing), es decir, la interpretación de una señal de frecuencia mayor a la de muestreo como una de frecuencia menor, se suele incluir un filtro paso bajas para limitar el espectro de frecuencias de entrada. De igual manera, se suele colocar un filtro en las salidas de los convertidores digital/analógicos. En nuestro diseño, se optó por incluir un filtro paso bajas en la salida del convertidor D/A para limitar el ruido y suavizar la respuesta de salida. Durante la construcción de un prototipo se pudo apreciar un nivel importante de ruido debido principalmente a su acoplamiento a un sistema digital.

También durante el desarrollo del proyecto se probaron diferentes tipos de filtro (Butterworth, Chebyshev, y Bessel), de distintas frecuencias de corte y de diferente orden. Concordando con la teoría se encontró que el filtro que más reducía el ruido dados el mismo orden y frecuencia de corte, era el Chebyshev; sin embargo, el que mejor respetaba la forma de onda generada era el Bessel. El filtro Bessel al producir un

retraso de fase constante, es el que proporcionaba la mejor respuesta en el tiempo. Por lo tanto, fue el tipo de respuesta que se seleccionó. Este tipo de filtro no produce una atenuación tan importante, pero no obstante, al incrementar el orden del filtro no se producía una reducción substancial del nivel de ruido. Es por ello que finalmente se decidió emplear una etapa Bessel de segundo orden. De esta manera se trató de obtener la mayor disminución del ruido con el mínimo número de componentes.

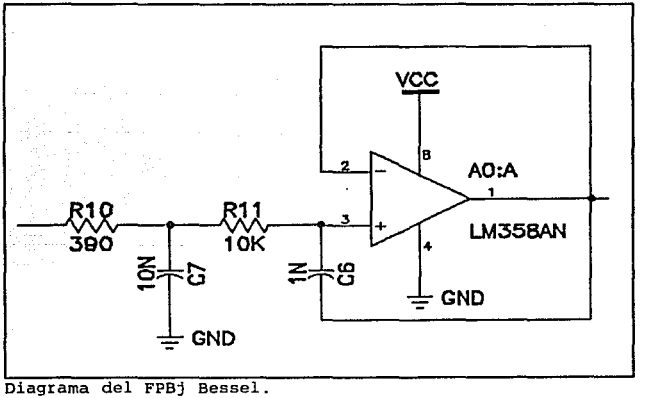


Diagrama del FPBj Bessel.

Con el fin de poder controlar el nivel del voltaje de offset de la señal generada, así como el nivel máximo de ésta se empleó

una configuración no-inversora con ganancia ajustable y una red resistiva que permite restar una componente de corriente directa a la salida; esto último con la finalidad de poder producir un voltaje igual a cero en la salida. Esta etapa se colocó entre el DAC y la etapa de filtrado, para que la impedancia de salida del DAC no influyera en el filtro.

El microcontrolador 80535 cuenta con una terminal denominada VPD, mediante la cual se le puede proporcionar un suministro de energía de respaldo para 40 localidades de la memoria RAM interna, mientras el suministro de voltaje a través de su terminal Vcc es suspendido. Esto permite poner al microcontrolador en un estado de espera que minimiza la energía consumida. Dentro del sistema de desarrollo no se incluyó una circuitería que hiciera uso de esta opción, sin embargo, se dejó la terminal VPD accesible a través de un conector.

El sistema de desarrollo SD535 cuenta con seis conectores. Estos incluyen uno para la línea de alimentación de voltaje de CD externa (+5V), uno para la conexión a un puerto serial de una computadora PC, uno para los puertos del PPI, otro para los puertos 1, 3, 4 y 5 del microcontrolador, otro más para las líneas de datos, direcciones y control del sistema, y finalmente uno para las entradas analógicas del microcontrolador (AN0-AN7), la línea de alimentación de respaldo, y la salida analógica del DAC.

La sección del convertidor analógico/digital del microcontrolador cuenta con dos terminales identificadas como VAREF y VAGND que sirven para definir los voltajes de referencia para la conversión de la señal. Estas terminales también están disponibles al usuario mediante el conector de entradas analógicas del microcontrolador, o bien, se pueden conectar a la fuente (Vcc) y la tierra del sistema por medio de unos jumpers (JP1 y JP2). En ese mismo conector podemos encontrar las terminales AN0 a la AN7 que son las entradas analógicas del convertidor, así como la salida de la sección del convertidor digital/analógico, VAN.

La interfase serial del microcontrolador permite programarla en cuatro modos distintos. De ellos, el modo uno es el más adecuado para implementar la comunicación serial con la computadora (PC). En este modo, se transmiten o reciben 10 bits a través de las terminales TXD o RXD respectivamente: un bit de inicio (0), 8 bits de datos (primero el LSB) y un bit de parada (1), siendo la tasa de transmisión variable. En el modo 1 la tasa de transmisión la fija la tasa a la que se desborda el temporizador 1 (Timer 1). Si el temporizador se programa en el modo de autorecarga, entonces la tasa de transmisión la define la ecuación:

$$Baudaje=\left(\frac{2^{smod}}{32}\right)*\left(\frac{Frec.osc.}{12[256-(TH1)]}\right)$$

Si hacemos el bit SMOD igual a uno, podemos generar los siguientes baudajes:

TH1	BAUDAJE	TH1	BAUDAJE
255	62 500	233	2 717
254	31 250	232	2 604
253	20 833	231	2 500
252	15 625	230	2 404
251	12 500	229	2 315
250	10 417	228	2 232
249	8 928	227	2 155
248	7 812	226	2 083
247	6 944	225	2 016
246	6 250	224	1 953
245	5 681	223	1 893
244	5 208	222	1 838
243	4 808	221	1 786
242	4 464	220	1 736
241	4 167	207	1 275
240	3 906	204	1 202
239	3 676	187	905
238	3 472	186	893
237	3 289	153	607
236	3 125	152	601
235	2 976	49	302
234	2 840	48	300

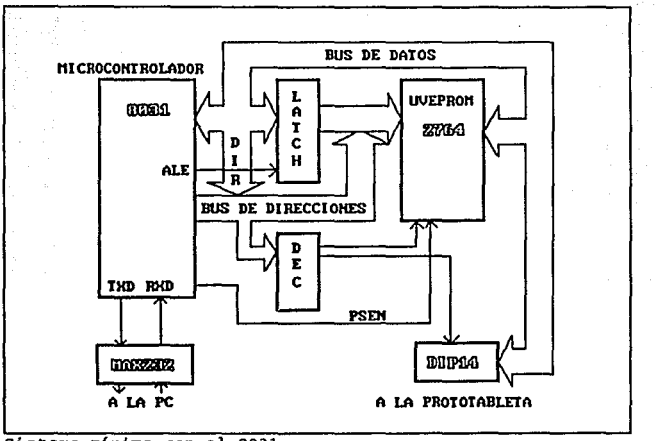
A partir de la tabla anterior, se observa que se pueden generar los baudajes estándar de 300, 600, 900, 1200, 2400 y 4800 con muy poco error, utilizando un cristal de 12 MHz para el reloj del microcontrolador. El manual de Intel sugiere el empleo de un cristal de 11.059 MHz para la obtención de baudajes exactos, sin embargo, se producirían ciclos de máquina con valores fraccionarios de tiempo, lo cual dificultaría el control del tiempo de ejecución y la elaboración de rutinas de retraso exactas.

Para el enlace por el puerto serial de la computadora PC, se decidió emplear una velocidad de 2400 bauds, ya que es una tasa de transmisión de una aceptable rapidez y debemos recordar que al aumentar la velocidad, disminuye la confiabilidad de los datos.

II.5 ENSAMBLADO DEL SISTEMA.

La construcción del prototipo consistió de dos etapas. Inicialmente se elaboró un sistema mínimo, mediante el cual pudiéramos probar aquellas partes de la arquitectura cuyo funcionamiento pudiera presentar una mayor incertidumbre. Posteriormente se procedió al diseño total del circuito impreso del sistema.

Para la primera parte se empleó una tarjeta de experimentación (modelo 152) perforada y con pads de cobre. En ésta se alambró y soldó el sistema mínimo que consistía en un microcontrolador, memoria UVEPROM, decodificador y un manejador (MAX232) de comunicación RS-232C. Lo primero que se probó fue la comunicación vía puerto serial con la computadora PC, ya que en ello se basa el funcionamiento del sistema de desarrollo SD535. Una vez que se verificó el funcionamiento de la configuración del MAX232 y de los programas de comunicación residentes en la memoria de la computadora y en la tarjeta, se



Sistema mínimo con el 8031.

procedió a probar la configuración del convertidor digital/analógico. De antemano sabíamos que al estar la salida analógica acoplada al sistema digital se iban a encontrar fenómenos de ruido, por lo tanto, era de suma importancia tratar de atenuar sus efectos antes de proceder al diseño final. Fue así que se probaron algunas configuraciones hasta elegir la que mencionamos anteriormente. Para facilitar la evaluación de la etapa de conversión D/A, se añadió un conector de cable plano a nuestro sistema mínimo con el que se tuvo acceso a una prototableta en la que se probó al convertidor y al filtro pasobajas. Aquí fue necesario usar la prototableta debido al constante intercambio de resistencias y capacitores que tuvimos

que realizar al intentar diferentes tipos de filtros en la salida del convertidor D/A.

Ya que se tenía definida la arquitectura del sistema de desarrollo se procedió al diseño del circuito impreso. Para ello se utilizó un programa denominado Tango, el cual nos permitió desarrollar el impreso de manera interactiva.

Después de seleccionar la ubicación de los componentes en la tarjeta, se empezó a efectuar el ruteo de las pistas. Para evitar ruido en las líneas de alimentación de voltaje debido a la conmutación de los circuitos lógicos, se colocaron capacitores de desacoplamiento entre las terminales de Vcc y Tierra (cerámicos de 0.1 μF), en las proximidades de todos los circuitos, y también un capacitor electrolítico de 47 μF en la tarjeta para respaldo y desacoplo de la fuente de alimentación. Las pistas encargadas de proporcionar la energía a todos los componentes del circuito se dibujaron lo más anchas posibles y alrededor de toda la tarjeta se reforzaron para minimizar los efectos de radiación y recepción de interferencia electromagnética. Como es usual en sistemas del grado de complejidad de éste, fue necesario el empleo de un circuito impreso con pistas en ambas caras; en una de las cuales se dibujaron todas las rutas verticales y en la otra se dió preferencia a las horizontales, esto tiene la finalidad de permitir un mayor número de cruces de pistas.

