ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS MICROCONTROLADORES DE LA FAMILIA MCS-51/52.

TESIS PREVIA A LA OBTENCION DEL TITULO DE
INGENIERO ELECTRONICO EN LA ESPECIALIZACION
DE ELECTRONICA Y TELECOMUNICACIONES

WASHINGTON GIOVANNI SIDEL TORRES

AGOSTO 1993

CERTIFICACION:

CERTIFICO QUE EL PRESENTE TRABAJO FUE DESARROLLADO INTEGRAMENTE POR EL SEÑOR:

WASHIGTON GIOVANNI SIDEL TORRES

ING. JAIME VELARDE
DIRECTOR DE TESIS

DEDICATORIA:

A MI MADRE
A MI HIJO ERIK JOEL

WASHINGTON GIOVANNI

AGRADECIMIENTO:

MI AGRADECIMIENTO AL ING. JAIME VELARDE POR SU COLABORACION EN EL DESARROLLO DE ESTE TRABAJO.
UN RECONOCIMIENTO ESPECIAL A TODAS LAS PERSONAS QUE DE UNA U OTRA MANERA COLABORARON DESINTERESADAMENTE EN SU REALIZACION.
AL ING. BOLIVAR LEDESMA
A MI PADRE POR SU ESFUERZO Y APOYO CONSTANTE.

WASHINGTON GIOVANNI

RESUMEN

El objetivo del presente trabajo es el de desarrollar un circuito basado en uno de los microcontroladores de la familia MCS-51/52 que sirva como apoyo a todos aquellos diseñadores de hardware y software que utilicen uno de estos microcontroladores.

El módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51; como se denominó al presente trabajo consta de una serie de ayudas al usuario que le permiten ejecutar y optimizar un programa de manera rápida y eficiente.

Adicionalmente en el módulo de desarrollo se contempla la posibilidad de tener entrada o salida de datos hacia o desde el módulo; para tener la posibilidad de interconección con otros sistemas.

El programa de control del módulo se realiza en el paquete de software SIDES-i51, por ser este, el utilizado con mayor frecuencia para el desarrollo de programas para los microcontroladores de ésta y otras familias.

La comunicación serial que ofrece el programa del módulo utiliza el formato INTEL de ocho bits presente también en el SIDES-i51.

INDICE

CAPITULO I

GENERALIDADES

1.1 - Introducción.

1.2 - Justificación del presente trabajo.

1.3 - Análisis de la familia de los Microcontroladores MCS-51.

1.4 - Elementos básicos para el funcionamiento de la familia MCS_51.

1.1 INTRODUCCION

Dentro del perfil profesional del Ingeniero Electrónico actualmente es indispensable el conocimiento y manipulación de los diferentes tipos de microprocesadores y microcontroladores, y más aún cuando estamos viviendo en una era en la cual se está tendiendo a modernizar obsoletos sistemas de procesamiento y control electromecánicos, electroneumáticos y electrónicos usando éste tipo de elementos. Debido a la gran versatilidad que presentan para adaptarse a uno u otro sistema y la confiabilidad que brindan en el trabajo con extrema precisión, sin dejar de mencionar el relativo bajo costo a mediano y largo plazo que éstos elementos tienen.

Es así que ahora no resulta nada difícil encontrar éstos elementos en casi cualquier artefacto, desde "simples" equipos caseros hasta complicados sistemas industriales tienen en su interior algún tipo de microprocesador o microcontrolador que está regulando determinada función del equipo.

Por esta razón el presente trabajo de Tesis trata de brindar un soporte teórico y práctico para aquellos usuarios que estén pensando desarrollar un sistema basado en los microcontroladores de la familia MCS-51.

1.2 JUSTIFICACION DEL PRESENTE TRABAJO

La Facultad de Ingeniería Eléctrica consiente de los cambios actuales y las necesidades de sus estudiantes, ofrece la materia de Sistemas Microprocesados, en la cual se da al estudiante un conocimiento básico acerca del uso y funcionamiento de los microprocesadores y microcontroladores; tomándose como ejemplo para una mejor comprensión la familia de microcontroladores MCS-51/52 de la casa INTEL.

En el transcurso del semestre el estudiante aprende el principio de funcionamiento que tienen éstos elementos, como y de que manera se los puede usar en un sistema dado, de que manera se los puede programar, cual es el conjunto de instrucciones que tienen y cuales son las operaciones que se realizan dentro del microprocesador o microcontrolador; y, un aspecto muy importante de la materia es que el estudiante aprenda a proyectar y construir su propio sistema de acuerdo a una situación específica de trabajo.

Para reforzar la materia que en gran parte es teórica la Facultad ofrece como complemento las prácticas de Sistemas Microprocesados, en las cuales, mediante el uso de un computador personal y el paquete de Software denominado SIDES en su versión i-51, el estudiante se adentra en el campo de la programación de los microcontroladores, y es así que aquí se comienzan a desarrollar pequeños programas

de aplicación y ejemplo para adiestrarse en el uso y la gran posibilidad de desarrollo de éstos elementos.

Como éste proceso de aprendizaje se realiza dentro del ambiente restringido del computador y sujeto a las limitaciones del programa controlador SIDES-i51, el estudiante comienza a formarse una falsa idea de lo que es la programación de un microcontrolador ya que como esta se la realiza totalmente en la pantalla del computador, donde se ensambla y se simula, se pierde la objetividad de ver trabajar el programa realizado en un circuito real, y lo que es más se pierde totalmente la noción del tiempo de ejecución (TIEMPO REAL) del programa. Otra desventaja de trabajar únicamente de ésta manera es que como el programa Sides-i51 cuenta con una serie de ayudas y macroinstrucciones predefinidas, existen varias aplicaciones en las cuales no se obtienen los mismos resultados cuando se ejecuta el programa en el computador que cuando se ejecuta el programa en un circuito real.

Analizando que ésta manera de enseñanza es insuficiente se ha diseñado y construido el presente trabajo con el afán de completar el programa de estudio acerca de los microcontroladores, y por otra parte la filosofía del presente trabajo es la de apoyar permanentemente a todos aquellos diseñadores de hardware y software que utilicen cualquier tipo de microcontroladores de ésta familia en especial.

La idea básica del uso del Módulo de Desarrollo es la siguiente:

Cuando se tiene un programa desarrollado mediante el Sides-i51, y una vez que dicho programa se encuentre probado y ensamblado en el computador, mediante la opción de grabación de EPROMS con la que cuenta dicho paquete se descarga por el pórtico serial del computador todos los códigos de máquina del programa de prueba hacia el módulo de desarrollo, en el cual se almacenarán; una vez que se tiene dichos códigos almacenados en el módulo de desarrollo se esta en la capacidad de:

- Observar el contenido de cualquier localidad de memoria o registro.
- Cambiar el contenido de cualquier localidad de memoria o registro.
- Llenar un bloque de localidades de memoria con un determinado valor a elegir.
- Insertar o borrar una localidad de memoria.
- Colocar y remover puntos de parada dentro del programa.
- Ejecutar el programa dentro del mismo módulo (ejecución en TIEMPO REAL).
- Utilizar subrutinas de uso general presentes en el programa del módulo de desarrollo.
- Adicionalmente el módulo desarrollado cuenta con varias líneas de habilitación para utilizar elementos de entrada/salida de datos externos.
- Monitoréo de cualquier señal del microcontrolador o líneas especiales de habilitación.

1.3 ANALISIS DE LA FAMILIA DE LOS MICROCONTROLADORES MCS-51/52.

La familia de los microcontroladores MCS-51/52 consta de una serie de elementos los cuales se encuentran resumidos en la Tabla 1.1.

Todos los elementos de esta familia constan del mismo conjunto de instrucciones, y de acuerdo al elemento se tiene una u otra característica adicional la cual se la puede observar en la tabla, por ejemplo si se escoge el elemento 8052AH, se tiene una versión sin ROM interna en el 8032AH, una versión con EPROM interna en el 8752BH, el tamaño de memoria es de 8K, se cuenta con 256 localidades de RAM, 4 pórticos de entrada/salida de 8 bits, 3 temporizadores/contadores, 8 interrupciones procedentes de 6 fuentes, y no consta con la característica de power down. El MCS-8051 es el miembro típico dentro de la familia de los microcontroladores de 8-bits MCS-51/52, por lo tanto éste es el patrón para todos sus miembros.

Las características fundamentales del 8051 son:

- CPU de 8-bits optimizado para aplicaciones de control y transmisión fullduplex de datos.
- Capacidad de procesamiento Booleano (Lógica bit a bit).
- Espacio direccionable de 64K bytes para memoria de programa.
- Espacio direccionable de 64K bytes para memoria de datos.

Elemento	Versión sin ROM int.	Versión con EPROM int.	Tamaño de ROM	Memoria RAM int.	Pórticos de 8-bits I/O	Timers/Count 16-bits	Canales DMA	Int./fuente Interrupc.	Power Down
8051	8031		4K	128	4	2		'6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		'6/5	
8052AH	8032AH	8752BH	8K	256	4	3		'8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		'6/5	SI
83C51FA	80C51FA	87C51FA	8K	256	4	3		'14/7	SI
83C51FB	80C51FA	87C51FB	16K	256	4	3		'14/7	SI
83C51GA	80C51GA	87C51GA	4K	128	4	2		'8/7	SI
83C152JA	80C152JA		8K	256	5	2	2	'19/11	SI
	80C152JB			256	7	2	2	'19/11	SI
83C152JC	80C152JC		8K	256	5	2	2	'19/11	SI
	80C152JD			256	7	2	2	'19/11	SI
83C451	80C451		4K	128	7	2		'6/5	SI
83C452	80C452	87C452P	4K	256	5	2		'9/8	SI

TABLA 1.1

WASHINGTON GIOVANNI SIXEL TORRES

- Memoria de programa de 4K bytes en el mismo chip.
- 128 bytes de memoria de datos en el mismo chip.
- 32 líneas bidireccionales de I/O direccionables individualmente.
- 2 Temporizadores/Contadores.
- Canal full duplex de transmisión Serial
- Una estructura de interrupciones de 6 fuentes/5 vectores, con dos niveles de prioridades
- Oscilador construido en el mismo chip.

En éste capítulo no se intenta transcribir o analizar detalles que se podrían encontrar en cualquier manual de estos microcontroladores, únicamente se exponen breves consideraciones que se deben tener en cuenta para el diseño de un hardware utilizando algún elemento de esta familia.

Cuando se va a realizar el diseño de un hardware, sea cual sea la finalidad en un principio siempre se tratará de escoger los elementos que sean lo más cómodos para trabajar y los que más ventajas ofrecen; es así que en un principio la elección más obvia podría ser elegir el microcontrolador MCS-8751 en el cuál se tiene EPROM incluida dentro del mismo chip, lo cual ahorraría tener que estar haciendo una circuitería externa para la ROM. Pero, si el programa a realizarse es extenso, y particularmente si tiene varios elementos a los cuales va a controlar; posiblemente durante la etapa de prueba y optimización del programa se tenga que hacer varias adecuaciones del mismo o en su defecto borrar, o aumentar

ciertas instrucciones dentro del programa; ésto implica que continuamente se va a tener que borrar y así mismo grabar nuevos datos dentro de la memoria del microcontrolador, ahora, puede ser que éstas continuas borradas y grabadas dañen al chip, y tomando en cuenta que el costo del microcontrolador en nuestro medio es elevado, aparte que es difícil de conseguir; talvez convenga en la etapa de desarrollo del programa usar una versión del microcontrolador que no tenga memoria incluida, sino que usa ROM externa al chip, las cuales tienen menor costo que el microcontrolador con EPROM incorporada, por lo tanto será menos crítico reemplazar una EPROM que reemplazar todo el microcontrolador.

Por éste motivo en el presente caso se elige trabajar con el microcontrolador MCS-8031.

Otro aspecto que siempre se debe considerar es la posibilidad de tener suficiente espacio para almacenamiento de datos, sí así lo requiere el hardware que se esta diseñando, es decir se deberá incluir algún tipo de RAM, que dependiendo del volumen de datos a manipular se debe considerar un adecuado margen para dicho almacenamiento. Sin tomar en cuenta que dentro del mismo microcontrolador, y dependiendo del elemento elegido se cuenta con 128 o 256 localidades de RAM, que son de fácil y rápido acceso.

1.4 ELEMENTOS BASICOS PARA EL FUNCIONAMIENTO DE LA FAMILIA MCS-51

Todos los elementos de la familia MCS-51 tienen espacios de direccionamiento separados para accesar a la Memoria de Programa o a la Memoria de Datos, esta separación entre los dos tipos de memoria, permite que la RAM interna pueda ser accesada por un direccionamiento de 8 bits con lo cual se logra un almacenamiento y manipulación mas rápida por parte de la CPU, sin embargo un direccionamiento de 16 bits puede utilizarse para RAM o ROM externa, generalmente a través del registro DPTR (Puntero de datos).

A la memoria de programa externa la familia MCS-51, puede accesar hasta 64 Kbytes, algunos miembros de esta familia poseen adicionalmente 4k, 8k, o 16 kbytes dentro del mismo chip (Tabla 1.1).

Los elementos de esta familia presentan una distribución de pines como la indicada en la figura 1.1.

En donde:

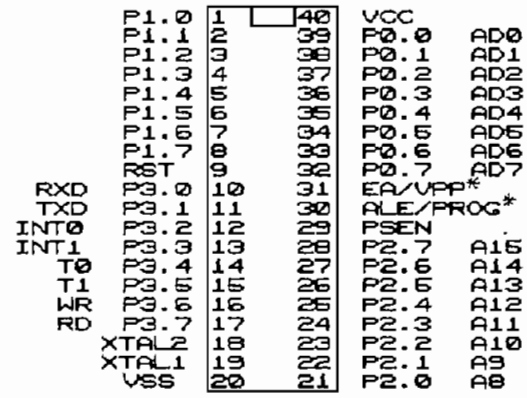
VCC: Voltaje de alimentación 5V DC.