Con el circuito impreso ya fabricado se da paso a la verificación de las pistas. A menudo resulta desastroso un error en alguna pista una vez que los componentes se han montado en la tarjeta, ya que hace muy difícil su detección. Por lo tanto, lo primero que debemos hacer antes de soldar cualquier componente es comprobar que las pistas no tengan algún corto. El empleo de paquetes de diseño asistido por computadora como el Tango, permite verificar si no existe algún corto circuito entre las diferentes conexiones aún antes de fabricar la tarjeta. No obstante, es posible que debido a errores en el proceso de fabricación existan algunos pequeños hilos de cobre, casi imperceptibles a simple vista, pero capaces de cortocircuitar dos pistas con lo que se altera o impide el funcionamiento del sistema. El procedimiento que empleamos para comprobar el estado de las pistas fue primero, realizar una inspección visual de la tarjeta, y después verificar que no existiera corto entre pistas adyacentes, utilizando un probador de continuidad. Finalmente revisamos si todas las pistas coincidían con el diagrama esquemático original.

Después de haber revisado el circuito impreso, empezamos a montar los componentes. Para facilitar el cambio de cualquier dispositivo defectuoso, todos los circuitos integrados se montaron sobre bases. Inicialmente montamos sólo los componentes mínimos para que trabajara el microcontrolador, y una vez que se encontraban operando adecuadamente se añadían más componentes y

así sucesivamente, hasta que se montó la totalidad de los componentes.

Una vez que la tarjeta se encontraba funcionando, se procedió a la elaboración de las rutinas de programación, tanto en lenguaje ensamblador, como en Pascal; las cuales describimos brevemente en el siguiente capítulo.

CAPITULO III.

EL AMBIENTE DEL SD535.

La programación es una parte indispensable en los sistemas con microprocesadores y microcontroladores, ya que para poder operar es necesario que cuenten con un conjunto ordenado de instrucciones. En esta época en que la tecnología se inclina cada vez más hacia la digitalización, es importante que el ingeniero en electrónica sea capaz de elaborar programas tanto en lenguaje ensamblador como en alto nivel, de manera que le sea posible entender el funcionamiento integral de los modernos sistemas electrónicos.

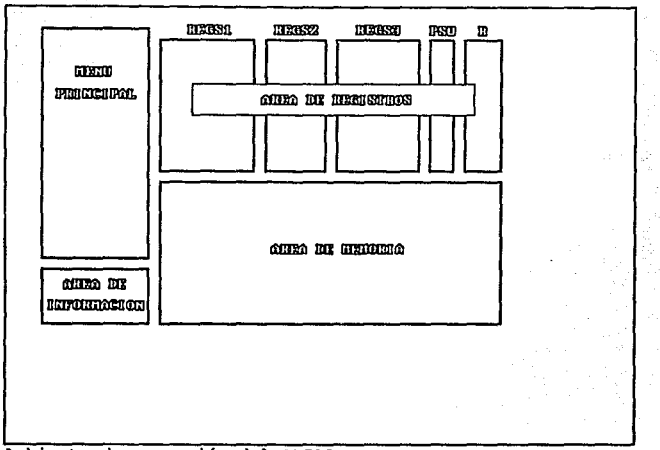
Dentro del sistema de desarrollo SD535, la programación es un área primordial, ya que de su desempeño depende la correcta operación de sus ventajas como herramienta en el diseño de sistemas basados en los microcontroladores 8031 y 80535.

Al ser un sistema controlado y monitoreado desde una microcomputadora PC, la programación estará formada por dos partes: lenguaje de alto nivel (Pascal) para la PC y ensamblador para la tarjeta con el microcontrolador.

Como se mencionó anteriormente, antes de diseñar el circuito impreso, probarlo y montar los componentes, ya se habían probado algunas rutinas de comunicación serial entre un sistema mínimo y la PC. Por lo tanto, una vez que se ensambló la tarjeta se partió de esas rutinas para el desarrollo total de la programación.

Como ya se tenían definidos los objetivos principales de la programación, se procedió a definir la distribución del ambiente de operación en la pantalla de la PC y después al desarrollo de las rutinas en Pascal y ensamblador. La tarea de programación resultó así, doblemente compleja, ya que se tenían que desarrollar simultáneamente rutinas para la computadora y rutinas para la tarjeta, y en caso de error, este podía encontrarse en cualquiera de las dos partes.

El ambiente tradicional de operación de los sistemas de desarrollo, basados en terminal de video, consiste únicamente en la posibilidad de introducir mediante el teclado un comando y, de ser el caso, observar alguna información como respuesta en el monitor. Este método para controlar al sistema de desarrollo era susceptible de mejorarse, y fue lo que intentamos hacer al proponer un ambiente de operación mucho más fácil de usar y más versátil.



Ambiente de operación del SD535.

Para hacer el ambiente más amigable se programaron menús de selección por barras y un ambiente de ventanas. En los menús de barras se tiene la posibilidad de señalar un comando mediante el posicionamiento de la barra (video inverso), con las teclas del cursor, en la opción deseada, o bien, presionando la primera letra de la misma.

Para lograr este tipo de menús existen varias alternativas. El algoritmo que se empleó lo describiremos a continuación. Primero se define un área de la pantalla en la que aparecerá el menú y se dibujan los contornos. Después se asignan las

opciones a un arreglo (matriz) de opciones y éstas se despliegan en la pantalla, activándose la barra en la primera selección. A continuación una rutina de lectura del teclado se encarga de decidir si se presiona una tecla de comando, una tecla de cursor, o una tecla inválida. Las teclas del cursor constan de un código doble, por lo tanto, en caso de detectarse un código 0 (primera parte del código doble) deberá efectuarse un segundo reconocimiento del buffer del teclado para poder identificar el tipo de caracter especial de que se trate. En el menú de barras se lleva un control de la posición de la barra, de tal manera que se sepa cual es la opción precedente y cual es la siguiente para que en caso de solicitarse mediante el cursor, se pueda actuar correctamente. También se controla para el caso de que si las opciones se terminan, se reinicie el movimiento de la barra. Por ejemplo, si se tienen 10 opciones, la barra apunta a la opción 10 y se presiona la tecla del cursor hacia abajo (siguiente opción en orden ascendente), la barra reaparezca en la primera opción. La rutina también cubre el caso en que la barra apunte a la primera opción y se presione la tecla de cursor hacia arriba (siguiente opción en orden descendente), en cuyo caso la barra reaparecería en la última posición (opción 10 en el ejemplo). Asimismo la rutina de selección no distingue entre letras mayúsculas o minúsculas. Ésto asegura que al presionar una tecla se activará la opción deseada, sin importar si el teclado tenía o no el seguro de mayúsculas. Si se presiona la tecla ENTER se activará la opción señalada por la barra.

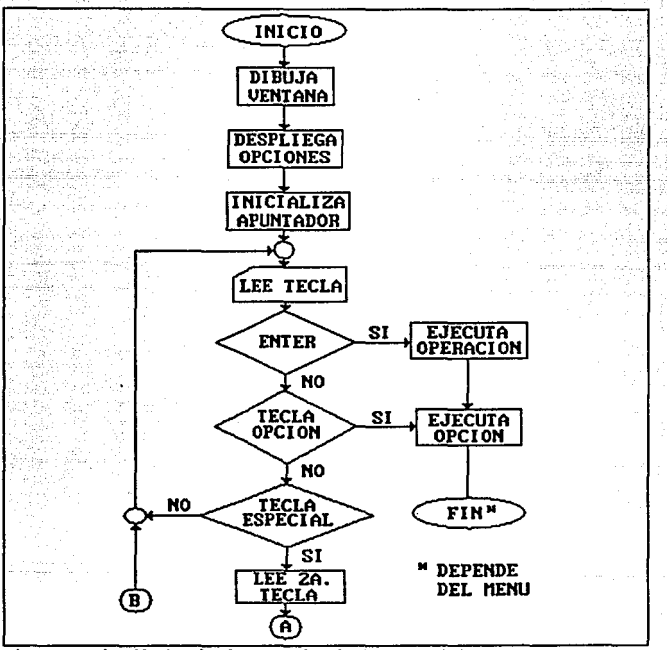


Diagrama de flujo de los menús de barras (I).

La misma filosofía del menú de barras se emplea para el menú principal y para todos los submenús. Sin embargo, se debieron hacer ajustes tales como: ubicación en la pantalla, tamaño, número de opciones, teclas reconocibles, así como la respuesta a cada una de ellas.

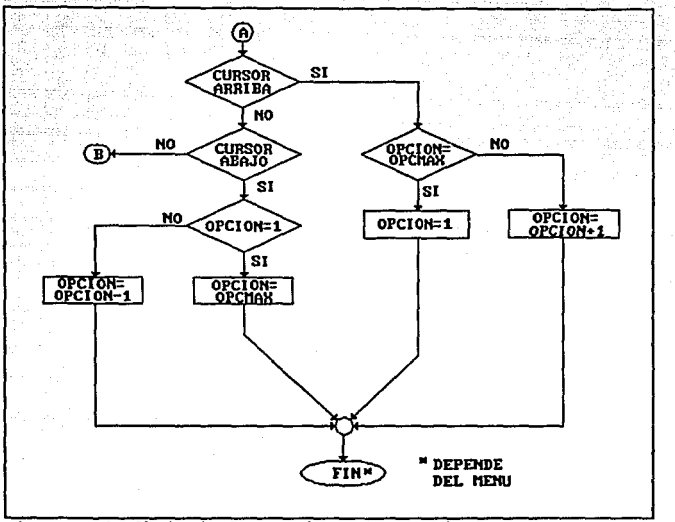
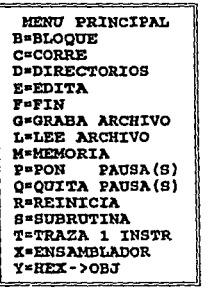


Diagrama de flujo de los menús de barras (II).

III.1 MENU PRINCIPAL

En el menú principal se encuentran todos los comandos de control del sistema de desarrollo, algunos de los cuales tienen

submenús más específicos. La mayoría activa una ventana de diálogo a través de la cual se especifican las funciones a realizar. El menú principal se encuentra activo siempre, a menos que se esté ejecutando algún comando, o bien, se le haya transferido el control de la tarjeta a algún programa del usuario. A continuación se muestra la apariencia del menú principal y enseguida se detalla el funcionamiento de cada uno de los comandos.



VENTANA DEL MENU PRINCIPAL DEL SD535.

III.2 MENU DE BLOQUES (B)

Este menú consiste de seis opciones: copia memoria de datos, copia memoria de código, copia de memoria de código a EEPROM (PROGRAMACION), copia de EEPROM a memoria de código, llena un bloque de datos y llena un bloque de código.

```
1=COPIA DAT
2=COPIA COD
3=COD-EEPROM
4=EEPROM-COD
5=LLENA DAT
6=LLENA COD
```

En las opciones de copia de memoria de código y copia de memoria de datos, se incluye un algoritmo de origen-destino, el cual determina la forma en que se tiene que realizar el traslado de la información byte a byte, de tal manera que no se pierda información.