VSS: Referencia o tierra del circuito.

$\overline{\text{PSEN}}$: (Program store enable), petición de lectura de memoria de programa externa. Mediante esta señal activa en bajo se logra el acceso a memoria externa de programa, esta señal no se activa cuando se ejecuta el programa desde ROM interna.

$\overline{\text{RD}}$, $\overline{\text{WR}}$: (Read, Write), señales que se activan cuando se

realiza el acceso a memoria de datos externa ya sea para lectura y escritura respectivamente.



* *Para los elementos con EPROM interna.*

Figura 1.1: Distribución de pines de los microcontroladores MCS-51/52.

$\overline{EA}$: Para el caso de los elementos en los cuales tenemos EPROM interna y externa; una u otra puede ser accesada mediante la entrada $\overline{EA}$ (External access), colocando ésta a VCC o a VSS respectivamente.

PORTICO 0:(P0.0,,P0.7),pórtico bidireccional de salidas con drenaje abierto para entrada/salida de datos. Una característica de ésta familia, es la particularidad que en el pórtico P0, se encuentran multiplexados la parte baja del bus de direcciones con el bus de datos; es decir que por éste pórtico en cualquier petición de acceso a memoria externa en un primer instante se dedica a los ocho bits menos significativos del bus de direcciones, el cual se completa con los ocho bits mas significativos del pórtico P2, para luego dedicarlo a la entrada o salida de datos (Bus de datos).

ALE: (Address latch enable), señal mediante la cual el

microcontrolador indica que en ese momento se encuentra la parte baja del bus de direcciones en el pórtico cero, esta generalmente se usa como señal de habilitación a un Latch externo de ocho bits con lo cual se retiene esta parte del bus de direcciones.

PORTICO 2: En este pórtico bidireccional de entrada/salida se emite la parte alta del bus de direcciones.

Un diagrama de tiempos del comportamiento de estas señales se encuentra en la figura 1.2, para un ciclo de lectura de memoria externa de programa.

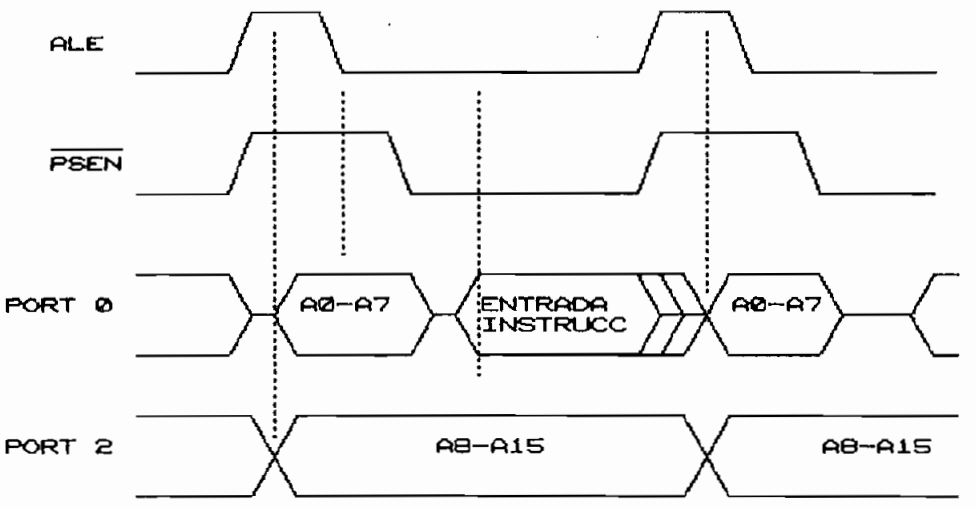


Figura 1.2: Diagrama de tiempos del acceso a memoria de programa externa.

Nótese que cuando el pórtico cero tiene la parte baja del bus de direcciones, la señal ALE es activada; luego se activa la señal $\overline{\text{PSEN}}$ para la lectura del código correspondiente, mientras el pórtico dos conserva la parte superior del bus de direcciones.

El circuito que se debe implementar para obtener acceso ROM externa se lo tiene en la figura 1.3.

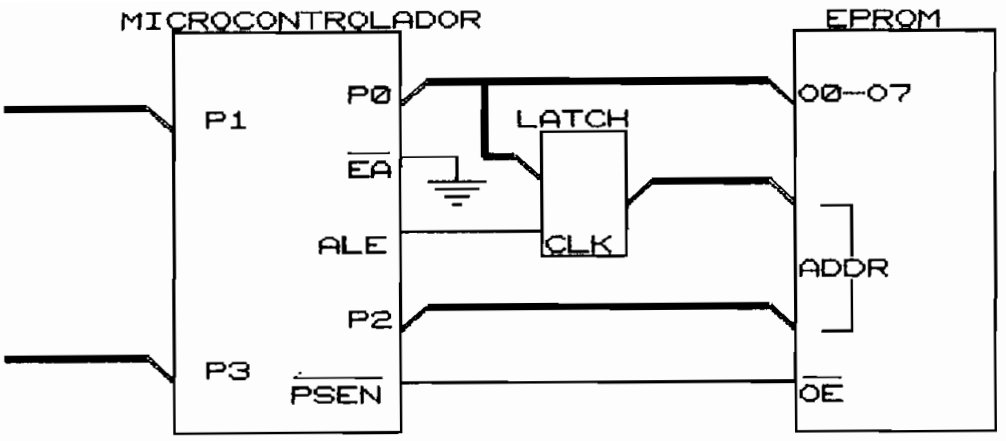


Figura 1.3: Hardware acceso a memoria de programa externa.

Para el caso de un acceso a RAM externa, ésta se logra mediante las señales $\overline{RD}$ y $\overline{WR}$, manteniendo el mismo esquema de direccionamiento.

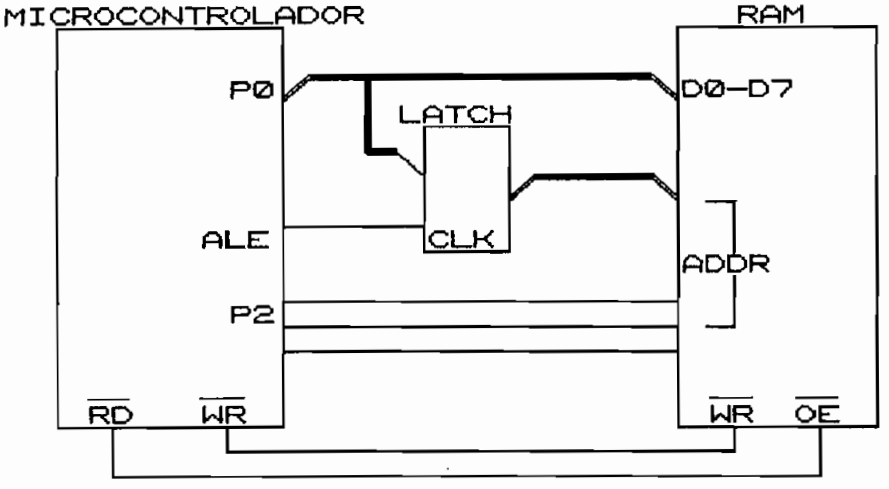


Figura 1.4: Hardware acceso a memoria de datos externa.

Se puede tener la necesidad, en cualquier aplicación de implementar memoria de programa y memoria de datos a la vez; la manera de lograr ésto es generalmente usando algún elemento decodificador de direcciones para tener acceso a

una u otra memoria de acuerdo a un direccionamiento dado.

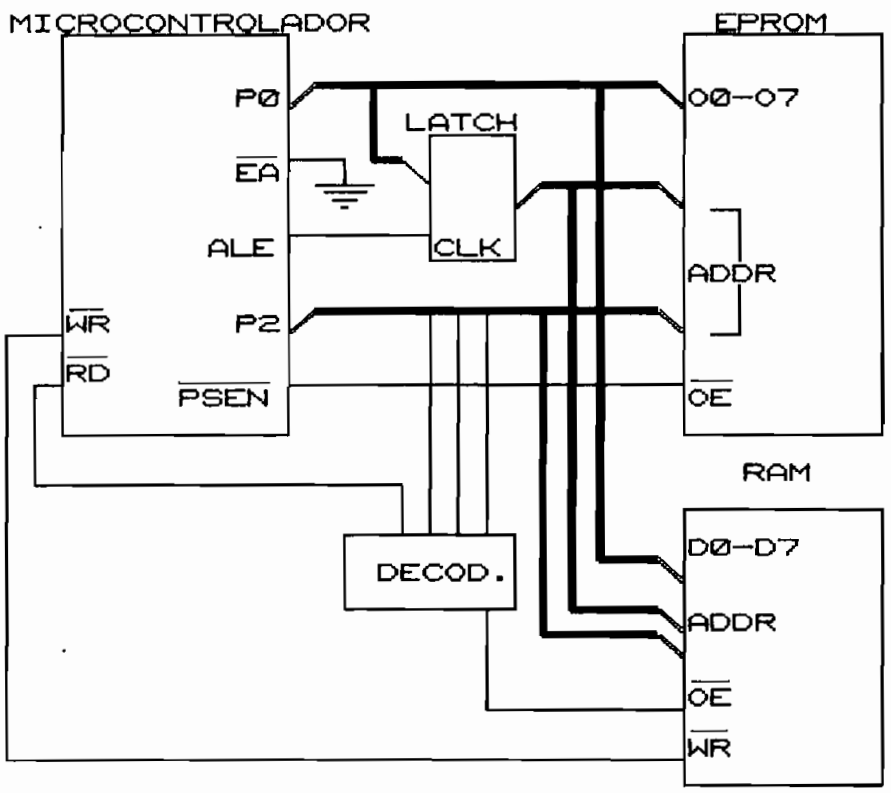


Figura 1.5: Hardware del acceso a memoria externa de programa y de datos.

Como elemento decodificador de direcciones se usa un 74LS138 (Decodificador 3 a 8), que es uno de los más versátiles para éste propósito.

CAPITULO II

HARDWARE

2.1 - Análisis del hardware de acuerdo a la necesidad del usuario

2.2 - Pórticos de Entrada/Salida de datos.

2.3 - Hardware para comunicaciones.

2.4 - Descripción del sistema desarrollado.

2.1 ANALISIS DEL HARDWARE DE ACUERDO A LA NECESIDAD DEL USUARIO

Se ha tratado en el capítulo anterior, algunas generalidades que hay que tener en cuenta para un diseño determinado; en el presente capítulo se analiza de manera más detallada algunas necesidades comunes para los diferentes usuarios del microcontrolador, así como también algunas funciones específicas que se tiene presentes en los microcontroladores de esta familia, y que no necesariamente se utiliza; pero siempre es conveniente conocerlas para que cualquier aplicación a desarrollarse no interfiera de una u otra manera con el hardware o software del diseño.

2.1.1 Estructura de interrupciones en la familia MCS-51

Al igual que la mayoría de los microprocesadores y microcontroladores de las últimas generaciones, la familia de los microcontroladores MCS-51/52 también poseen una estructura de interrupciones, las cuales en determinados casos y aplicaciones son muy versátiles en su uso y constituyen una herramienta de gran ayuda al usuario.

En éste punto es muy conveniente basar el análisis en un elemento típico de los componentes de ésta familia, para detallar de manera más específica su manipulación y

programación, pero hay que tomar en cuenta que todos los componentes de la familia tienen la misma estructura con ligeras variaciones dependiendo del elemento escogido.

De aquí para adelante, se tomará al microcontrolador 8031 como el elemento típico dentro de la familia y se hace referencia de él como el representante de todo el conjunto. El microcontrolador 8031 consta de 5 fuentes de interrupción las cuales se dividen en:

2 interrupciones externas, $\overline{\text{INT0}}$ e $\overline{\text{INT1}}$; 2 interrupciones efectuadas por los contadores/temporizadores, cuyas banderas se denominan TF0 y TF1; y una interrupción efectuada por el pórtico serial.

Cada una de estas interrupciones se encuentra vectorizada dentro del microcontrolador; es decir que cuando una de las interrupciones se activa el programa salta a ejecutarse en una localidad de memoria predeterminada.

Cada fuente de interrupción generalmente tiene a su servicio 10 localidades de memoria predefinidas las cuales pueden incrementarse si así se lo requiere por la ejecución de una instrucción de salto incondicional.

Las localidades de memoria dedicadas para esta aplicación en la mayoría de los casos se encuentran definidas desde la dirección 0003h, hasta la localidad 0030h; es por éste motivo que en los programas de aplicación se recomienda comenzar el programa desde la localidad 0030h para evitar cualquier interferencia de la programación con el servicio de atención a las interrupciones.

2.1.1.1 Habilitación de interrupciones.

Cada una de las fuentes de interrupción puede ser habilitada o deshabilitada activando o limpiando el bit respectivo en el registro de función especial IE (Interrup Enable) dedicado para éste propósito.

Así mismo existe otro registro de función especial denominado IP (Interrup Priority), en el cual de la misma manera que en el registro anterior se puede programar dos diferentes niveles de "prioridad"; es decir que una interrupción determinada puede tener un nivel "alto" o "bajo" de prioridad, ésto se lo realiza para el caso en que dos diferentes interrupciones pidan atención al mismo tiempo; la interrupción que tenga mayor prioridad será la que se atienda primero, para luego seguir con la interrupción que tiene la prioridad más baja.

Si el usuario del microcontrolador, está seguro de no necesitar las interrupciones, se tiene la posibilidad de deshabilitar éstas para aprovechar las localidades de memoria que quedarían sin uso, pero generalmente se recomienda dejar éstas localidades vacías, para tener a futuro la posibilidad de ampliación utilizando interrupciones.

2.1.1.2 Acceso a memoria de programa y memoria de datos.

Como se ha mencionado, el acceso a memoria de programa o memoria de datos; dentro del 8031 esta claramente definido por las señales $\overline{PSEN}$, $\overline{RD}$, y $\overline{WR}$ con las cuales se puede trabajar en diferentes tipos de combinaciones, para lograr una amplia gama de acceso a diferentes tipos de elementos como memorias, teclados, displays, conversores, etc.

Cabe anotar que con la utilización de pocos elementos adicionales a la configuración elemental de funcionamiento del 8031 se logra manipular y accesar a casi cualquier elemento que se necesite en un diseño a desarrollarse.

La elección de la memoria de programa adecuada, está básicamente determinado por la extensión en kilobytes que tenga la aplicación diseñada, recomendándose siempre dejar un margen de localidades vacías, para tener la posibilidad de ampliación o variación en el programa.

Los chips de UVEPROM mas conocidos dentro de nuestro medio son los de la casa INTEL tales como el 2716 (2K x 8); 2732 (4K x 8); 2764 (8K x 8), con los cuales se pueden hacer diferentes arreglos para obtener la extensión mas adecuada para cualquier aplicación.

Un ejemplo típico del acceso a memoria de programa externa se lo puede encontrar en la figura 2.6, en donde se elige el chip 2732 como la memoria de programa.

Para el caso de memoria de datos; el chip 6116 (2K x 8), es uno de las más conocidos, así mismo si se llegara a

necesitar un volumen mayor de almacenamiento, se pueden hacer diferentes tipos de arreglos con varios chips de dicha memoria. Un ejemplo del acceso a memoria de datos externa se lo puede encontrar en la figura 2.7.

De igual manera para este caso se recomienda dejar un espacio adecuado de localidades vacías para tener la posibilidad de un mayor almacenamiento de datos, si así se lo requiere en un futuro.

Para el caso en que se necesite los dos tipos de memoria, bastaría con juntar en un solo hardware los dos configuraciones anteriores (figura 2.6 y figura 2.7), nótese que las dos memorias trabajan con las mismas direcciones (iniciándose en la localidad Q000H), pero las señales de habilitación son diferentes, motivo por el cual no se tiene interferencia de ningún tipo entre las dos.

FIGURA 2.6: ACCESO A MEMORIA EXTERNA DE PROGRAMA.

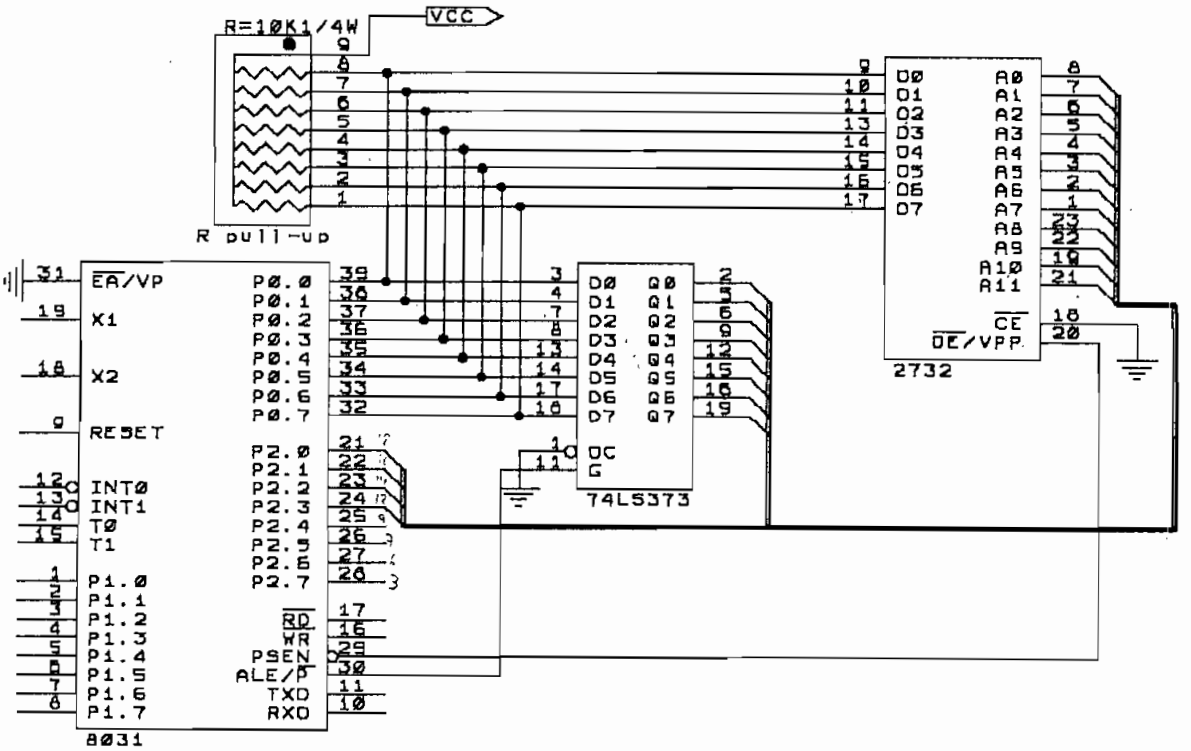
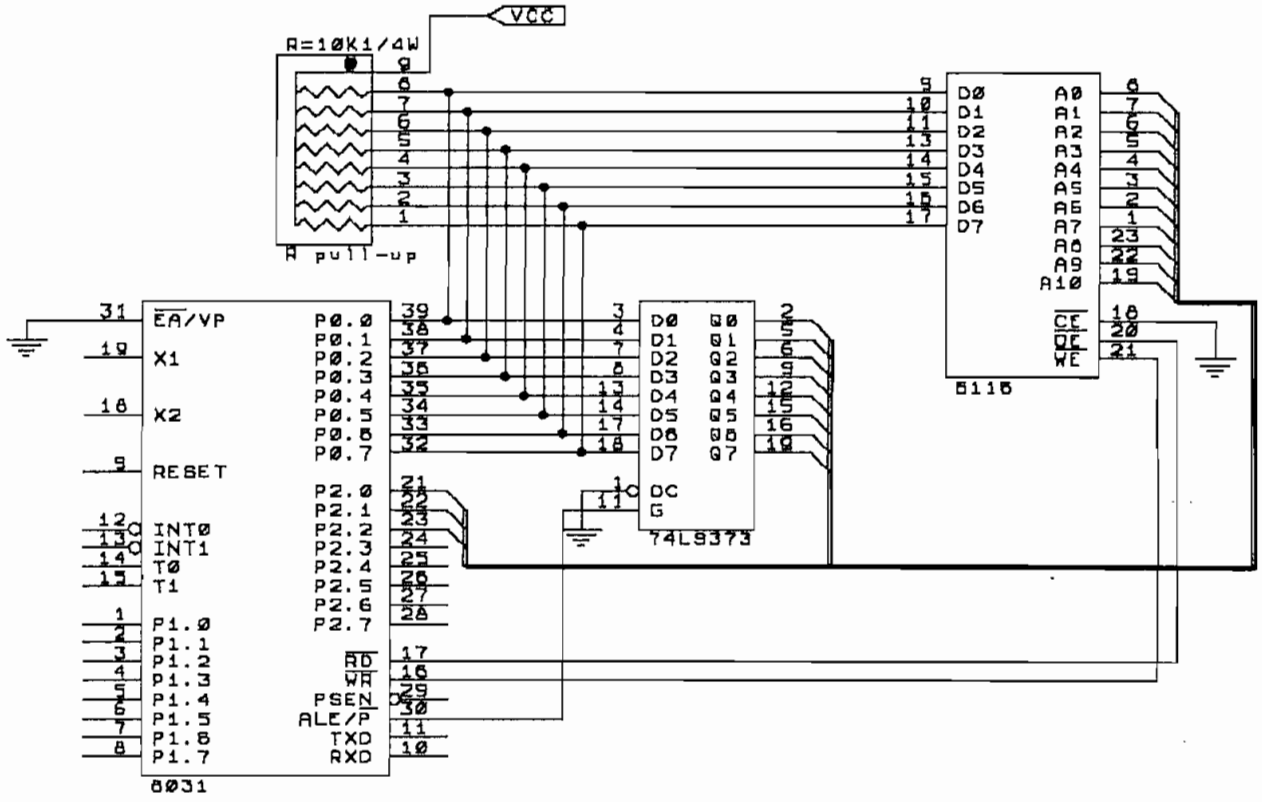


FIGURA 2.7: ACCESO A MEMORIA EXTERNA DE DATOS.



2.2 PORTICOS ADICIONALES DE ENTRADA/SALIDA DE DATOS

Una de las ventajas de los microcontroladores de la familia MCS-51, es precisamente la versatilidad con la que se puede introducir datos de diferente naturaleza al microcontrolador; ya que cualquier ingreso de datos es tratado como una lectura desde RAM externa, lo cual da una simplificación tanto en hardware como en software.

2.2.1 Decodificación de direcciones.

Generalmente cuando se quiere implementar varios pórticos ya sean de entrada o salida de datos, se realiza mediante el direccionamiento de cada pórtico, con lo cual se tiene completo control sobre cada uno de los elementos del sistema.

Como elemento decodificador de direcciones se usa el integrado 74LS138, el cual es un Decodificador de 3 a 8 con salidas activas en bajo (OL) y tres entradas de habilitación al integrado, dos en bajo y una en alto.

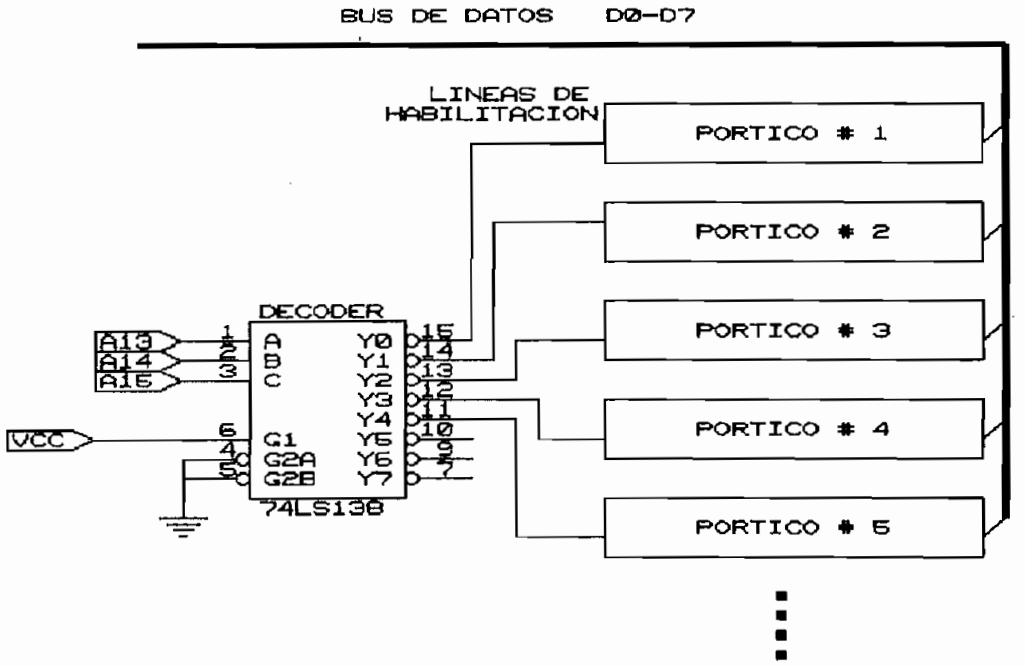
La manera de decodificar y la dirección que se quiera dar a un elemento dado es absoluto criterio del diseñador, pero a continuación se expone una de ellas como ejemplo de aplicación.

Se recomienda usar los bits más significativos del bus de direcciones como entradas al decodificador, de ésta manera se tiene los bits menos significativos del bus de

direcciones a disposición de los elementos a conectarse.

La idea general del direccionamiento se lo puede observar en la figura 2.8.

Figura 2.8: Direccionamiento de pórticos.



Como se aprecia en el gráfico anterior mediante el empleo de los tres bits más significativos del bus de direcciones se tiene la posibilidad de accesar hasta un máximo de 8 pórticos adicionales, ya sean de entrada o salida de datos.