Una vez que se selecciona la opción el programa le pide al usuario introduzca en hexadecimal las direcciones inicial y final del bloque que se quiere copiar, así como la dirección inicial a partir de la cual se copiará el bloque y calcula la dirección final. Si en algún punto de la ventana de diálogo se presiona <ENTER> o <ESC> se cancelará el comando.

Además el programa permite la corrección de las direcciones. Para ello bastará con presionar la tecla <BACKSPACE> y reescribir el caracter correcto. Una vez que se teclea el último caracter de la dirección, automáticamente se acepta el valor sin necesidad de teclear <ENTER>. Esto impide corregir, en su caso, un error en el último caracter de la dirección, pero en la generalidad de los casos, evitará el uso adicional de la tecla <ENTER>. Además la programación se simplificaba en este caso.

Ahora bien, supongamos que el bloque inicial (fuente) está definido por las direcciones DIR1-DIR2, donde DIR2>DIR1, de lo contrario se pide al usuario que rectifique las direcciones; y supongamos también que se quiere mover este bloque a las localidades que comienzan con la dirección DEST1. Entonces si tenemos que DEST1<DIR1 entonces el bloque se deberá desplazar hacia arriba en el mapa de memoria (donde la parte más alta sería la dirección 0000H), por lo tanto para evitar el caso de que se traslapen los bloques, el primer byte a copiarse será el primer byte del bloque y se irá incrementando la cuenta hasta copiar el último byte del bloque.

En el caso de que DEST1>DIR1 entonces el bloque deberá ser desplazado hacia abajo y para evitar la pérdida de información debido al traslape de los bloques, el primer byte a copiarse será el último byte del bloque y la cuenta se irá decrementando hasta que se copie el primer byte del bloque.

La tercera opción de este submenú, nos da la posibilidad de programar una memoria EEPROM copiando en ella el contenido de una porción de la memoria de código. El programa se encarga de validar las direcciones fuente, de tal manera que no se intente copiar un bloque mayor a 8 kbytes de memoria, que es la capacidad máxima de la EEPROM.

Para el manejo de bloques que impliquen el empleo de la EEPROM, se decidió aislarla de su ubicación en el mapa de memoria de la tarjeta, de tal manera que el usuario pudiese prescindir de este dato. Así, para el usuario existirá un mapa independiente para la EEPROM con direcciones 0000H-1FFFH, lo cual evita que se tengan que hacer los cálculos de direccionamiento con direcciones absolutas (A000H-BFFFH).

En las opciones de llenado de memoria de código y memoria de datos, simplemente se le solicitan al usuario las direcciones inicial y final, así como el byte en hexadecimal con el cual deberán llenarse las localidades.

III.3 OPCION DE EJECUCION DE PROGRAMAS (C)

En esta opción se transfiere el control de la tarjeta a un programa que empieza en una dirección que se le solicita al usuario. En caso de teclear <ENTER> se transfiere el control a partir de una dirección por omisión, la cual al ingresar al sistema es igual a 0000H, y después es igual a la última dirección inicial aceptada por el programa o la siguiente dirección a ejecutarse si se encontró una pausa.

Al transferir el control éste sólo se puede devolver al monitor del sistema de desarrollo por alguna de las tres formas siguientes:

-Que el programa del usuario termine con una instrucción LCALL 8100H.

-Que se hayan programado pausas y se llegue a una durante la ejecución.

-Presionando la tecla <ESC> y dándole reset a la tarjeta (NOTA: esto último puede borrar la información obtenida durante la ejecución).

En los primeros dos casos, la ejecución terminará con la actualización de todos los registros y memoria activos en la pantalla y con un mensaje indicando hasta que localidad se ejecutó el programa, además se preguntará al usuario si desea o no proseguir con la ejecución a partir de ese punto. En el tercer caso, se retomará el control de la tarjeta de una manera similar a si se acabara de conectar, con la diferencia de que los programas en memoria permanecerán.

III.4 MENU DE DIRECCIONES (D)

Este submenú permite seleccionar una de tres opciones como subdirectorio para las operaciones con archivos que se realicen.

```
A:\
B:\
C:\SD535\
```

Los archivos tanto de código como de datos, así como los programas ensamblador y convertidor de formato HEX a OBJ (ver opción Y) se buscarán en el directorio activo, el cual se indica siempre en el recuadro de información.

```
CORRE:  0000
  PC:   0000
PAUSAS  NO
C:\SD535\
```

RECUADRO DE INFORMACION

III.5 MENU DE EDICION (E)

Este menú consiste de seis opciones mediante las cuales se pueden modificar los contenidos de los registros internos del microcontrolador, la memoria de datos, la memoria de código, la memoria RAM interna del microcontrolador de direccionamiento directo (00-7FH) y la memoria RAM interna de direccionamiento indirecto (80-FFH), así como alterar la dirección de ejecución de la modalidad paso a paso (TRAZA). La opción de edición de registros consiste de un submenú, el cual permite seleccionar de una manera más específica uno de cinco grupos en que están divididos los registros internos.

MENU DE EDICION

III.5.1 SUBMENU DE EDICION DE REGISTROS (R)

En el submenú de edición de registros se presentan las siguientes opciones:

OPCIONES DE EDICION DE REGISTROS

Como una de las ventajas de la implementación del SD535, se tiene que todos los registros internos del microcontrolador susceptibles de modificarse permanecen desplegados siempre en pantalla, lo cual permite un monitoreo continuo del estado del microcontrolador. Los registros SCON y SBUF no aparecen en la pantalla y no son alterables por comandos del ambiente del SD535, debido a que se reserva su uso al sistema monitor, en tanto que son utilizados como vía de comunicación por el puerto serial. En

el caso de que estos registros sean modificados por medio de la ejecución de algún programa del usuario, se corre el riesgo de perder el control sobre la tarjeta.

De igual manera, no aparecen en la pantalla los registros P0 y P2, debido a que son utilizados por el sistema para direccionar memoria externa al microcontrolador.

A continuación describiremos brevemente el funcionamiento de cada una de las opciones de este submenú.

Con la primera opción tenemos acceso a modificar los registros A, B, SP, TMOD, IP0, IP1, CCEN, PCON, ADDAT, y DAPR.

REGS1

```
 ACC=00000000
   B=00000000
  SP=00000111
TMOD=00100000
 IP0=10000000
 IP1=11000000
CCEN=00000000
PCON=01111111
ADAT=00000000
DAPR=00000000
```

Para estos registros, así como para los de la tercera opción se muestra su contenido en formato binario, debido a que la mayor parte de ellos son direccionables por modo bit (se puede manipular un sólo bit de manera independiente) o bien, cada bit representa una función especial o estado del microcontrolador.

Es posible modificar el contenido de estos registros tanto en binario como en hexadecimal. Para la modificación en binario, basta con que nos posicionemos en el bit de interés mediante las teclas del cursor e introduzcamos el nuevo valor mediante el teclado. Si modificamos el último bit de un registro, automáticamente el cursor se posiciona en el primer bit del siguiente registro, y si nos encontrábamos en el último registro, automáticamente saldremos del menú. En caso de que se desee abreviar la modificación de los registros existe la posibilidad de introducir los valores en hexadecimal; para ello, se debe ubicar el cursor en el primer bit del registro (bit más significativo) y presionar la tecla "H" para indicar valores en hexadecimal. Entonces aparece una letra "H" intermitente en la posición del cursor y podremos teclear los dos caracteres hexadecimales que correspondan al nuevo contenido del registro. Después de introducir los caracteres el cursor se ubicará en el siguiente registro y se desactivará el modo hexadecimal. Al igual que en el caso binario, si nos encontramos en el último registro, después de modificarlo automáticamente saldremos del menú.

En la segunda opción podemos modificar los registros DPTR, T0, T1, T2, CC1, CC2, CC3, y CCR. Como todos estos registros son de 16 bits, están desplegados en hexadecimal para ahorrar espacio, ya que se buscó poder mostrar todos los registros de manera simultánea. Para la modificación de estos registros sólo existe la modalidad hexadecimal, sin embargo, en este caso no es

necesario presionar la tecla "H", ya que es la única modalidad disponible.

REGS2

```
DPTR=0000
  T0=0000
  T1=F3F8
  T2=0000
 CC1=0000
 CC2=0000
 CC3=0000
 CRC=0000
```

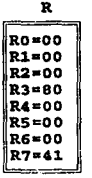
En la tercera opción tenemos a los registros TCON, T2CON, IEN0, IEN1, IRCON, P1, P3, P4,y P5, los cuales funcionan de la misma manera que los registros REGS1. Como se observará, no están incluidos los registros de los puertos P0 y P2, ya que estos se utilizan para el manejo de memoria externa al microcontrolador y por lo tanto no están disponibles para el usuario.

REGS3

```
TCON=11000000
T2CO=00000000
IEN0=00000000
IEN1=00000000
IRCO=00111110
  P1=00001100
  P3=11111111
  P4=11111111
  P5=11111111
ADCO=00000000
```

La opción número cuatro corresponde a un sólo registro: el PSW. Por medio de esta opción podemos modificar cada uno de los bits del PSW, al mismo tiempo que ubicamos su función con un identificador para cada uno. La modificación se realiza ubicando el bit a modificar con el cursor y tecleando el nuevo valor (1 ó 0). Después de modificar un bit, el cursor se mueve automáticamente al siguiente. Al modificar el último bit, se abandona al submenú.

Por último, la quinta opción nos muestra el contenido de los registros 'R' del banco (0, 1, 2 ó 3) activo en ese momento. La alteración de un registro en específico se realiza con ayuda de las teclas del cursor e introduciendo el nuevo contenido en formato hexadecimal.



III.5.2 SUBMENU DE EDICION DE DATOS (D)

En este submenú podemos modificar el contenido de la memoria de datos. Para ello el programa pregunta por la dirección inicial a partir de la que se quieren realizar modificaciones. Si se presiona la tecla <ENTER> se tomará la dirección inicial actual del apuntador a memoria de datos (inicialmente 0000H).

```
DIREC INICIAL
DATOS 0000
NUEVA DIRECCION
DATOS _
```

Una vez que se selecciona la dirección inicial, un bloque de 128 localidades de memoria a partir de la dirección señalada se despliega en la ventana de memoria y aparece un cursor (caracter intermitente) en la primera localidad del bloque. Para hacer cualquier modificación en la localidad de memoria señalada por el caracter intermitente basta con teclear el nuevo dato en hexadecimal. Si se desea modificar alguna otra dirección dentro del bloque se pueden emplear las teclas del cursor para seleccionar el byte deseado y después modificar el contenido de la localidad de manera hexadecimal. Una vez que se modifica una localidad, el cursor automáticamente se posiciona en la siguiente dirección, y después de alterar la última dirección del bloque se regresa al menú principal. Una vez que se abandona la edición de memoria de datos, ésta permanecerá activa en la ventana de

memoria, sin importar que memoria estaba activa antes de entrar al menú.

DATOS	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000 :	00	0F	0B	0F	04	4F	27	0F	41	41	41	41	41	41	41	41
0010 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0020 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0030 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0040 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0050 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0060 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41
0070 :	0F	0F	0F	0F	0F	0F	0F	0F	41	41	41	41	41	41	41	41

Los datos están arreglados de tal manera que en el primer renglón tenemos los contenidos de las direcciones 0000, 0001, 0002, etc, en el segundo renglón los contenidos de 0010, 0011, 0012, etc, y así sucesivamente.

III.5.3 SUBMENUS DE EDICION DE CODIGO (C), MEMORIA RAM INTERNA 1 (M) Y MEMORIA RAM INTERNA 2 (N)

Estos submenús funcionan de manera similar al de edición de datos, con la diferencia de que trabajan con 128 bytes de la memoria de código o con los primeros 128 bytes de la RAM interna del microcontrolador, o bien con los segundos 128 bytes de la RAM interna.

III.5.4 OPCION DE INCREMENTO Y DECREMENTO EN LA MEMORIA DESPLEGADA

Para agilizar el análisis del contenido de la memoria de código y datos, se incluyó una opción de auto incremento en la dirección de despliegue.

Si presionamos la tecla de suma <+>, automáticamente se desplegará en la ventana de memoria los siguientes 128 bytes, de datos o código según corresponda. Por otro lado, si presionamos la tecla de menos <->, se desplegará en la pantalla los 128 bytes anteriores. Mediante estas teclas, el análisis del contenido de la memoria en bloques adyacentes se simplifica, y se vuelve más ágil.

III.6 OPCION DE FINALIZAR (F)

Al seleccionar la opción F del menú principal se dará por concluida la ejecución del programa SD535.EXE y se retornará al sistema operativo de la computadora PC.

III.7 MENU DE GRABACION DE ARCHIVOS (G)

En este menú se puede grabar en disco el contenido de la memoria de código (programas) o de la memoria de datos. La

ventana de diálogo pregunta las direcciones inicial y final, así como el nombre del archivo, y si se teclea <ENTER> se toma el archivo por omisión (datos.dat o prog.obj). Si el archivo ya existe, preguntará si se desea sobreescribirlo y en caso afirmativo procede a grabar la información (se utilizará el directorio que esté activo).

```
DIRECCION INICIAL(HEX): 0000
DIRECCION FINAL  (HEX): 003F

 CUAL ES EL NOMBRE DEL ARCHIVO
[C:\SD535\DATOS.DAT]? _
```

III.8 MENU DE LECTURA DE ARCHIVOS (L)

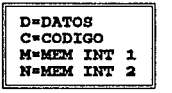
Aquí se tiene la opción de leer archivos y transferirlos a la memoria de datos o a la memoria de código (programas). Primero se pide el nombre del archivo (si se teclea <ENTER> se tomará la opción por omisión), si éste existe pregunta la dirección a partir de la cual se debe cargar, y si no, se da un mensaje de advertencia. Una vez que se lee el archivo se actualiza la ventana de memoria, y se activa según corresponda (código o datos).

```
 CUAL ES EL NOMBRE DEL ARCHIVO
[C:\SD535\PROG.OBJ]?

 ERROR: NO SE ENCONTRO EL ARCHIVO
```

III.9 MENU DE SELECCION DE MEMORIA ACTIVA (M)

Este menú permite seleccionar el banco de memoria a desplegar. La ventana de memoria comprende 128 bytes y puede ser alguna de las siguientes opciones: memoria de código, memoria de datos, primeros 128 bytes de la memoria RAM interna o segundos 128 bytes de la memoria RAM interna del microcontrolador. En cada uno de los casos los datos serán desplegados con el mismo formato que se explicó en el párrafo relativo a la edición.



Una vez que se selecciona un banco de memoria, éste permanecerá activo (desplegado en la pantalla) hasta que se seleccione otro banco, o se efectúe una operación de lectura o escritura de memoria que involucre un área distinta (código o datos), en cuyo caso será esta última la que se active.

III.10 OPCION DE ACTIVACION DE PAUSAS (P)

Al entrar a esta opción aparece una ventana en la que se muestran las direcciones de inserción de ocho posibles puntos de ruptura o pausas en el código del programa. Inicialmente la dirección señalada por las pausas es 0000H, que se interpreta

como pausas inactivas. Es decir la dirección 0000H no es válida para introducir una pausa.

```
PAU1=0000
PAU2=0000
PAU2=0000
PAU3=0000
PAU4=0040
PAU5=0300
PAU6=0400
PAU7=0700
PAU8=2000
```

Para modificar y activar alguna pausa basta con desplazarse con las teclas del cursor e introducir la nueva dirección en formato hexadecimal. Al salirse del menú ,si existen direcciones válidas, automáticamente se activarán las pausas, lo cual se indicará mediante la ventana de información. Después de activar las pausas, se selecciona automáticamente la memoria de código en la ventana de memoria, con la finalidad de que el usuario compruebe que se han introducido las pausas indicadas (código 12 81 00).

Esta misma opción puede servir para verificar la direcciones en las que se ha insertado una pausa. Para ello bastará con volver a entrar a esta opción. En la ventana aparecerán las direcciones de las pausas introducidas ordenadas en orden ascendente de arriba a abajo, tal y como se observó en el recuadro anterior.

III.11 OPCION DE DESACTIVACION DE PAUSAS (Q)

Esta opción permite desactivar las pausas y restablecer el código original del programa del usuario. Al desactivar las pausas se indica en la ventana de información el mensaje 'PAUSAS NO', y se activa la memoria de código en la ventana de memoria para que el usuario pueda comprobar que las pausas han sido desactivadas (se restablece el código original).

III.12 OPCION DE REINICIO (R)

Esta opción permite la actualización de todos los datos desplegados en la pantalla, ya que se encarga de solicitarlos nuevamente a la tarjeta. Sirve para renovar, comprobar o restablecer la información en caso de algún error.

III.13 OPCION DE EJECUCION DE SUBRUTINA (S)

Existen ocasiones en que sólo nos interesa verificar el comportamiento de una subrutina que estemos desarrollando; es por ello que se implementó la opción de ejecución de una subrutina. Esta opción nos permite transferir el control de la tarjeta a una subrutina definida por el usuario. La ventana de diálogo le solicita al usuario la dirección de inicio o en su caso toma la dirección por omisión(0000H o la última dirección solicitada).

El control de la tarjeta será retomado por el monitor del sistema de desarrollo en el momento en que se encuentre la instrucción de retorno de subrutina 'RET'.

III.14 OPCION DE EJECUCION PASO A PASO 'TRAZA' (T)

Para la depuración de ciertos procesos resulta imprescindible contar con la posibilidad de ejecutar una rutina instrucción por instrucción y así poder rastrear y monitorear el funcionamiento correcto de la misma. La ejecución paso a paso es una de las características más útiles de un sistema de desarrollo.

Esta opción toma como dirección de inicio de la instrucción por ejecutarse aquella definida en el submenú de memoria [opción P=PC (TRAZA)]. Después de ejecutarse la instrucción, automáticamente se actualizan los valores de los registros y del mapa de memoria activo, para que el usuario pueda verificar el resultado de la misma; también aparece indicada la dirección de la siguiente instrucción por ejecutarse, la cual se tomará como dirección de inicio en caso de volverse a solicitarse una ejecución paso a paso.

El comando 'TRAZA' es capaz de identificar correctamente la dirección de la siguiente instrucción por ejecutarse, aún y

cuando se traten de saltos condicionales, gracias al algoritmo que desarrollamos para tal efecto.

III.15 OPCION DE ENSAMBLAR PROGRAMA (X)

La opción de ensamblado consiste en la posibilidad de ejecutar un programa ensamblador que se encuentre en el directorio activo, con la finalidad de cargarlo posteriormente en el sistema de desarrollo, y sin necesidad de abandonar el ambiente del sistema de desarrollo (programa SD535). El programa SD535 está ajustado para reconocer al ensamblador CYS8051.EXE, que deberá encontrarse en el directorio activo. En caso de contarse con otro, deberá cambiarse su nombre para que sea reconocido. Si no se encuentra el ensamblador, se retornará inmediatamente al ambiente del SD535 sin darse ningún mensaje de error adicional.

Mediante el empleo de un editor de texto residente en memoria como el SideKick, podemos crear y/o modificar un programa en ensamblador, manteniendo activo el programa SD535. Una vez realizado el programa, se puede salvar al archivo y abandonar al editor (residente). Es así como mediante la opción de ensamblado podremos correr un programa que realice la conversión a código máquina de nuestro programa y retornar al programa SD535 automáticamente.

El ensamblador arriba mencionado entrega como resultado un archivo en formato HEX, el cual no es apropiado para bajarse directamente a la memoria de nuestra tarjeta. Es por ello que una vez ensamblado el programa, deberemos ejecutar una opción de conversión de formato HEX a formato OBJ, siendo este último el adecuado para bajarse a la tarjeta.

III.16 OPCION DE CONVERSION DE FORMATO HEX A OBJ (Y)

Esta opción permite ejecutar un programa de conversión de archivos en formato .HEX a formato .OBJ sin necesidad de abandonar la ejecución del programa SD535. El programa de conversión se debe encontrar en el directorio activo y nombrarse HEXOBJ.EXE.

Usada en conjunción con la opción de ensamblado y el editor de textos residente en memoria (SideKick), el sistema de desarrollo SD535 se convierte en un poderoso ambiente integrado para la creación y depuración de programas para los microcontroladores 8031 y 80535.

CAPITULO IV.

EL PROGRAMA MONITOR.

Para controlar todas las funciones de la tarjeta e interpretar todas las instrucciones que provienen de la computadora PC, se desarrolló un programa en ensamblador que reside en una memoria UVEPROM 2764 y que constituye el sistema monitor de la tarjeta, el cual describimos brevemente a continuación.

IV.1 INICIALIZACION

Después de darle RESET a la tarjeta, el chip que contiene al programa monitor (CROM1) se encuentra direccionado en las localidades 0000H a 1FFFH y simultáneamente en las localidades 8000H a 9FFFH, es decir que el primer byte del chip se encuentra ubicado al mismo tiempo en la dirección 0000H y en la 8000H.