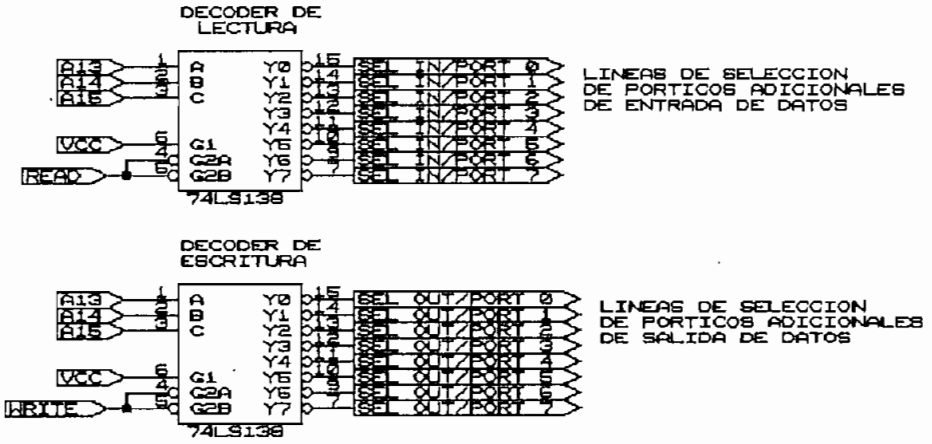
El acceso a los diferentes pórticos se logra en el ejemplo anterior mediante las direcciones:

BUS DE DIRECC.				HEX.	LINEA DE HABILITACI. ACTIVA
A15	A14	A13	A12-A0		
0	0	0	XX	0000-1FFF	PORTICO # 1
0	0	1	XX	2000-3FFF	PORTICO # 2
0	1	0	XX	4000-5FFF	PORTICO # 3
0	1	1	XX	6000-7FFF	PORTICO # 4
1	0	0	XX	8000-9FFF	PORTICO # 5
1	0	1	XX	A000-BFFF	PORTICO # 6
1	1	0	XX	C000-DFFF	PORTICO # 7
1	1	1	XX	E000-FFFF	PORTICO # 8

Nótese que en este caso el decodificador de direcciones, se encuentra permanentemente habilitado; siguiendo la misma filosofía se puede conectar las entradas de habilitación del decodificador a cualquier señal del microcontrolador de tal manera que éste entre en funcionamiento en algún ciclo especial definido por el usuario, generalmente se utilizan las señales de $\overline{RD}$ y $\overline{WR}$ de tal manera de obtener ciclos de lectura o escritura a un pórtico determinado en alguna dirección predefinida.

Una manera de ampliar y a la vez separar pórticos, se puede lograr habilitando un decodificador específico mediante la señal de $\overline{RD}$; y de la misma manera habilitar otro decodificador diferente mediante le señal de $\overline{WR}$, así se obtiene ocho pórticos exclusivos para entrada de datos, y ocho pórticos exclusivamente para salida de datos. Una configuración como la mencionada se encuentra en la figura 2.9.

Figura 2.9: Habilitación de pórticos exclusivos de lectura/escritura.



2.2.2 Salida de datos hacia displays.

Una de las salidas de datos que por lo general se encuentra en la mayoría de aplicaciones, es la referente a la salida de datos hacia displays; uno de los displays mas conocidos es el de cátodo común LTS-5501 de siete segmentos, con el cual y de acuerdo a la cantidad de datos que se desee mostrar se puede hacer un arreglo de varios dígitos para obtener la visualización deseada.

Para planificar el arreglo de dígitos, primero se necesita saber el número de datos que se deben mostrar, por ejemplo, en una aplicación dada se necesita conocer al mismo tiempo que dato se encuentra contenido en una localidad de memoria determinada, partiendo de ésto se analiza que para mostrar un dato de ocho bits, se necesita tener dos dígitos; ahora para poder mostrar la localidad de memoria en la cual está dicho dato, se necesita tener cuatro dígitos más, ya que el bus de direcciones cuenta con 16 bits; es decir el arreglo que se necesita debe de

constar de seis dígitos.

Cada dígito (Display LTS 5501) no es mas que un arreglo de diodos Led dispuestos en una determinada posición, con todos los cátodos unidos a un mismo punto (para el presente caso), y con el ánodo del Led en alguna patilla del display. Para encender un determinado segmento del display basta con conectar el cátodo común a la referencia y luego colocar un valor positivo de voltaje (1L) al ánodo considerando siempre la respectiva resistencia de limitación de corriente.

Para el display LTS 5501, cada segmento soporta una corriente máxima de 30 mA, y el voltaje directo VF de los led´s es de 1.9 V. Para obtener un adecuado nivel de iluminación en los diodos se recomienda escoger una corriente del orden de los 20 a 25 mA.

La representación del arreglo de segmentos en el display se encuentra en la figura 2.11.

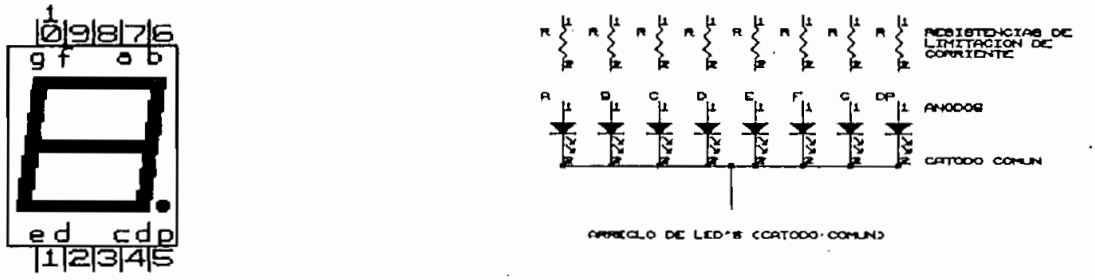


Figura 2.11: Representación esquemática y en segmentos del display LTS 5501.

Una técnica que generalmente es la más utilizada para mostrar los datos deseados en este tipo de arreglos es la denominada técnica de barrido, en la cual se van mostrando los datos de "uno" en "uno" habilitando un dígito a la

vez, para luego hacer un barrido permanente a alta velocidad de todo el arreglo de displays de tal forma que aparentemente, todos los datos que se muestren parecen como que si estuvieran presentes al mismo tiempo en el arreglo de dígitos.

La capacidad de carga de las salidas del microcontrolador, cumplen con las especificaciones de todo circuito TTL,(Corriente de entrada en alto = 1-2 mA, corriente de salida en bajo = 20 mA) por tanto siempre se necesita de un elemento de enlace entre las salidas del microcontrolador y el display, para no sobrecargar las salidas del microcontrolador. Este elemento generalmente es el circuito integrado 74LS373 que es un Latch octal con flip-flops tipo D disparados por flanco positivo.

En general se puede decir que para cualquier salida de datos a un elemento externo, el 74LS373 es el circuito mas idóneo para el efecto.

Otro aspecto relevante de la salida de datos hacia displays es lo referente a la habilitación de cada uno de los dígitos; como estos son de cátodo común, para habilitar o deshabilitar cada dígito basta con conectar o desconectar el cátodo del terminal negativo o tierra.

Para este efecto se usa un transistor que actue como switch de alta velocidad para habilitación.

El diseño propuesto para la salida de datos hacia displays se lo presenta en la figura 2.12, en este diseño para las resistencias de limitación de corriente de los segmentos de cada dígito se tiene que como los transistores de

habilitación trabajan en la zona de saturación (switch) se considera un voltaje típico de 0.2 V entre colector-emisor, por tanto:

$$R=\frac{VCC-VF-VCE}{20mA}=\frac{5V-1.9V-0.2V}{20mA}=145\Omega$$

$$\therefore \quad R=150\Omega$$

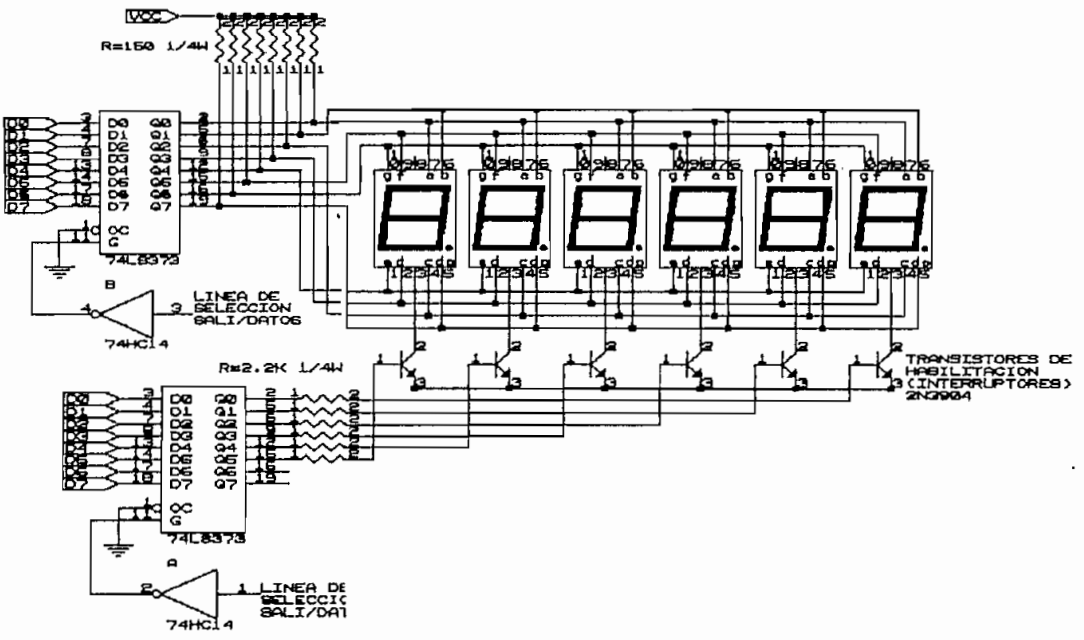
Para las resistencias de base en los transistores, se tiene que como $\beta_{SAT}=20$ y la corriente de colector es de 20 mA entonces:

$$I_{B_{SAT}}=\frac{I_C}{\beta_{SAT}}=\frac{20mA}{20}=1\ mA.$$

$$R_B=\frac{V_{OH_{TTP}}-V_{B_{SAT}}}{1mA}=\frac{3.4V-0.7V}{1mA}=2.7\ K\Omega$$

$$\therefore \quad R_B = 2.2\ K\Omega \quad (PARA\ ASEGURAR\ LA\ SATURACION).$$

Figura 2.12: Circuito de Salida de datos hacia Displays.



Retomando el ejemplo anterior, digamos que en la localidad 7654H se encuentra contenido el dato 12H, para mostrar

ésto en el display primero se tiene que sacar al bus de datos el código de 7 segmentos del nibble menos significativo del dato contenido en la localidad, en éste caso el "2"; por la dirección que tenga la línea de selección de salida de datos (1), de tal suerte que el dato sacado se quede retenido en el latch a continuación, por la dirección de la línea de selección de datos (2) sacamos un byte tal que habilite el dígito menos significativa, en éste caso se debe de poner el # 01h, con ésto se logra mostrar el 2H en el primer dígito del display, a continuación se debe dejar habilitada éste dígito por unos 30 a 40 milisegundos, se desabilita este dígito; cambiamos el dato en el latch (1) ahora por el código de 7 segmentos del número "1", a continuación por el latch (2) se habilita el segunda dígito colocando # 02H, siguiendo el mismo procedimiento que en el caso anterior se sigue sacando a continuación los números "4", "5", "6", "7" por los subsiguientes dígitos habilitándolos con los números 04H, 08H, 10H, 20H; seguidamente se vuelve a repetir todo el procedimiento nuevamente, de ésta manera se tiene en el arreglo todos los datos que inicialmente se quiso tener.

Nótese que los datos se van sacando de uno en uno pero debido a la velocidad de realización del proceso el ojo humano es incapaz de notar ésto y para el aparentemente todo el dato está en conjunto puesto en las pantallas.

Un diagrama de tiempos de estas señales se tiene en la figura 2.13.

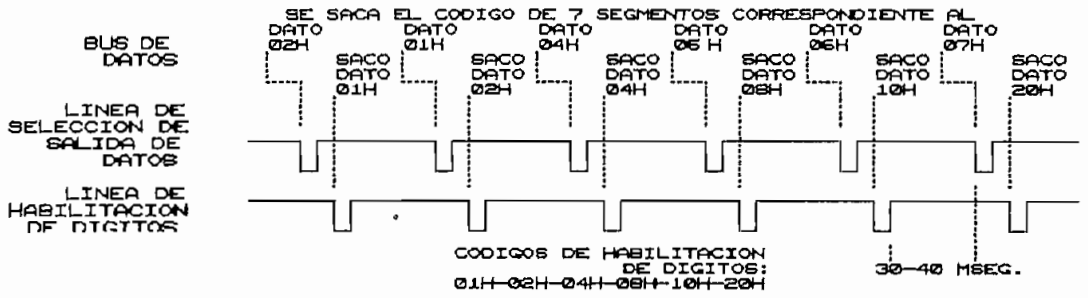


Figura 2.13: Diagrama de tiempos de las señales de habilitación para salida hacia displays.

Un aspecto que hasta aquí no se ha mencionado, es la particularidad de codificación que se tiene que hacer con los datos, antes de enviarlos al display para que estos se presenten de manera legible al usuario.

- Codificación de datos para salida a displays.

Como cada dígito del display es un arreglo de 7 segmentos mas el punto decimal (arreglo de leds) para presentar un dato en el display, estos tienen que sufrir una transformación previa para que el display muestre el dato deseado. Como en el circuito implemento (figura 2.12) se tiene que:

Q0 del latch de salida de datos U-10 controla al segmento "a" del display.

Q1 al segmento "b". Q2 al segmanto "c". Q3 al segmento "d". Q4 al segmento "e". Q5 al segmento "f". Q6 al segmento "g" y Q7 al segmento "dp".

Si por ejemplo en el display se quiere mostrar el número "2", en el latch de salida de datos se tendría que tener

el número "5BH", el número "3" se muestra con el número "4FH" en el latch de salida de datos. Es decir cada número tiene su correspondiente código para poder mostrarse como tal en el display, por tanto se puede realizar una tabla de codificación dispuesta convenientemente en una subrutina para llamarla cuando se necesite sacar un dato hacia los displays. La tabla de codificación para salida de datos hacia displays se presenta a continuación.