La primera instrucción que se ejecuta es un salto incondicional a la dirección 8004H, esto permite que a partir de este momento todas las instrucciones del sistema monitor se ubiquen en la segunda área (localidades 8000H-9FFFH). En seguida se salvan los registros PSW, B, R1, DPL, DPH, R0, y ACC en la

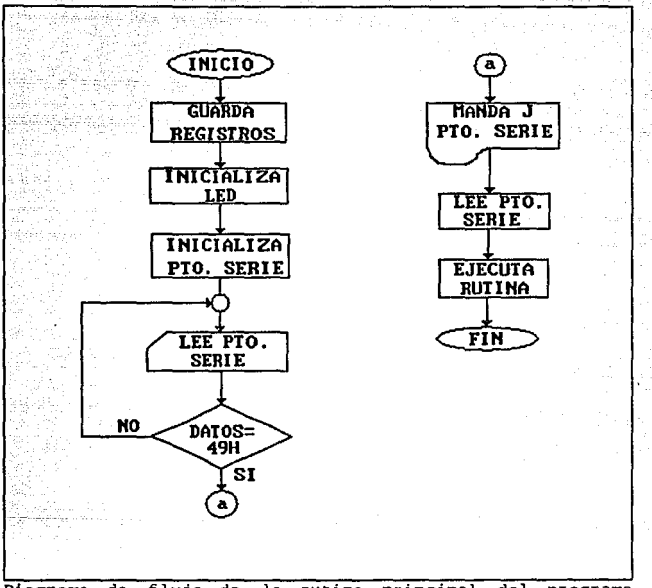


Diagrama de flujo de la rutina principal del programa monitor.

pila (STACK). Es necesario respaldar estos registros ya que continuamente serán empleados por el programa monitor; sin embargo, cada vez que se tenga que leer o escribir en los registros del usuario (que se encuentran respaldados en la pila), estos se sacarán de la pila para que los pueda usar, y antes de que el programa monitor retome el control del sistema, se encargará de guardar los registros en la pila.

A continuación se efectúa la inicialización del puerto serie para poder comunicarse con la computadora central PC y espera a recibir el protocolo adecuado y un comando. Una vez recibido se va a la tabla ubicada a partir de la dirección 8300H y se efectúa un salto a la rutina correspondiente.

IV.2 MANEJO DE DATOS Y CODIGO

Las primeras rutinas de la tabla corresponden a la lectura y escritura de datos, ya sean estos registros del microcontrolador, memoria RAM interna del mismo, memoria de datos externa, o memoria de código externa.

La estructura que tienen estas rutinas es básicamente la misma. Una vez que se activan esperan, de ser necesario, datos adicionales, como lo podrían ser direcciones específicas de memoria. Después, siguen el protocolo necesario para enviar o recibir la información solicitada a la PC.

Para el caso de memoria de datos y memoria de código externas al microcontrolador (CRAM y DRAM) se envían y reciben 128 bytes para lectura y escritura respectivamente.

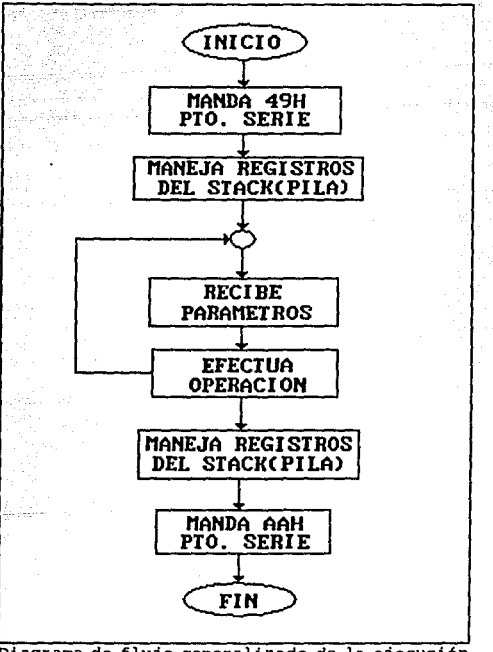


Diagrama de flujo generalizado de la ejecución de una subrutina del programa monitor.

Cuando se trata de la lectura y escritura de registros internos del microcontrolador siempre se reestablecen los contenidos originales sacándolos de la pila, y una vez que se

ha realizado el intercambio de la información se vuelven a guardar en la pila.

Las subrutinas de copia de bloques de memoria de código y memoria de datos se subdividen en dos cada una. Una de ellas es dedicada a la copia hacia direcciones menores del mapa de memoria (copia hacia arriba), y otra especializada en la copia de la información hacia direcciones mayores (copia hacia abajo). Esta división tiene la finalidad de evitar la posible pérdida de información debido al traslape de los bloques. Para el manejo de la información se hace un uso extenso de la pila (STACK) para evitar tener que utilizar otros registros del microcontrolador además de los ya definidos para el programa monitor.

En el diagrama se observa que cuando la nueva ubicación del bloque de datos será en una dirección menor del mapa de memoria (el bloque se desplaza hacia arriba), es necesario empezar por mover el primer byte del bloque original, para que en caso de que se traslapen los bloques, en el momento de reescribir en DIRINC, ya se haya movido su contenido a la nueva dirección (NDIRIN).

El movimiento de los bytes se efectúa byte por byte, llevando un conteo de las direcciones fuente y destino, de acuerdo con el diagrama de flujo.

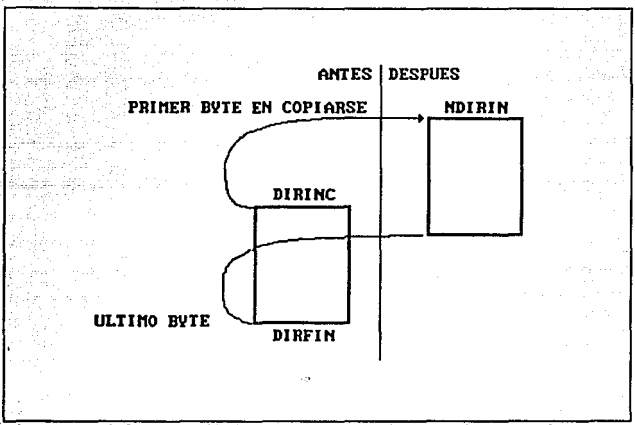


Diagrama del procedimiento para copiar un bloque de datos hacia arriba en el mapa de memoria.

En el caso de que el bloque de datos se deba desplazar hacia abajo en el mapa de memoria, el primer byte que se copia es el último del bloque, después el penúltimo y así sucesivamente, para evitar la pérdida de información en caso de que se traslapasen los bloques.

Cuando se trate de las opciones de copia de código desde o hacia memoria EEPROM, el procedimiento que se sigue es similar, con la única salvedad de que durante la escritura a la memoria EEPROM, se genera un retraso de alrededor de 10 ms antes de escribir otro dato, debido a que es el tiempo requerido para grabar el dato en esta memoria.

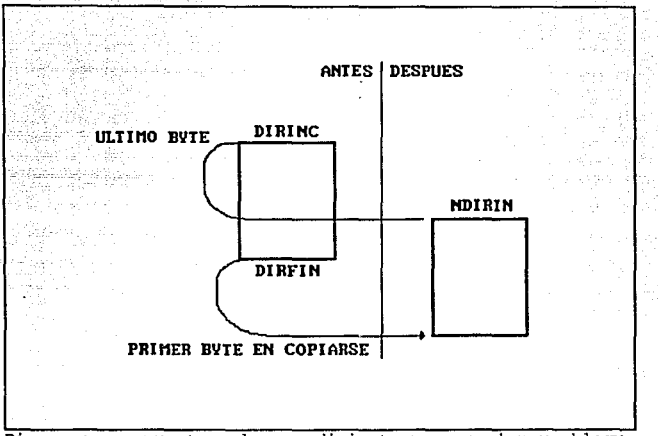


Diagrama que muestra el procedimiento para copiar un bloque de memoria hacia abajo.

La decisión de cual de las rutinas se debe emplear la efectúa el programa SD535 residente en la computadora PC, por lo que esta última deberá enviar el código de subrutina adecuado a la operación.

Una versión algo más simple de la que constituyen las subrutinas de copia de bloque, la forman las rutinas para llenado de bloques. Estas siguen un procedimiento similar pero con la ventaja de que el dato fuente se mantiene constante. Las rutinas de llenado de memoria de código y memoria de datos se encargan de recibir las direcciones inicial y final del bloque de memoria,

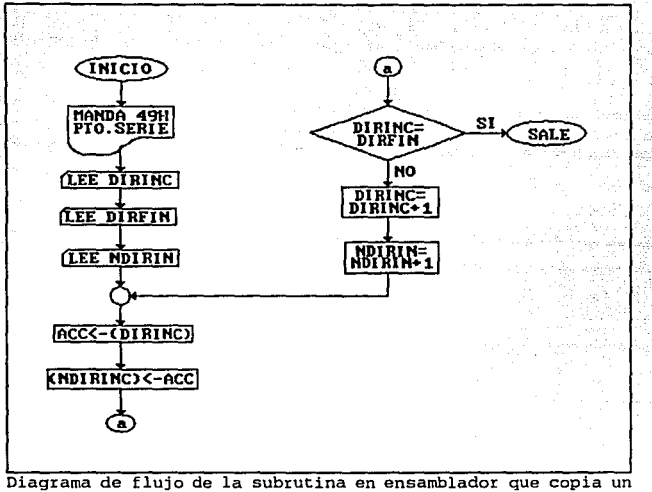


Diagrama de flujo de la subrutina en ensamblador que copia un bloque de datos hacia arriba en la memoria.

así como el dato (byte) que se quiere guardar en esas localidades. Una vez recibida esta información proceden a copiar el dato en tantas direcciones como hubiera sido indicado.

Las rutinas de edición de registros y memoria, siguen la misma estructura general de atención a una subrutina presentada en un diagrama de flujo anterior. Una vez que el programa residente en la PC coordina la actualización de los datos a nivel pantalla, se encarga de enviar el código de operación adecuado a la tarjeta del sistema de desarrollo y enviarle los datos que

se deban actualizar. En este mismo caso caen las rutinas de lectura y escritura de archivos a memoria, sólo que ahora además la PC deberá ser capaz de manejar los archivos en disco.

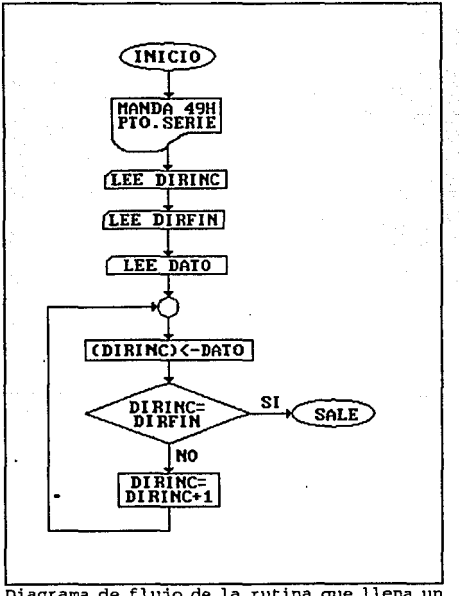


Diagrama de flujo de la rutina que llena un bloque de memoria con un dato.

Para la implementación de las rutinas de lectura de datos de memoria de código, memoria de datos y memoria RAM interna, se conservó el empleo de la subrutina de atención básica; la cual, como se graficó más arriba se basa únicamente en el intercambio de información. Para la lectura de datos se prefirió el uso de apuntadores, con los cuales, al ir incrementando una variable, pudiésemos señalar al siguiente dato. Como se podrá observar en el listado de las rutinas en ensamblador, esta idea no se conservó en el caso de la lectura de la memoria RAM interna en su parte baja (bytes 00H-7FH). Esto se debió a que entraba en conflicto con el STACK y los registros internos del usuario que se encontraban respaldados en éste. Por lo tanto, se tuvo que implementar la subrutina haciendo una lectura byte por byte de cada una de las localidades en esta área de la memoria del microcontrolador.

IV.3 EJECUCION DE UN PROGRAMA

La subrutina de ejecución se encarga de transferir el control de la tarjeta a un programa del usuario. Debido a que dentro del conjunto de instrucciones del microcontrolador no existe la posibilidad de hacer saltos largos a la dirección apuntada por una variable, fue necesario buscar otra alternativa. La solución que se empleó consiste en recibir la dirección inicial del programa del usuario y escribirla en una parte predefinida del área de código, junto con el código de

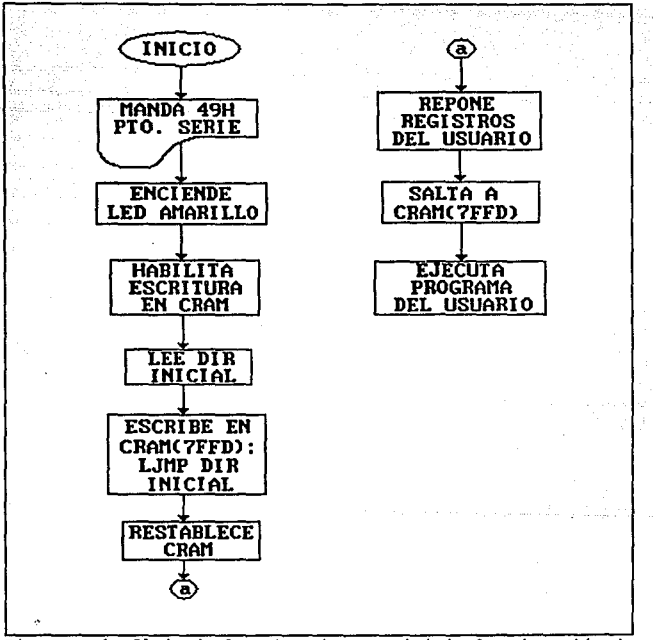


Diagrama de flujo de la subrutina que inicia la ejecución de un programa en ensamblador del usuario.

la instrucción LJMP (salto largo). Esto tiene la finalidad de construir la instrucción LJMP DIR en una localidad conocida del mapa de código (en este caso 7FFDH) para que al efectuar un

salto a esa localidad, el resultado final sea una transferencia del control a la dirección DIR enviada por el usuario. Antes de ejecutar esta instrucción se sacan de la pila los contenidos originales de los registros que usa el programa monitor para que el usuario pueda manipularlos.

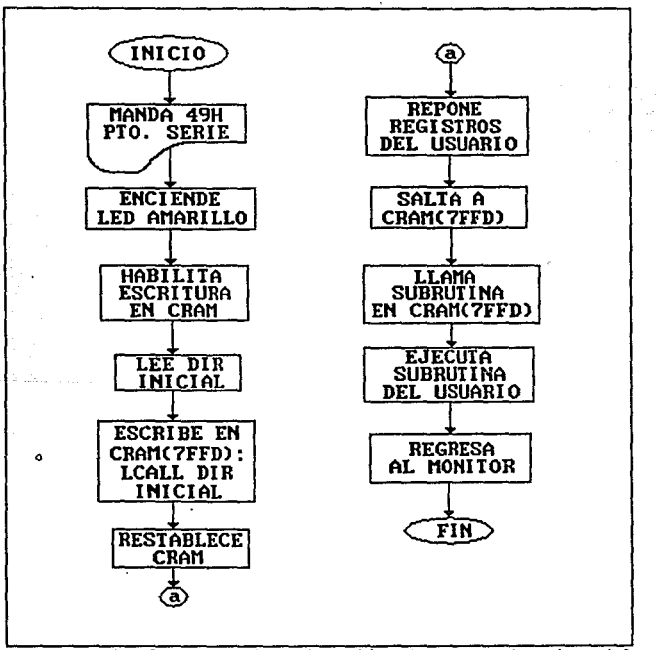


Diagrama de flujo de la ejecución de una subrutina del usuario.

La ejecución de una subrutina se activa de manera similar a la de ejecución de un programa del usuario con la diferencia de que la llamada se efectúa mediante la instrucción LCALL 7FFDH, en lugar de un salto directo LJMP 7FFDH, con la finalidad de que cuando se llegue a una instrucción RET, el programa monitor retome el control del sistema.

IV.4 MANEJO DE PAUSAS

En la rutina de inserción de pausas, el programa monitor se encarga de recibir las direcciones donde se desean los

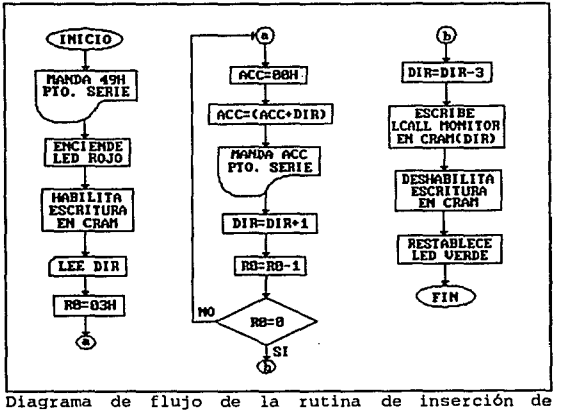


Diagrama de flujo de la rutina de inserción de pausas.

puntos de ruptura, insertar en su lugar un salto a una rutina de atención, y mandar a la PC el contenido original de las direcciones. Asumiendo que la dirección que indica el usuario es la del primer byte de una instrucción, y que la longitud máxima de una instrucción es de tres bytes, serán esos tres bytes los que se guarden enviándolos a la PC. Existía la posibilidad de almacenar esos tres bytes en el STACK (pila), o bien, en un área reservada de la memoria; sin embargo, se prefirió enviarlos a la PC con la idea de ocupar lo menos posible los recursos microcontrolador.

Aquí es importante hacer mención de que esta misma metodología no fue utilizada para guardar los registros internos del microcontrolador, debido a que su implementación era bastante complicada. De hecho se intentó desarrollar una rutina con ese fin, pero lo resultados no fueron satisfactorios.

La siguiente rutina que se implementó fue la de desactivación de las pausas o puntos de ruptura. Esta rutina se encarga de recibir las direcciones donde fueron insertadas las pausas y restablecer su contenido original con los datos que recibe de la PC.

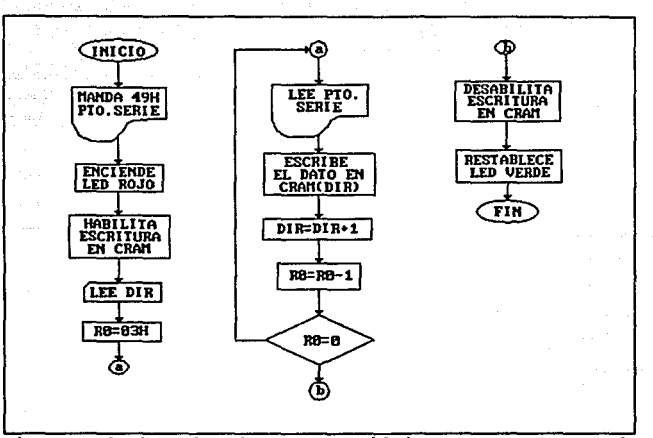


Diagrama de la subrutina en ensamblador que se encarga de eliminar las pausas.

IV.5 EJECUCION PASO A PASO

La ejecución paso a paso es una de las más importantes y más útiles. Para su implementación lo primero que se pensó fue en colocar una pausa en la dirección de la siguiente instrucción a ejecutar; sin embargo esta opción resulta muy complicada en el caso de que la instrucción a ejecutarse sea un salto condicional, ya que existen dos posibles ubicaciones de la siguiente instrucción, los cuales se deben identificar correctamente. El

mismo inconveniente resultaría en el caso de que se tomara la instrucción a ejecutar, se copiara en un área especial de la memoria y ahí se ejecutase. Para resolver estos inconvenientes se empleó un propiedad muy importante de las interrupciones del microcontrolador.

Una de las características de los microcontroladores de la familia MCS-51 de INTEL es que una vez que se activa una de las interrupciones, y se ejecuta su rutina de atención, si al regresar al flujo normal del programa permanece activa la interrupción, al menos se ejecutará una instrucción más. Es precisamente esta cualidad, la que nos permitió implementar la rutina de ejecución de un programa instrucción por instrucción (TRAZA). De esta manera, si mandamos ejecutar la instrucción que deseamos y mantenemos activa una interrupción, entonces sólo se ejecutará una instrucción y después el programa monitor puede retomar el control de la tarjeta, por medio de la subrutina de atención a la interrupción.

La rutina principal de ejecución paso a paso se encarga de recibir la dirección de la instrucción a ejecutarse DIR, y escribir en un área de memoria reservada (7FFDH) el código de un salto incondicional hacia esa localidad (LJMP DIR). Al mismo tiempo restablece los registros internos del microcontrolador con los datos del usuario y activa la interrupción serial. Previamente se ha ajustado el vector de interrupción serial para

que éste apunte a una rutina de atención específica situada en la dirección 9200H. Una vez activada la interrupción, el programa sigue una serie de pasos que son controlados por la rutina de atención y que culminan con la ejecución de la instrucción en la localidad indicada.