TABLA DE CODIFICACION DE DATOS									
DP	G	F	E	D	C	B	A	# HEX.	DIGITO
Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0		
0	0	1	1	1	1	1	1	3F	0
0	0	0	0	0	1	1	0	06	1
0	1	0	1	1	0	1	1	5B	2
0	1	0	0	1	1	1	1	4F	3
0	1	1	0	0	1	1	0	66	4
0	1	1	0	1	1	0	1	6D	5
0	1	1	1	1	1	0	1	7D	6
0	0	0	0	0	1	1	1	07	7
0	1	1	1	1	1	1	1	7F	8
0	1	1	0	1	1	1	1	6F	9
0	1	1	1	0	1	1	1	77	A
0	1	1	1	1	1	0	0	7C	b
0	0	1	1	1	0	0	1	39	C
0	1	0	1	1	1	1	0	5E	d
0	1	1	1	1	0	0	1	79	E
0	1	1	1	0	0	0	1	71	F

Tabla 2.2: Codificación de datos para salida a displays.

2.2.3 Entrada de datos desde teclado.

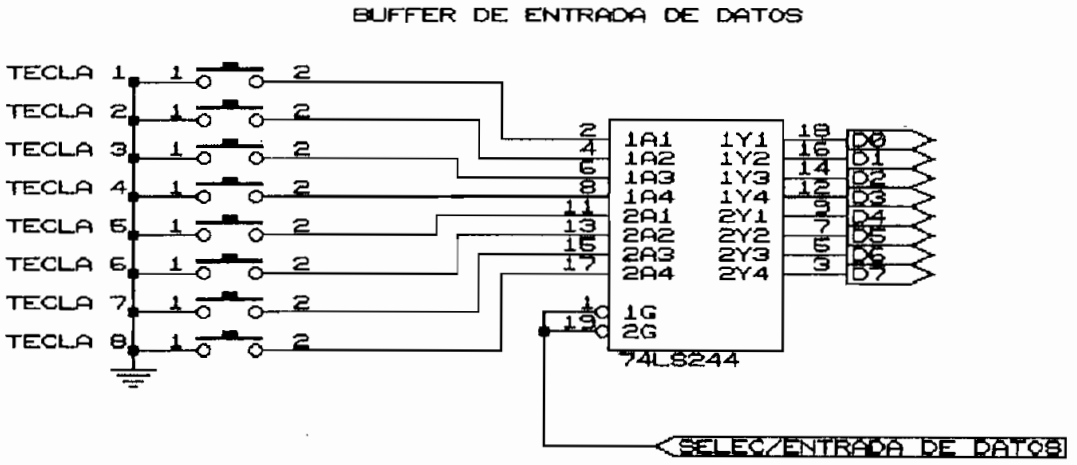
Otro pórtico adicional que es muy importante y utilizado en casi todas las aplicaciones es la entrada de datos desde teclado.

El número de teclas que tenga este, depende de la aplicación y de la necesidad del diseñador.

Para la entrada de datos desde teclado únicamente se necesita de un buffer que esté de acuerdo al número de teclas que se tenga.

Un diseño de una entrada de datos desde ocho teclas, se lo puede apreciar en la figura 2.14

Figura 2.14: Entrada de datos desde teclado



En general se puede decir que para cualquier entrada de datos el elemento más idóneo es el circuito integrado 74LS244, el cual es un buffer octal con dos entradas de habilitación en bajo y con drivers de salida tres estados.

En este diseño cuando se presiona una tecla, en el buffer

de entrada de datos se tiene que la línea a la cual esta conectada dicha tecla, cambia de estado, de 1L a 0L, entonces para saber cual tecla se ha presionado únicamente hay que averiguar cual de las ocho líneas del bus de datos ha cambiado de estado.

Por ejemplo cuando se presiona la tecla número 8, el código que se obtiene al leer este pórtico será el # 7FH, de igual manera si se presiona la tecla 7 se obtiene el # BFH, para la 6 es el # DFH, y para las subsiguientes EFH, F7H, FBH, FDH, FEH respectivamente; con estos códigos dispuestos convenientemente en una tabla de decodificación se puede saber cual tecla se ha presionado.

Este diseño de entrada de datos esta realizado en la técnica denominada Polling o de barrido permanente.

Es decir cada determinado tiempo hay que accesar a la dirección asignada al pórtico adicional de entrada de datos desde teclado y averiguar si ha existido un cambio en los bits de entrada.

Cuando no se presiona ninguna tecla, el código de entrada cuando se lea este pórtico será el número "FFH"

2.3 HARDWARE PARA COMUNICACIONES

Una de las ventajas de los elementos de la familia de los microcontroladores MCS-51, es la posibilidad de implementar comunicación con cualquier elemento externo que utilice el formato serial.

Esto se puede lograr mediante el uso del Pórtico Serial del propio microcontrolador; éste pórtico puede trabajar en el modo Full Duplex, lo cual implica que se puede recibir y trasmitir al mismo tiempo.

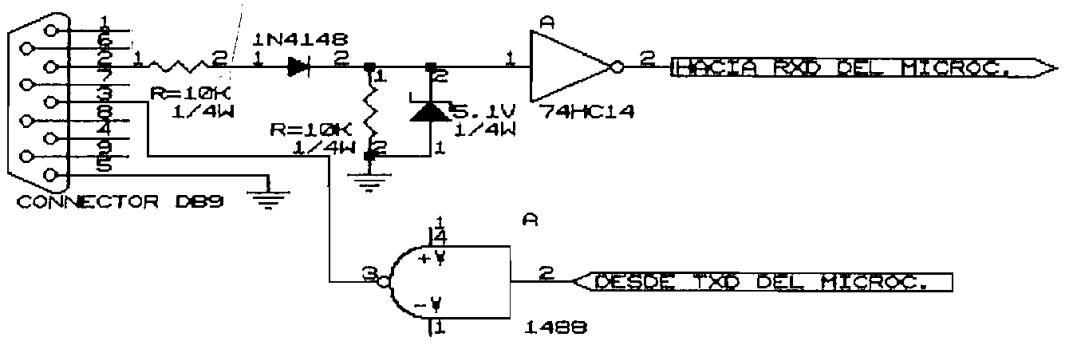
A los registros de transmisión y recepción del pórtico serial se puede acceder por un registro único de enlace denominado SBUF (Serial Buffer).

El hardware que se debe implementar para tener la posibilidad de comunicación serial, está determinado por la Norma de Comunicación Serial RS 232-C. En la cual un "0L" tiene un nivel de voltaje mayor a +3V y un "1L" se representa con un nivel de voltaje menor a -3V.

Los computadores personales para la comunicación serial generalmente operan dentro de los niveles +/-9V hasta +/-12V.

El circuito implementado es tal que, cuando por el pórtico serial del computador se tiene un nivel positivo de voltaje, en la salida se obtiene un "0L", y para un nivel negativo de voltaje se obtiene un "1L", adicionalmente se hace una transformación a niveles TTL, que son los niveles con los cuales trabaja el microcontrolador. El diseño implementado se lo presenta en la figura 2.15.

Figura 2.15: Hardware para comunicaciones.



Cuando se tiene un valor positivo de voltaje, el diodo 1N4148 se polariza directamente dejando pasar la señal hacia el diodo zener el cual al polarizarse inversamente fija la entrada al 74HC14 en 5.1V, con lo cual a la salida del mismo se obtiene un 0L; cuando en cambio se tiene un valor negativo de voltaje, el diodo se polariza inversamente, a la entrada del 74HC14 se tiene un 0L, por tanto a la salida del mismo se tiene un 1L.

2.4 DESCRIPCION DEL SISTEMA

Para el caso de la presente aplicación y debido a la naturaleza de la misma; el hardware implementado es lo más general posible para adaptarse a la mayoría de las aplicaciones y necesidades de los usuarios.

El microcontrolador elegido para éste caso, se trata del MCS-8031, obviamente se necesita la presencia de memoria de programa, para lo cual se tiene la UV-EPROM 2732 de la casa Intel, esta tiene la posibilidad de almacenar hasta 4 Kilobytes de información, en ésta memoria se encuentra residente el programa Monitor del módulo de desarrollo.

Como dentro del sistema se tiene varios pórticos de entrada/salida de datos adicionales, se cuenta con dos decodificadores de pórticos uno para pórticos de entrada de datos y otro para pórticos de salida de datos.

Uno de los pórticos adicionales es una salida de datos hacia displays, la cual cuenta con seis dígitos de cátodo común, habilitados por transistor.

Para el ingreso de datos, se cuenta con un pórtico adicional de entrada de datos desde teclado; éste cuenta con ocho teclas con una entrada activa en bajo.

El procesamiento de datos analógicos dentro de un entorno digital representaba hasta hace no mucho tiempo una dificultad casi insalvable, pero hoy en día con el perfeccionamiento de conversores Analógico/Digital y Digital/Analógico éste procesamiento se ha simplificado enormemente; debido a la importancia que ésto representa,

en el módulo de desarrollo se considera un pórtico de entrada de datos análogos a través de un conversor Análogico/Digital.

Existe un pórtico de entrada de datos desde DIP Switch, mediante un banco de ocho DIP switchs cuya entrada activa es en bajo, el DIP número uno es el menos significativo y el número ocho es el más significativo.

Se cuenta con el pórtico de comunicaciones seriales, el cual tiene una salida a través de un conector DB9 en la parte posterior del módulo.

2.4.1 Modos de operación del módulo de desarrollo.

El módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51/52 cuenta con dos modos de operación en lo que se refiere a memoria de almacenamiento de datos. Cada uno de éstos modos es accesible independientemente a través de un conmutador localizado en el panel frontal de control.

- Modo Uno (Modo Normal).

En el módulo de desarrollo se cuenta con dos memorias RAM 6116 de 2K por ocho bits, éstas memorias están organizadas para trabajar en dos modos de operación. En el modo uno o Modo de Operación Normal las dos RAMs son accesibles para operaciones de lectura/escritura, mediante dos diferentes líneas de selección.

La característica fundamental de éste modo de operación es que los códigos de máquina que necesita el microcontrolador para su funcionamiento se toman desde la EPROM 2732 dispuesta para éste efecto.

En este modo de operación se cuenta con 4K de ROM localizada en la dirección hexadecimal 0000H-0FFFH, y con los dos RAMs localizadas en la dirección:

8000H-87FFH; RAM de propósito general.

A000H-A7FFH; RAM que simula EPROM.

- **Modo Dos (Simulación de EPROM).**

Este modo de operación, denominado Modo de Simulación de EPROM, se caracteriza por que los códigos de máquina ya no se toman desde la EPROM 2732; sino que en este caso una de las RAMs ocupa el lugar de esta memoria, y los códigos de máquina se toman desde RAM. Por éste motivo a éste modo se le ha denominado Simulación de EPROM. En este modo de operación se cuenta con dos ROMs localizadas en las direcciones:

0000H-07FFH; RAM que toma el lugar de la 2732.

1000H-1FFFH; EPROM 2732 que contiene las subrutinas de uso general de los usuarios.

8000H-87FFH; RAM de propósito general.

Nótese que la RAM que en el modo uno tenía la dirección A000H-A7FFH, en este modo se relocaliza a la dirección 0000H-07FFH.

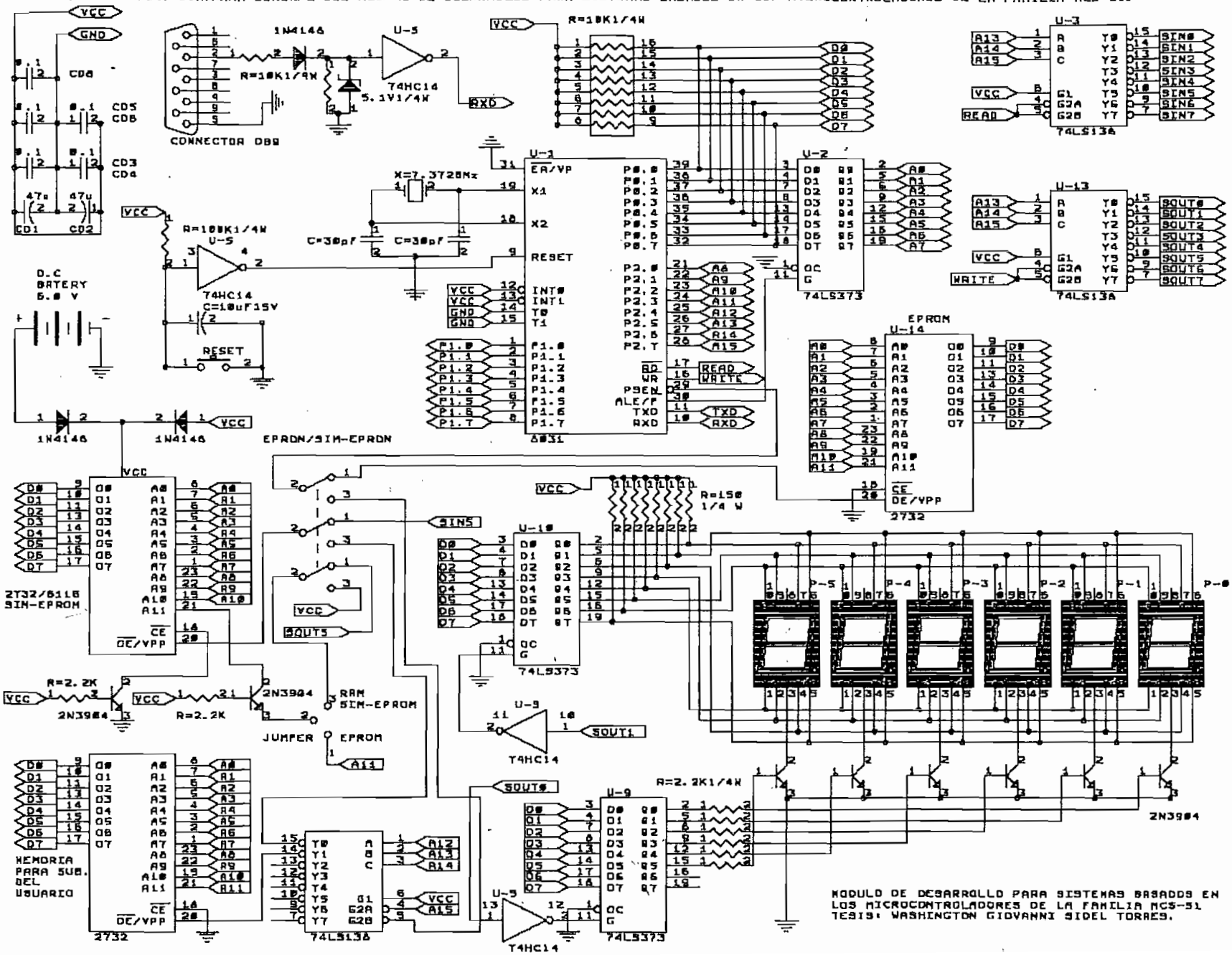
Dentro de este modo de operación mediante el uso de un