Para implementar esta rutina de ejecución paso a paso se hubiera podido utilizar cualquier fuente de interrupción, incluso una interrupción externa. Sin embargo, la opción más adecuada era la utilización de la interrupción serial, ya que de antemano el control de este puerto estaba asignado como exclusivo para el programa monitor del sistema de desarrollo, debido a su empleo constante como vía de intercambio de información y comandos con la computadora central (PC). Además si se hubiera empleado una interrupción externa, se hubiera sacrificado otro vector de interrupción y se hubiera tenido que implementar alguna circuitería adicional.

La rutina de atención se encarga de monitorear en que momento se ha ejecutado la instrucción deseada y de recuperar el control de la tarjeta para el programa monitor. Una vez que el programa activa la interrupción serial cada vez que se ejecuta una instrucción se transfiere el control a la rutina de atención. Esta desactiva momentáneamente las interrupciones y evalúa de común acuerdo con la PC si se ha ejecutado la instrucción de la dirección indicada. De ocurrir así, se efectúa una serie de

operaciones con el STACK, que permiten determinar cuál es la dirección de la siguiente instrucción a ejecutarse, de lo contrario se rehabilita la interrupción serial y se continua el procedimiento.

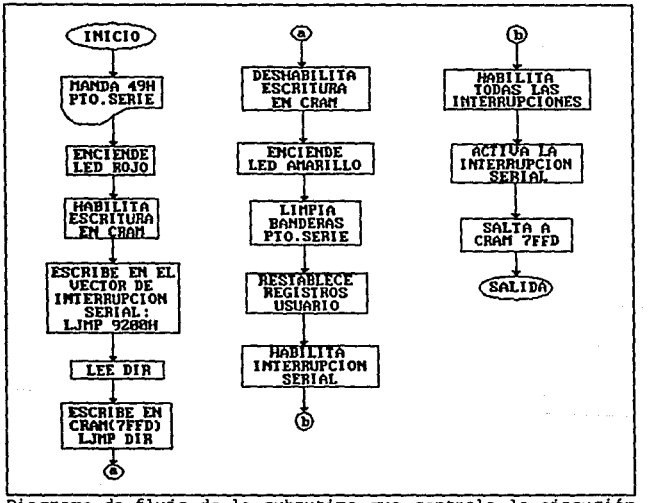


Diagrama de flujo de la subrutina que controla la ejecución de una instrucción en ensamblador.

Es importante que una vez ejecutada la instrucción indicada, se obtenga la dirección de la siguiente, para permitir que este

procedimiento de ejecución pueda ser repetitivo de una manera automática. De esta forma, si el usuario desea ejecutar la siguiente instrucción, no se verá en la necesidad de proporcionar la dirección de ésta.

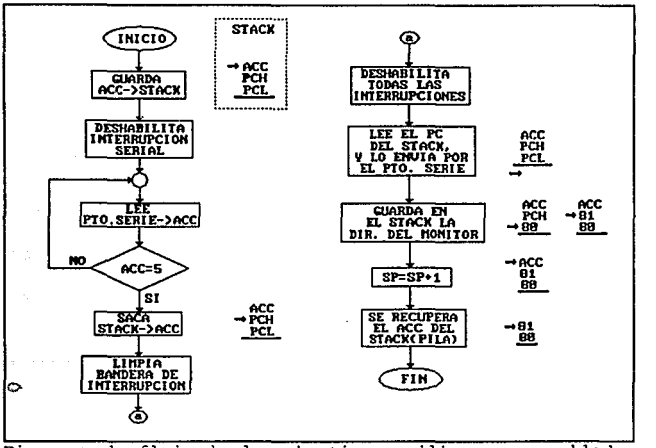


Diagrama de flujo de la subrutina auxiliar en ensamblador para la ejecución paso a paso (interrupción serial).

Una vez que se ejecutó la instrucción solicitada, el programa SD535 residente en la PC, se encarga de solicitarle una actualización de todos los datos presentes en la pantalla, permitiendo una visualización inmediata del resultado de la operación.

La posibilidad de ejecución paso a paso (instrucción por instrucción) constituye una de las herramientas más importantes en la depuración de programas. Aunada a las posibilidades de visualización de datos, almacenamiento de programas y la corrección de los mismos que permite el ambiente integrado del sistema de desarrollo SD535, podemos decir que proporciona una herramienta muy versátil para la elaboración de sistemas basados en los microcontroladores de la familia 8051 y 80535.

CONCLUSIONES

La implementación del sistema de desarrollo SD535 cristalizó con el cumplimiento de los objetivos que se habían planteado. Se logró crear una herramienta poderosa y fácil de usar. Al estar controlado desde una computadora PC, se tiene la posibilidad de desplegar una gran cantidad de información en su pantalla. De hecho, simultáneamente tenemos la visualización de todos los registros internos del microcontrolador, de 128 bytes de memoria y de los estados de algunas funciones del sistema de desarrollo. Esta capacidad de despliegue, aunada a la facilidad de almacenamiento de información que proporcionan los archivos en disco, justifican plenamente el no haber elaborado un sistema autónomo.

Dentro de las ventajas de un sistema controlado por una computadora PC, se presenta la posibilidad de la edición y el ensamblado de programas dentro del ambiente del SD535. Es decir, se logra *integrar* todas las etapas de desarrollo de un programa: edición, ensamblado, y ejecución.

Para facilitar la tarea de depuración se cuentan con varias alternativas, de las cuales la ejecución paso a paso resulta ser invaluable en la prueba de programas complejos.

El ambiente basado en ventanas posibilita un manejo rápido e intuitivo de las funciones del sistema. Los periféricos que se incorporaron en la tarjeta, sumado a los que incluye el microcontrolador, le proporcionan una gran versatilidad. De esta manera, el sistema de desarrollo SD535 encuentra cabida junto al ingeniero de diseño y al mismo tiempo se presenta como un apoyo didáctico en la enseñanza.

Desde luego, cualquier diseño es susceptible de mejorarse, y como tal lo es el presente trabajo. Siempre se encontrarán nuevas alternativas y posibilidades, que nacen de la interacción con el problema y de las experiencias que se producen; sin embargo, es conveniente tener en mente cuales son los objetivos, para no caer en el error de efectuar una pequeña mejoría a cambio de una gran inversión de tiempo y esfuerzo.

A pesar de los errores que pueda tener, este proyecto me ha dejado satisfecho, porque me ha permitido entender un poco mejor las posibilidades de conjuntar la electrónica con la computación, en la realización de proyectos.

BIBLIOGRAFIA

Analog Devices."Data acquisition databook". Analog Devices Inc, 1980.

Bursky, Dave."Microcontroller ICs offer many on-chip features". Electronic Design. August 9,1990.

Dailey, Denton J."Operational amplifiers and linear integrated circuits: theory and applications". McGraw-Hill Book Company, 1989.

Fletcher, William J."An engineering aproach to digital design". Prentice-Hall Inc.,1989.

Fujitsu."MOS memory products". Fujitsu Microelectronics Inc.,1989.

Gillings, Brian."IC support for the RS-232-C and its derivatives". Electronic Engineering. October, 1986.

Horowitz, Paul, and Hill, Winfield."The art of electronics". Second edition. Cambridge University Press, 1989.

Intel."Microsystem componentes handbook. Peripherals Vol. I and Vol. II".Intel Corp.,1989.

Ibid."Embedded control applications handbook". Intel Corp., 1989.

Ibid."8-bit embedded controllers handbook". Intel Corp., 1989.

Motorola."Linear and interface ICs". Motorola Inc., 1980.

Ibid."M68HC11EVM evaluation module user's manual". Motorola Inc, 1989.

National."Linear databook Vols. I and II".Rev. 1. National Semiconductor Corp.

Ibid."LS/S/TTL logic databook". National Semiconductor Corp., 1987.

Ibid."Programmable logic devices databook". National Semiconductor Corp., 1990.

Newburn, Frank."Applying micro-programmable controllers". Hmt Engineering. August 27, 1987.

O'Brien, Stephen K."Turbo Pascal 5.5 the complete reference". McGraw-Hill, 1989.

Peatman, John B." Design with microcontrollers". McGraw-Hill Bk Company, 1988.

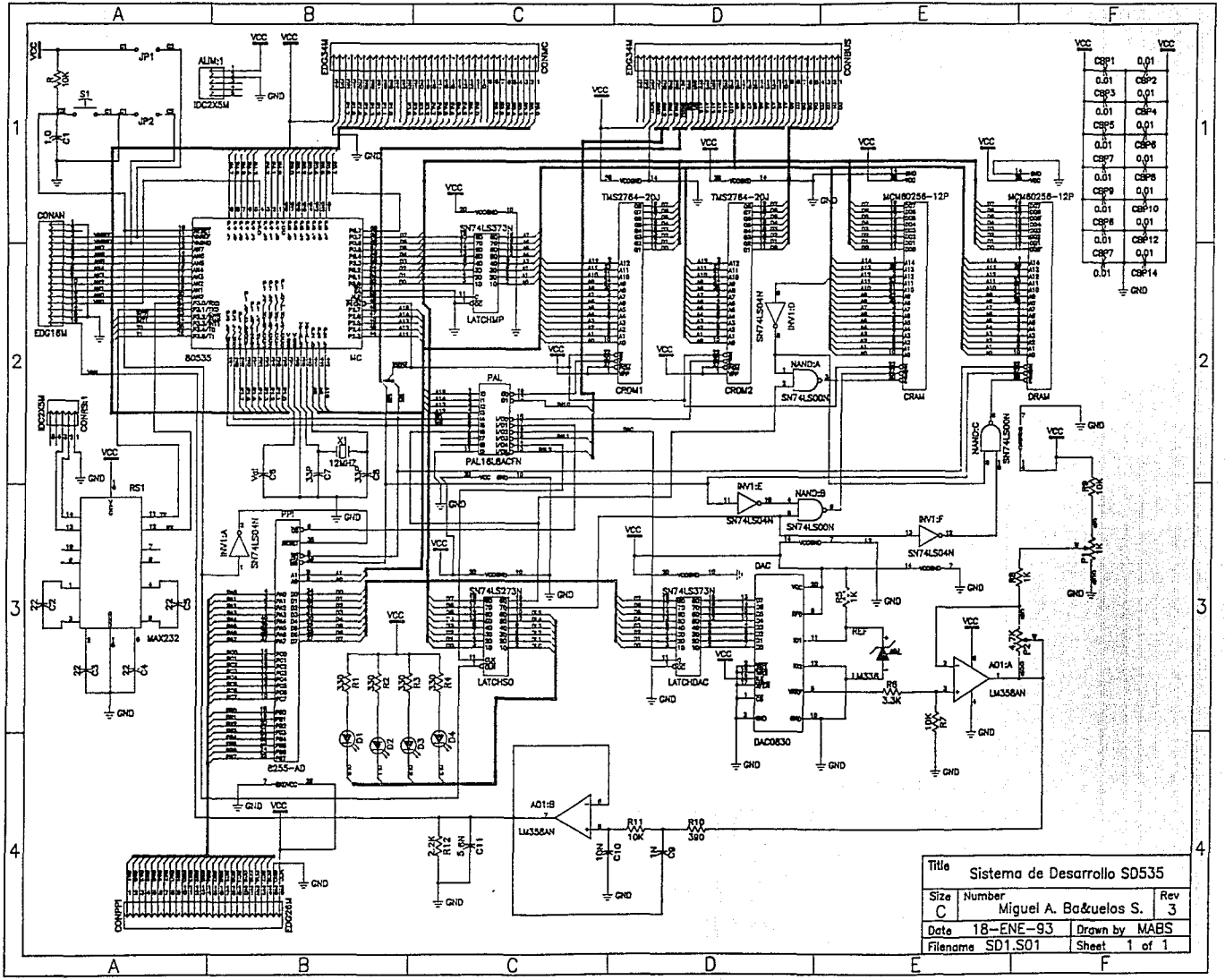
Siemens."Microcontrollers, microprocesors, peripherals and memory data book 1988/89".Siemens Components U.S.A.