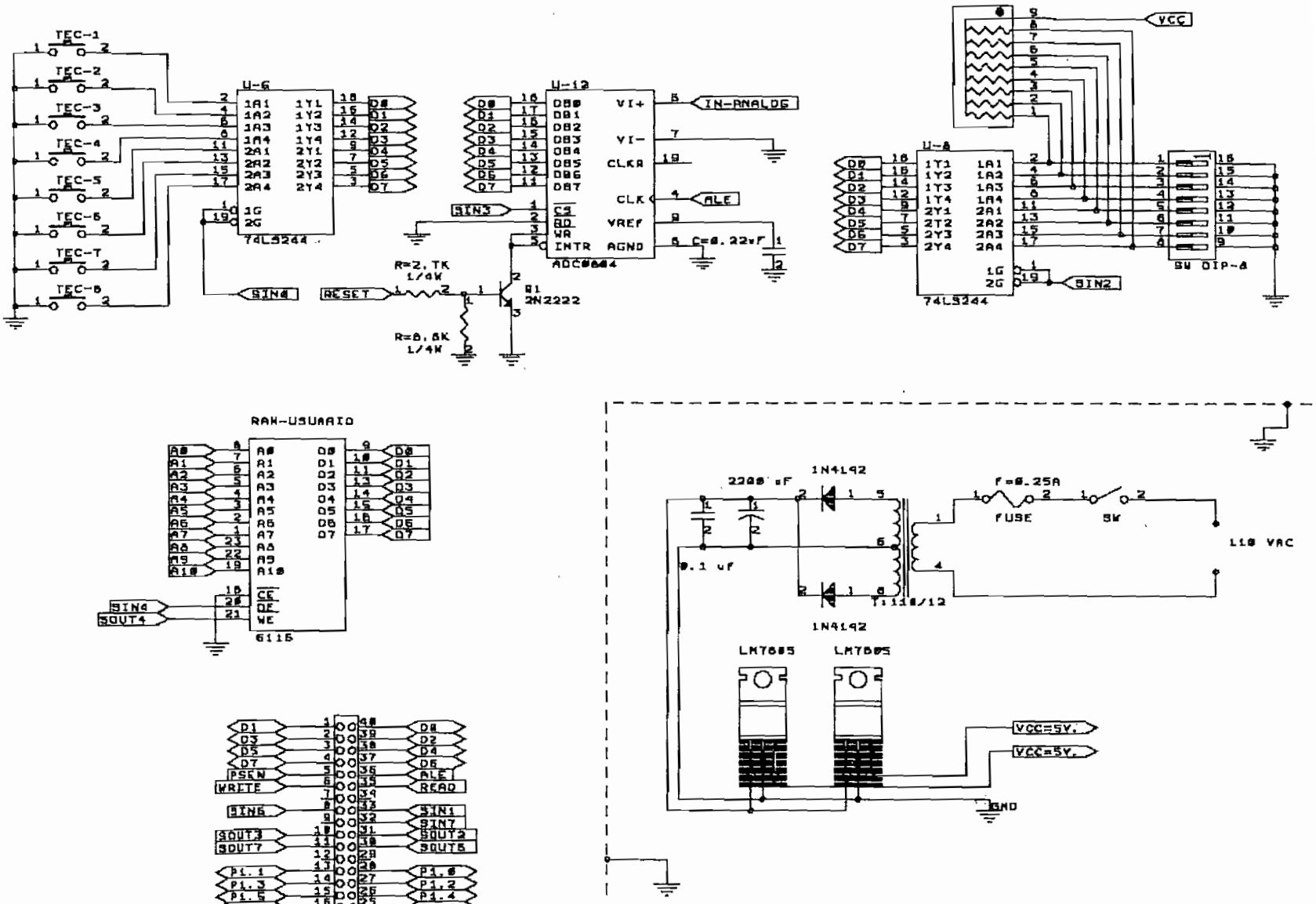
jumper localizado en la parte posterior del módulo se tiene la posibilidad de ejecutar el programa desde RAM simulando memoria de programa, que es generalmente la opción ha usarse, pero adicionalmente se puede cambiar la posición del jumper y ejecutar el programa desde EPROM pregrabada con anterioridad. La condición para esta opción es que la memoria tiene que ser 2732.

Cabe anotar la versatilidad de estos dos modos de operación trabajando conjuntamente ya que a una RAM a la cual se la puede escribir, cambiar de datos, o sobreescibir en cualquier localidad (modo uno), se la hace trabajar como EPROM (modo dos), por tanto si se carga un programa en esta memoria; luego se cambia al Modo de Simulación y una vez que se observen los resultados, se necesita hacer un cambio en el programa, basta con salir del Modo de Simulación y realizar cualquier cambio tratando la memoria como RAM, para luego entrar nuevamente al modo de simulación y ver si los resultados son los esperados. Si éste mismo proceso se quisiera hacer con una EPROM real se tiene que hacer una serie de pasos que implican largo tiempo de trabajo hasta lograr el mismo resultado. Cualquier diseñador que haya pasado por una situación similar sabe lo largo y tedioso que resulta hacer un pequeño cambio en una EPROM.

Un diagrama general del módulo de desarrollo se encuentra en la figura 2.16, en donde se pueden apreciar todos sus componentes.

FIGURA 2.15: DIAGRAMA GENERAL DEL MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS MICROCONTROLADORES DE LA FAMILIA MCS-51.





MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS MICROCONTROLADORES DE LA FAMILIA MCS-51
TESIS: WASHINGTON GIOVANNI SIDEL TORRES.

2.4.2 Direccionamiento y mapa de memoria.

Como se ha explicado, cada pórtico dentro del módulo de desarrollo puede ser accesado mediante una dirección determinada en un ciclo de lectura o escritura externa.
Un resumen de las direcciones en la que se localizan los diferentes pórticos se puede observar en la siguiente tabla.

DIRECCION DE PORTICOS						
BITS DE DIREC			DIRECC.	LINEA	DENOMINACION	
A15	A14	A13	HEX.	SELECCI	READ	WRITE
0	0	0	0000-1FFF	0	TECLADO	HAB. DE DIGITOS *
0	0	1	2000-3FFF	1	USUARIO	SAL. DAT DISPLAYS *
0	1	0	4000-5FFF	2	DIP-SW.	USUARIO
0	1	1	6000-7FFF	3	C. AN/DIG.*	USUARIO
1	0	0	8000-9FFF	4	RAM	RAM
1	0	1	A000-BFFF	5	RAM SIMUL EPROM	RAM SIMUL EPROM*
1	1	0	C000-DFFF	6	USUARIO	USUARIO
1	1	1	E000-FFFF	7	USUARIO	USUARIO

Tabla 2.3: Dirección de pórticos adicionales externos.

* -C.AN/DIG: Conversor Analógico/Digital
* -HAB.DE DIGITOS: Salida de habilitación de dígitos.
* -SAL.DAT.DISPLAYS: Salida de Datos hacia Displays.
* -RAM SIMUL EPROM: Memoria RAM que simula EPROM.

El mapa de memoria del módulo esta de acuerdo al modo de operación en el que se encuentre. En la siguiente tabla se puede encontrar el mapa de memoria para los dos modos de operación.

MAPA DE MEMORIA			
MODO #1		MODO #2	
R.O.M	R.A.M	R.O.M	R.A.M
DIRECCION EN HEXADECIMAL			
0000-0FFF	8000-87FF	0000-07FF	8000-87FF
	A000-A7FF	1000-1FFF	

Tabla 2.4: Mapa de memoria de acuerdo al modo de operación.

Se recomienda usar únicamente la RAM localizada en la dirección inicial 8000H (8000H-87FFH) como banco de almacenamiento de datos, ya que la RAM que simula EPROM cumple con la función de almacenar los códigos de máquina simulando EPROM, además de no estar disponible en todos los modos.

Así mismo de acuerdo a la posición del jumper en el modo 2 se puede disponer de 6K (0000h-07FFh 1000-1FFFh) u 8K (0000h-1FFFh) bytes como memoria de programa.

2.4.4 RAM no volátil.

Cuando se trabaja en el modo dos del módulo de desarrollo, se tiene que una RAM simula ser EPROM, una desventaja de esto constituye que si se corta la energía de alimentación todos los datos que contiene las RAMs se pierden, y peor aún si se trata de un programa en el cual ya se haya trabajado por largo tiempo, el resultado es desastrozo. Para remediar esto el módulo de desarrollo cuenta con

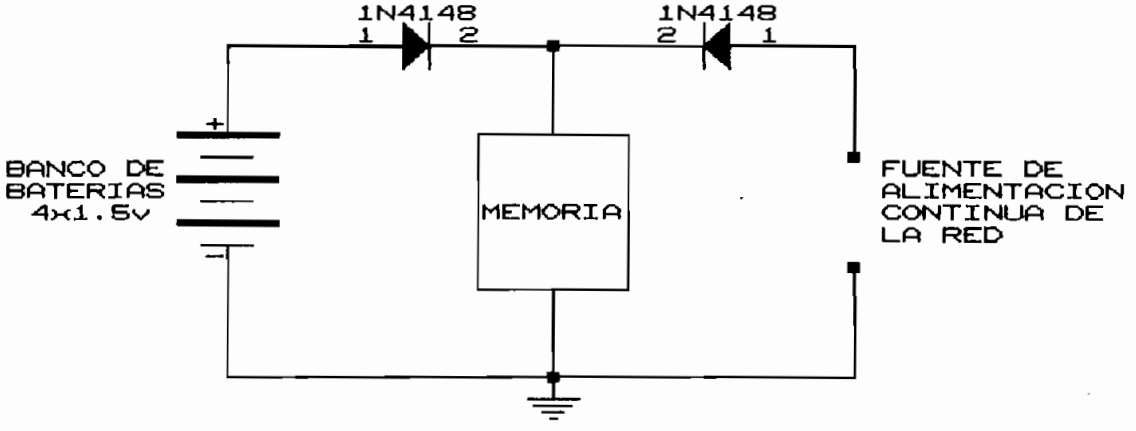
baterías auxiliares como fuente de alimentación continua a la memoria, esta fuente dispone de un interruptor electrónico que conmuta de la alimentación normal al banco de baterías. De esta forma se consigue que cualquier dato almacenado en la memoria permanezca intacto hasta que se reestablesca la alimentación normal.

La fuente de alimentación auxiliar únicamente es para la SIM-EPROM, y no para el resto del circuito.

Cuando el módulo esta trabajando con la alimentación de la red, el banco de baterías esta desconectado; pero si se pierde la energía de la red o esta decae a un valor crítico entra a funcionar las baterías alimentando a la memoria y desconectándola del resto del circuito.

El circuito implementado para esta función se presenta en la figura 2.17.

Figura 2.17: Fuente de alimentación ininterrumpida a memoria.



CAPITULO III

SOFTWARE

3.1 - Interface Serial de la familia MCS-51.

3.2 - Protocolo de comunicaciones.

3.3 - Programación de pórticos en la familia MCS-51.

3.4 - Programa del módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51.

3.1 INTERFACE SERIAL DE LA FAMILIA MCS-51

Como se mencionó anteriormente el Pórtico Serial de la familia MCS-51 trabaja en el modo Full-Duplex, el byte de datos "ingresa" o "sale" a través de la línea RXD o TXD respectivamente en el microcontrolador.

Los buffers de recepción y transmisión del microcontrolador cuentan con un registro de enlace "SBUF" (Serial Buffer), al cual se lo trata como una localidad de memoria facilitando enormemente su uso.

Para trasmitir un byte determinado, basta con realizar una operación de escritura al registro SBUF; mientras que si se realiza la operación ciclo de lectura desde SBUF, se accede al buffer de recepción.

Para la recepción se cuenta con un segundo buffer el cual puede recibir un segundo byte de datos antes de que el primero, previamente recibido, haya sido leido desde el registro de enlace SBUF. Si el primer byte no ha sido leido en el tiempo en que el segundo byte se completa, el primer byte se perderá.

El pórtico serial puede operar en cuatro modos seleccionables a través de los bits 6 y 7 (SM1 y SM0) del registro de funciones especiales SCON (Serial Port Control).

- Modo 0 :

En éste modo de operación la palabra de información a trasmitir o recibir es de 8 bits, siendo el primero en salir o en llegar el bit menos significativo.
La velocidad de transmisión expresada en baudios (baudio=bits/seg) es igual a 1/12 de la frecuencia de reloj del microcontrolador.
La velocidad de comunicación en Baudios esta dada por:

$$BAUDIOS_{MODO\ 0} = \frac{Frecuencia\ de\ reloj\ del\ microcontrolador}{12}$$

- Modo 1 :

Este modo de operación en la comunicación esta mas acorde con los estandares habituales. Utiliza 10 bits que son trasmitidos a través de TXD o recibidos a través de RXD. Los bits utilizados son:

- 1 bit de Partida (0L), que señala el comienzo de la palabra de datos.
- 8 bits de datos
- 1 bit de Parada (1L), que señala el final de la palabra de datos.