Texas Instruments."TMS370 configurable µCs product bulletin". Texas Instruments Inc.

Vaglica, John J., and Gilmour, Peter S."How to select a microcontroller". IEEE Spectrum. November 1990.

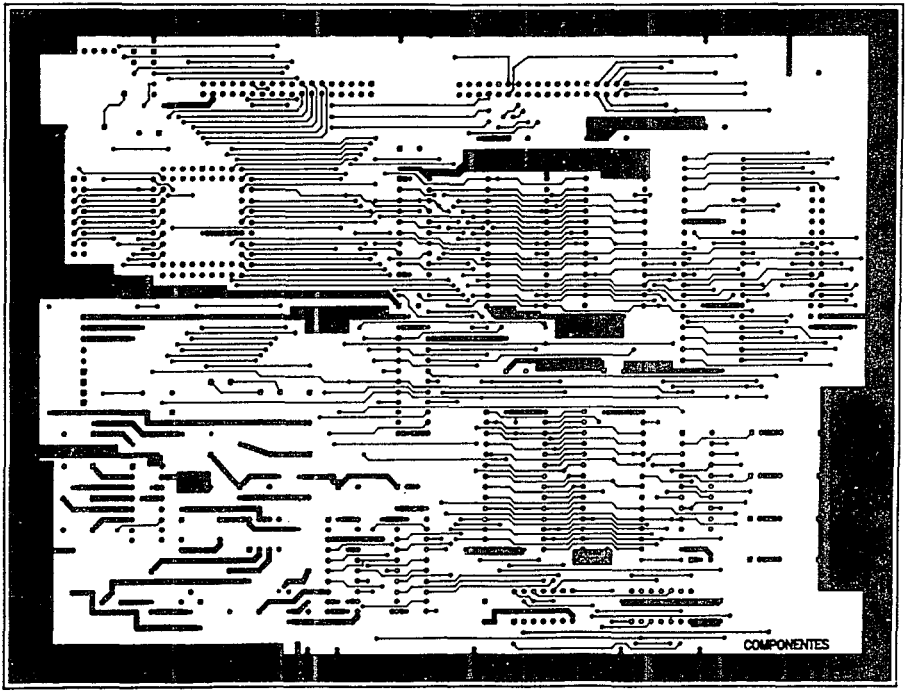
APENDICE A.

DIAGRAMA ESQUEMATICO.

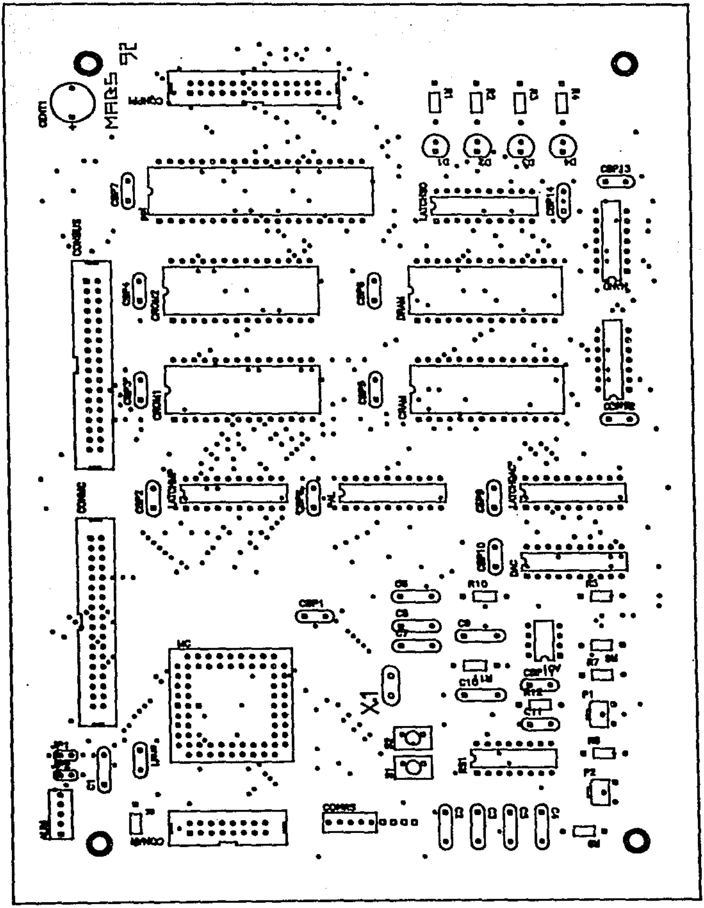

Title
Sistema de Desarrollo SD535
Size
C
Number
Miguel A. Ba&uelos S.
Rev
3
Date
18-ENE-93
Drawn by
MABS
Filename
SD1.S01
Sheet
1 of 1

APENDICE B.

MASCARILLAS DEL CIRCUITO IMPRESO.


COMPONENTES

SOLDADURA



CBP13
R10
CBP1
C9
C10
R12
C11
P1
P2
R7
MC
X1

APENDICE C. DETALLE DE CONEXIONES

Conector CONMC (conector al microcontrolador).

PIN	FUNCION
1	P5.0
2	P5.1
3	P5.2
4	P5.3
5	P5.4
6	P5.5
7	P5.6
8	P5.7
9	sin conexión
10	sin conexión
11	P4.0
12	P4.1
13	P4.2
14	P4.3
15	P4.4
16	P4.5
17	P4.6
18	P4.7
19	P3.2
20	P3.3
21	P3.4
22	P3.5
23	sin conexión
24	sin conexión
25	P1.0
26	P1.1
27	P1.2
28	P1.3
29	P1.4
30	P1.5
31	P1.6
32	P1.7
33	GND
34	VCC

Conector CONBUS (conector del bus).

PIN	FUNCION
1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	A0
10	A1
11	A2
12	A3
13	A4
14	A5
15	A6
16	A7
17	A8
18	A9
19	A10
20	A11
21	A12
22	A13
23	A14
24	A15
25	-RD
26	-WR
27	-PSEN
28	PAL0
29	PAL1
30	PAL2
31	GND
32	VCC
33	sin conexión
34	sin conexión

Conector CONPPI (conector del puerto paralelo).

PIN	FUNCION
1	PA1
2	PA0
3	PA3
4	PA2
5	PA5
6	PA4
7	PA7
8	PA6
9	PB1
10	PB0
11	PB3
12	PB2
13	PB5
14	PB4
15	PB7
16	PB6
17	PC1
18	PC0
19	PC3
20	PC2
21	PC5
22	PC4
23	PC7
24	PC6
25	GND
26	VCC

Conector CONAN (conector de E/S analógica).

PIN	FUNCION
1	GND
2	sin conexión
3	VAREF
4	VAGND
5	AN7 (entradas al ADC del 80535)
6	AN6
7	AN5
8	AN4
9	AN3
10	AN2
11	AN1
12	AN0
13	VPD
14	sin conexión
15	GND
16	VAN (salida del DAC)

Conector CONRS (conector RS232)

PIN	FUNCION
1	GND
2	TX
3	RX
4	sin conexión
5	sin conexión

Conector ALIM (alimentación)

PIN	FUNCION
1	VCC
2	GND
3	GND
4	GND
5	sin conexión

APENDICE D.

LISTA DE COMPONENTES.

Cantidad	Tipo
1	Microcontrolador SAB80535
1	Memoria UVEPROM 8Kx8 TMS27C64
1	Memoria EEPROM 8Kx8 2864
2	Memorias SRAM 32Kx8 60256
2	Latch octal 74LS373
1	Latch octal 74LS273
1	PAL TIBPAL16L8
1	Interface periférica programable PPI 8255
1	Manejador de interface serial RS-232C MAX232
1	Inversor lógico séxtuple 74LS04
1	Compuerta lógica NAND 74LS00
1	Convertidor D/A 8 bits DAC0830
1	Referencia de voltaje LM336-2.5Z
1	Amplificador operacional Dual LM358N
4	Resistencias de 330 Ω a 1/2W
1	Resistencia de 390 Ω a 1/2W

Cantidad	Tipo
2	Resistencias de 1K a 1/2W
1	Resistencia de 2.2K a 1/2W
1	Resistencia de 3.3K a 1/2W
4	Resistencias de 10K a 1/2W
1	Minipreset de 1K
1	Minipreset de 10K
3	LEDs
2	Capacitores de 33 pF
2	Capacitores de 1 nF
1	Capacitor de 5.6 nF
1	Capacitor de 10 nF
14	Capacitores de 0.1 µF
1	Capacitor de 1 µF
4	Capacitores de 22 µF
6	Filas de postes para conexión
2	Conectores EIS macho de 5 pines
2	Jack banana
1	Botón de presión
3	Conectores DB9
2	Conchas para conector DB9
2	mts. de cable coaxial estéreo

Cantidad	Tipo
1	Circuito impreso
1	Base para circuito impreso

APENDICE E. LISTADO DE PROGRAMAS

El listado completo de los programas del SD535 se incluye en un diskette anexo al presente trabajo.

Los archivos incluidos son:

ARCHIVO	DESCRIPCION
SD535.PAS	Listado del programa en Pascal
SD535.ASM	Listado del programa en Ensamblador
SD535.LST	Listado del programa ensamblado junto con el código fuente
SD535.HEX	Listado en formato HEX del programa ensamblado
SD535.OBJ	Listado en formato OBJ del programa ensamblado
SD535.EXE	Programa ejecutable para computadora IBM PC o compatible