La velocidad de transmisión puede ser ajustada por el usuario dentro de un amplio rango.

- Modo 2 :

En este modo de transmisión/recepción se utilizan 11 bits distribuidos de la siguiente forma.

- 1 bit de Partida.
- 8 bits de datos.

- 1 bit de dato programable por el usuario (9º bit).
- 1 bit de Parada.

El noveno bit, en la transmisión puede ser "0" o "1", y es la imagen del bit TB8 que se encuentra en el registro SCON; así el bit de paridad P en PSW puede ser cargado en TB8 para ser asignado al noveno bit en la transmisión. En la recepción este noveno bit se refleja en RB8 del registro SCON, mientras que el bit de parada es ignorado. La velocidad de transmisión es programable a 1/32 o 1/64 de la frecuencia de reloj del microcontrolador.

Los baudios en este modo, dependen del valor del bit 7 (SMOD) del registro PCON; si SMOD=0, los baudios son 1/64 de la frecuencia de oscilación del microcontrolador. En cambio si SMOD=1 la velocidad de comunicación es 1/32 de la frecuencia de oscilación del microcontrolador. Es decir:

$$BAUDIOS_{MODO\ 2} = \frac{2^{SMOD}}{64} \times Frecuencia\ de\ reloj\ del\ microcontrolado.$$

- **Modo 3 :**

Este modo de transmisión/recepción es igual al modo anterior excepto que la velocidad de transmisión puede ser ajustada por el usuario dentro de un amplio rango.

Los baudios en los modos 1 y 3 están determinados por el valor del bit SMOD y la relación de "recarga automática" del timer 1; en el 8052 o similares la velocidad de comunicación puede ser obtenida utilizando el timer 1 o el

timer 2 o con ambos (uno para transmisión y uno para recepción).

El timer 1 debe ser configurado como temporizador en modo de autorecarga (TMOD=27H), en este caso los baudios vienen dado por:

$$BAUDIOS_{MODO\ 1\ Y\ 3} = \frac{\frac{2^{SMOD}}{32} \times Frecuencia\ de\ reloj}{12 \times (256 - TH1)}$$

Generalmente lo que se necesita conocer es el valor de recarga automática del timer TH1 en función de la velocidad en baudios; por lo que se tiene:

$$TH1 = 256 - \frac{Frecuencia\ de\ reloj}{\frac{X}{Velocidad\ en\ Baudios}}$$

Si SMOD=0 → X=384

Si SMOD=1 → X=192

Es con esta relación con la que se calcula el valor de recarga del timer uno para establecer la comunicación con el programa Sides-51 mediante la opción de programación de memorias, conociendo que esta trasmite los datos a 2400 baudios y la frecuencia de reloj del microcontrolador es igual a 7.3728 MHz, SMOD=0.

$$TH1 = 256 - \frac{7.3728MHz}{\frac{384}{2400}} = 248$$

$$\rightarrow TH1 = F8H$$

En todos los modos, la transmisión es iniciada mediante cualquier instrucción que use el registro SBUF como destino, una vez que el microcontrolador ha terminado de enviar el bit de stop activa la bandera TI (1L) como señal que el buffer de transmisión se encuentra vacío y se puede enviar otro dato; previamente hay que limpiar vía software la bandera TI.

En cambio la recepción se inicia, en el modo 0 por la condición RI=0 , REN=1 y en los modos 1, 2 y 3 cuando se recibe el bit de partida y REN=1.

Así mismo cuando el microcontrolador ha recibido el bit de parada activa la bandera RI (1L), como señal que el buffer de recepción se encuentra completo y se puede leer el dato recibido para vaciar el buffer, y nuevamente empezar la recepción de un nuevo dato, previamente hay que limpiar vía software la bandera RI.

3.2 PROTOCOLO DE COMUNICACION SERIAL.

Toda comunicación sea cual sea, tiene un protocolo preestablecido por una norma determinada al cual los usuarios tienen que sujetarse para poder entablar la comunicación deseada.

Dentro de las comunicaciones seriales existen varios tipos de protocolos con sus respectivas normas; pero para el caso del paquete de software SIDES-51 del cual se quiere recibir datos en forma serial hay que sujetarse a las normas del formato INTEL de ocho bits para la transmisión/recepción de archivos.

3.2.1 Formato para la recepción serial.

Un mensaje trasmitido/recibido en el formato Intel de ocho bits tiene una estructura como la que se indica en la siguiente tabla:

DESIGNACION DEL FORMATO	N° CARAC ASCII	DESCRIPCION DEL CONTENIDO
Caractéres de Cabecera.	1	Los dos puntos ":" (3AH) es señal del comienzo de un mensaje.
Longuitud del Mensaje.	2	Dos dígitos hexadecimales que indican el número de bytes de datos del mensaje .
Dirección de Partida	4	Cuatro dígitos hexadecimales que indican la localidad de memoria donde debe localizarce el primer "dato" del mensaje.
Tipo de Mensaje	2	00= Mensaje de datos. 01= Final de mensaje
Checksum	2	Dos dígitos hexadecimales que representan el complemento de dos de la suma módulo 256 de todos los bytes precedentes (exepto ":")

Tabla 3.5: Estructura del formato Intel de 8 bits.

El mensaje en formato Intel en hexadecimal puede ser:

:10000000270 40DA5159327D7D5BF8C043A9A8FA83E

En donde : → Cabecera.

10 → Número de datos (hex.) en el mensaje.

0000 → Dirección donde debe almacenarse el 1er dato.

00 → Mensaje de datos.

27..A8→ Datos.

3E → Checksum.

Esto es en hexadecimal, pero nótese que por el pórtico serial del computador personal los datos que salen de este se encuentran en formato ASCII por tanto se tiene que el mensaje a través del pórtico de computador será:

3A3130303030303032373034304441353135393332374437

443542463843303433413941384641383345

Estos códigos ASCII son los que el microcontrolador está recibiendo, por tanto hay que hacer una doble decodificación para obtener los datos del mensaje. El

primer paso es la transformación de los códigos ASCII a dígitos hexadecimales, y una vez que se tiene los códigos objeto, eliminar todo lo que es cabecera, longitud, dirección, control de errores, y obtener únicamente los bytes de datos.

El checksum es el complemento de dos de la suma módulo FFH (256), de todos los bytes contenidos en el mensaje, exepto el byte de cabecera 3AH (:).

3.3 PROGRAMACION DE PORTICOS EN LA FAMILIA MCS-51.

El microcontrolador 8031 y todos los elementos de la familia, tienen una serie de pórticos y registros a los cuales el usuario tiene acceso a programar o definir la forma o en que situación tienen que entrar en funcionamiento.

Cuando un programa que use un determinado pórtico o registro definible esta en la etapa de desarrollo, depuración u optimización el usuario generalmente esta accediendo a estos pórticos o registros de manera continua y realizando cambios en la operación o definiéndoles en un modo diferente de trabajo hasta encontrar el resultado deseado.

Ahora bien si no se tiene una metodología de programación adecuada estos cambios o redefiniciones pueden por un lado volverse tan confusos e intrincados que un simple cambio de parámetros por sencillos que estos sean resulte una enorme tarea, y por otro lado en lugar de optimizar la ejecución del programa lo vuelva mas complicado.

Una forma con la cual se obtienen excelentes resultados para cambios o reprogramaciones en pórticos o registros definibles es mediante el uso de etiquetas, y la realización de la programación del pórtico en una parte específica o mejor aún en una subrutina determinada del programa.

En realidad esto no es nada nuevo, pero si se etiqueta un

pórtico de frecuente uso, y que las etiquetas estén de acuerdo a las denominaciones de la función que desempeña o de acuerdo a la denominación del manual, se nota que para realizar cualquier cambio en la función del pórtico o registro basta con consultar el manual, o si las etiquetas puestas están acorde con la función que se necesita ni siquiera se tiene que hacer esta consulta, sino que únicamente observamos en el mismo programa la etiqueta de función que se desee para luego ir a la sección específica o subrutina del programa en donde se debe realizar el cambio, únicamente añadiendo o quitando etiquetas; sin necesidad de preocuparse por el valor (número en Hex.) que se deba de programar.

De esta manera ya no se tiene que recorrer el programa de pies a cabeza buscando todas las partes en donde se tiene que realizar las modificaciones correspondientes.

Y además no se necesita estar buscando el número o valor adecuado para la programación, sino que se programa la etiqueta adecuada y observamos el resultado.

Uno de los pórticos dentro del 8031 que regularmente siempre se necesita estar cambiando o redefiniéndolo (especialmente en la etapa de pruebas de comunicación), es el pórtico serial, para el cual existe la siguiente forma de etiquetarlo:

```
Inicio del programa...
....
....

Inicio de etiquetas

MODO0:  EQU     00H     ;Frecuencia de oscilación/12
MODO1:  EQU     40H     ;8 bits UART variable
MODO2:  EQU     80H     ;9 bits UART Fosc/64 o 32
MODO3:  EQU     0C0H    ;9 bits UART variable
SM2:    EQU     20H     ;Habilita RI
ENBRX:  EQU     10H     ;Habilita RX
NOVBITTX:       EQU     08H     ;Noveno bit a Tx
NOVBITRX:       EQU     04H     ;Noveno bit RX
BANDTX: EQU     02H     ;Bandera de interrupción en TX
BANDRX: EQU     01H     ;Bandera de interrupción en RX

.....
.....

Programa

.....
                LCALL   INIPSER ;Inicializo pórtico Serial.
.....

INIPSER:        MOV             DPTR,#SSCON
                MOVC            A,@A+DPTR
                MOV             SCON,A
                RET
SSCON:  DB              MODO3+SM2+ENBRX+NOVBITTX+BANDTX

.....
.....
```

Nótese que al final se programa el pórtico únicamente sumando o quitando etiquetas, lo que ahorra tiempo y esfuerzo.

Esta metodología de programación de pórticos se vuelve muy versátil cuando en un programa se tiene varios pórticos a programar, evitando confusiones y errores en la programación.

3.4 PROGRAMA DEL MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS MICROCONTROLADORES DE LA FAMILIA MCS-51

El programa del módulo de desarrollo aunque es extenso es fácil de analizarlo, ya que esta realizado de manera modular, es decir cada función del programa tiene una localización específica para evitar confusiones; además la mayor parte del programa son subrutinas de aplicación general que realizan alguna función centralizada y luego regresan al programa principal, sin efectuar saltos de un lado a otro; así mismo tratando de facilitar el seguimiento y la comprensión del mismo.

3.4.1 Diagramas de flujo del programa.

A continuación presentamos una serie de diagramas de flujo del programa del módulo con el fin de visualizar cada parte de la que consta el programa y los saltos a las subrutinas que lo realizan.

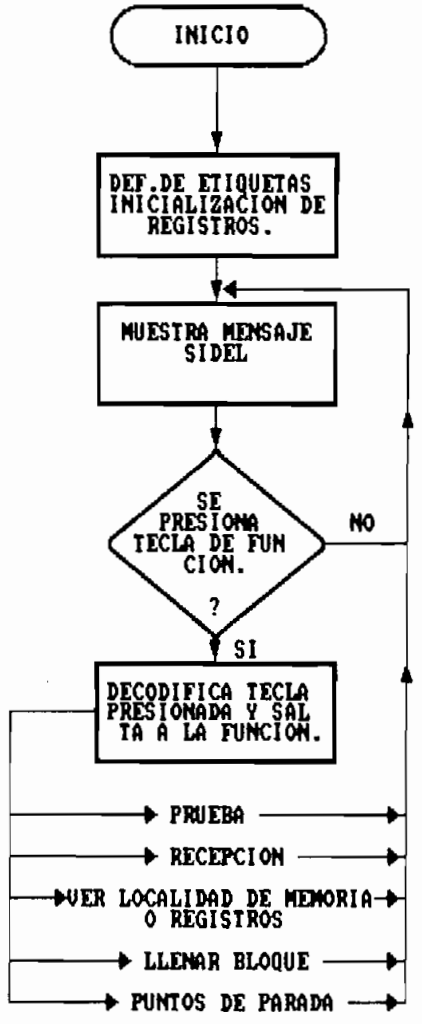
- Programa principal.

Aqui se definen todas las etiquetas ha usarse, se inicializan los registros principales del microcontrolador, se define al banco número tres de registros como el área de RAM a utilizarse; y a

continuación se presenta en el display el mensaje "SIDEL" como señal que el módulo esta en espera de una tecla de función. Mientras no se presione una tecla el módulo permanecerá en este estado.

Al presionar una tecla de función el módulo de desarrollo salta a ejecutar la subrutina pedida. Una vez terminada esta se vuelve al estado inicial.

PROGRAMA PRINCIPAL.



- Función cero (FUNC0, tecla # 1)

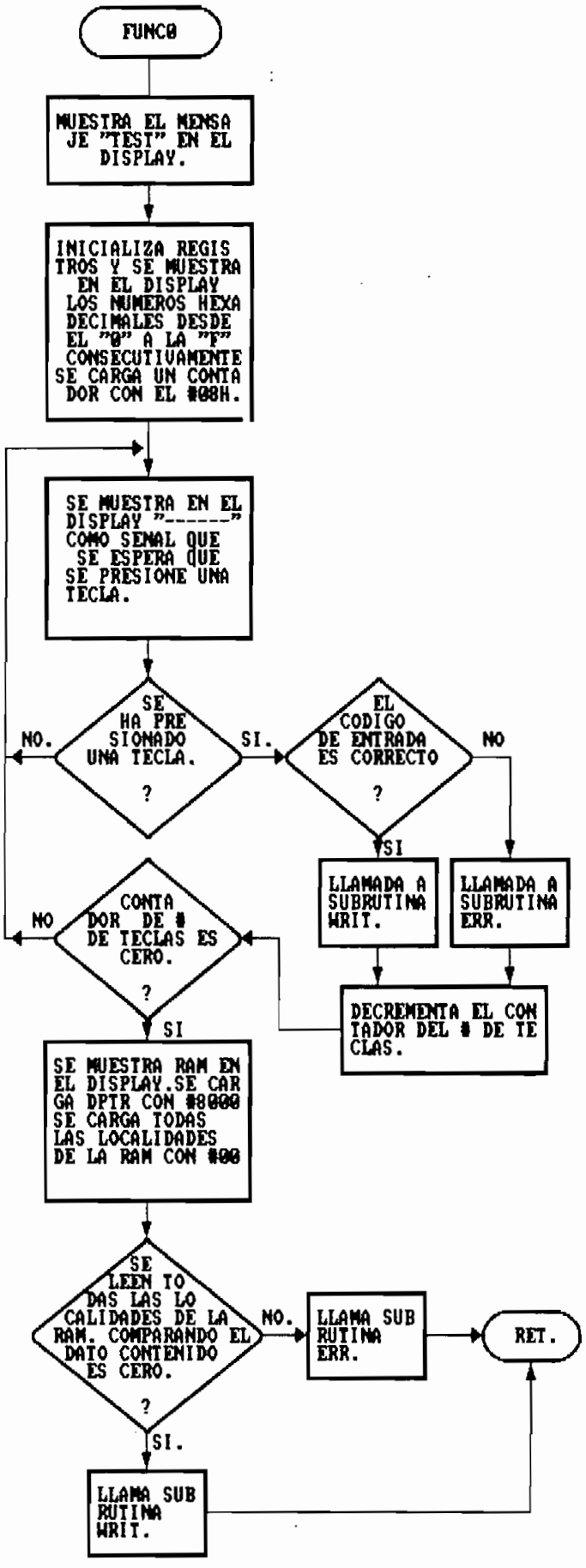
  Prueba del display, teclado, y RAMs externa.

Al presionar la tecla número uno (tecla menos significativa), se ingresa a la función cero del programa del módulo (FUNC0), esta función realiza una prueba de los componentes principales del circuito como son el display, el teclado, y la RAM externa.

Para comprobar el buen funcionamiento del display, se presentan en los seis dígitos los números hexadecimales desde el "cero" hasta la "efe" consecutivamente; una vez que termina esta parte de la prueba, se presenta en el display el mensaje "------" como señal que se espera que se presione una tecla. Si al presionar una tecla el código de entrada es el correspondiente a uno de los códigos preestablecidos en la tabla de decodificación de teclas, se presenta en el display el mensaje de "OK.", caso contrario se muestra "Error".

Existe un contador precargado con el número 08h, de acuerdo al número de teclas que se tiene en el módulo, por tanto para comprobar el buen funcionamiento de todas las teclas se recomienda presionar una sola vez cada una de las ocho teclas. Terminada esta parte del programa se examina la RAM externa, para lo cual se escribe todas las localidades de memoria con el valor 00H, y a continuación se leen dichas localidades comprobando que en estas se encuentre el valor anterior.

PRUEBA DEL DISPLAY, TECLADO Y MEMORIA RAM.



Luego se escriben las localidades con el valor FFH, y nuevamente se comprueba que en todas las localidades este dicho valor.

Si esta prueba culmina con exito se presenta el mensaje de "OK." caso contrario el mensaje de "Error".

A continuación se realiza la misma prueba anterior pero en la SIM-EPROM.

Esta función utiliza los registros A, B, DPTR, R0, R1 Y R2 del banco número 3.

- Función uno (FUNC1, tecla # 2)

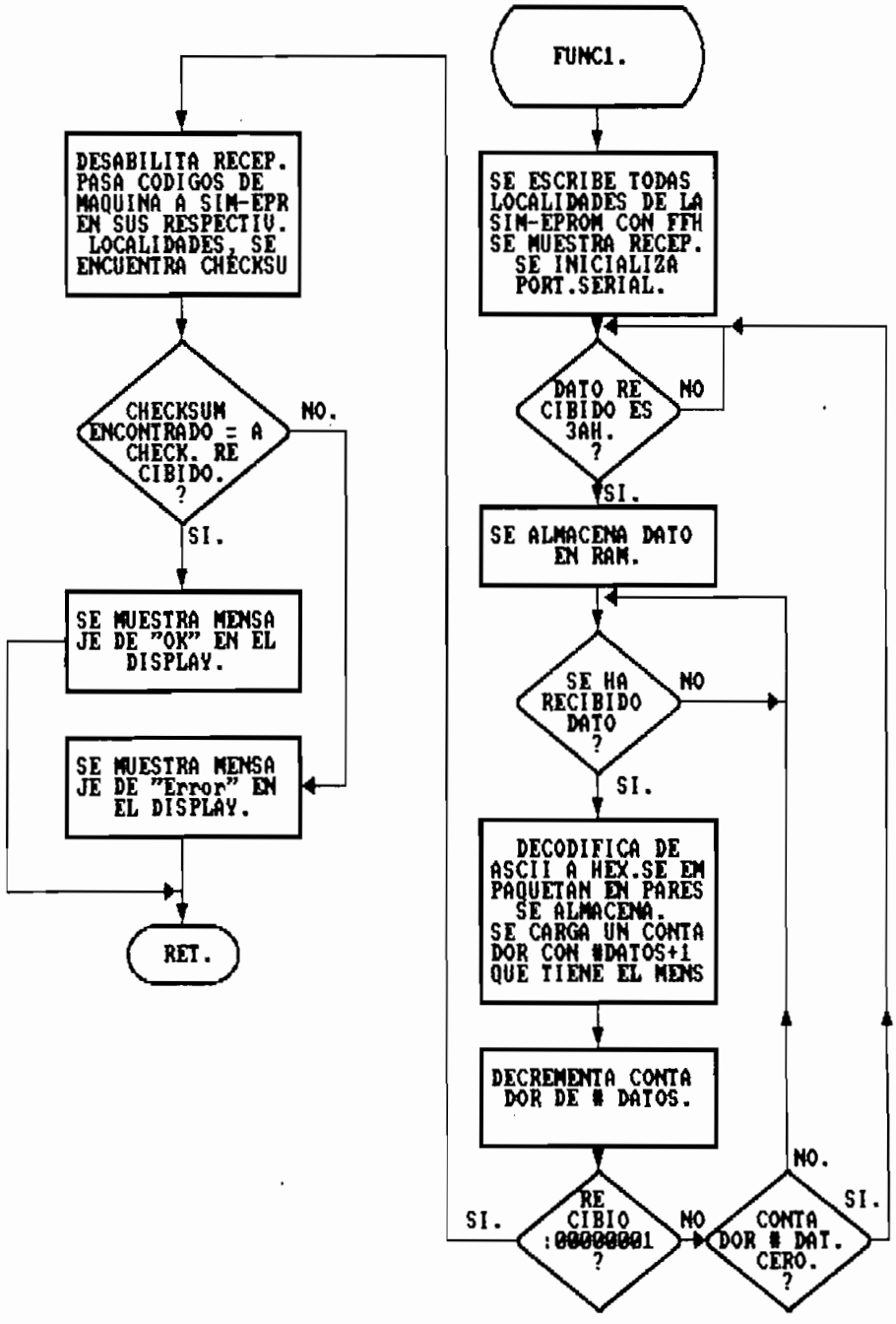
  Recepción serial.

La recepción serial desde el computador hacia el módulo de desarrollo se inicia al presionar la tecla número dos (FUNC1), en el display aparece el mensaje de "RECEP", y luego se blanquea la pantalla como señal que el módulo esta en el modo de recepción.

Esta función trabaja con los dos bancos de memoria, inicialmente escribe todas las localidades de la SIM-EPROM con el número FFH, luego entra en un lazo de espera hasta recibir la cabecera del mensaje trasmitido ":" (3AH), a continuación conforme van ingresando los datos se realiza la decodificación de ASCII a hexadecimal; el número de datos que van a ingresar en el mensaje se almacena en un contador para una vez que han ingresado todos los datos incluido el byte de control de errores (checksum), se entra nuevamente al lazo de espera por la cabecera de un

nuevo mensaje.

RECEPCION SERIAL



Todos los datos se van almacenando en la RAM, el modo de recepción termina cuando se recibe la secuencia de finalización :00000001FF.

Una vez que se termina la recepción serial, se pasa a decodificar los mensajes recibidos, los códigos de máquina trasmitidos se pasan de la RAM a la SIM-EPROM a su localidad correspondiente, previo chequeo que éstos no contengan errores. Una vez terminado este proceso se muestra en el display el mensaje de "OK".

Esta función usa los registros R0, R1, R2, R3, R4, R5, R6, R7, ACC, B, DPTR trabajando en el banco número tres de la RAM interna.

- Función dos (FUNC2, tecla # 3)

  Observar localidad de memoria, cambiar datos, insertar,borrar.

Esta función una vez que muestra en el display el mensaje "VER", espera por una tecla de selección de dirección de memoria, se puede escoger la RAM, SIM-EPROM, REGISTROS de uso general (RAM interna del microcontrolador), ó los registros de funciones especiales SFR.

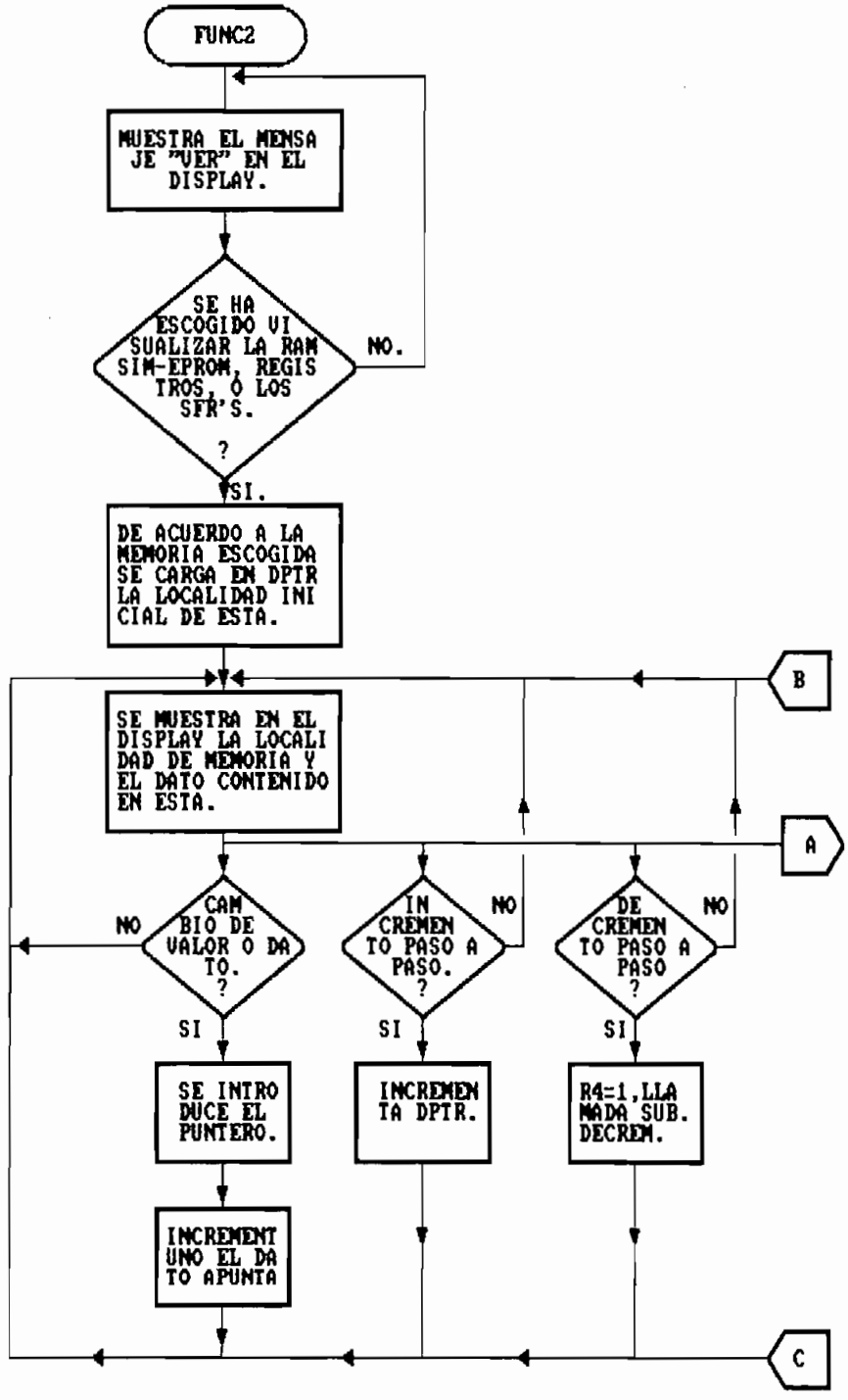
Escogida una de ellas, se muestra en el display la localidad de memoria y el dato contenido en dicha localidad o registro.

Una vez que se tiene en el display la localidad inicial de la memoria escogida y el dato contenido en ella, mediante el teclado se puede escoger entre:

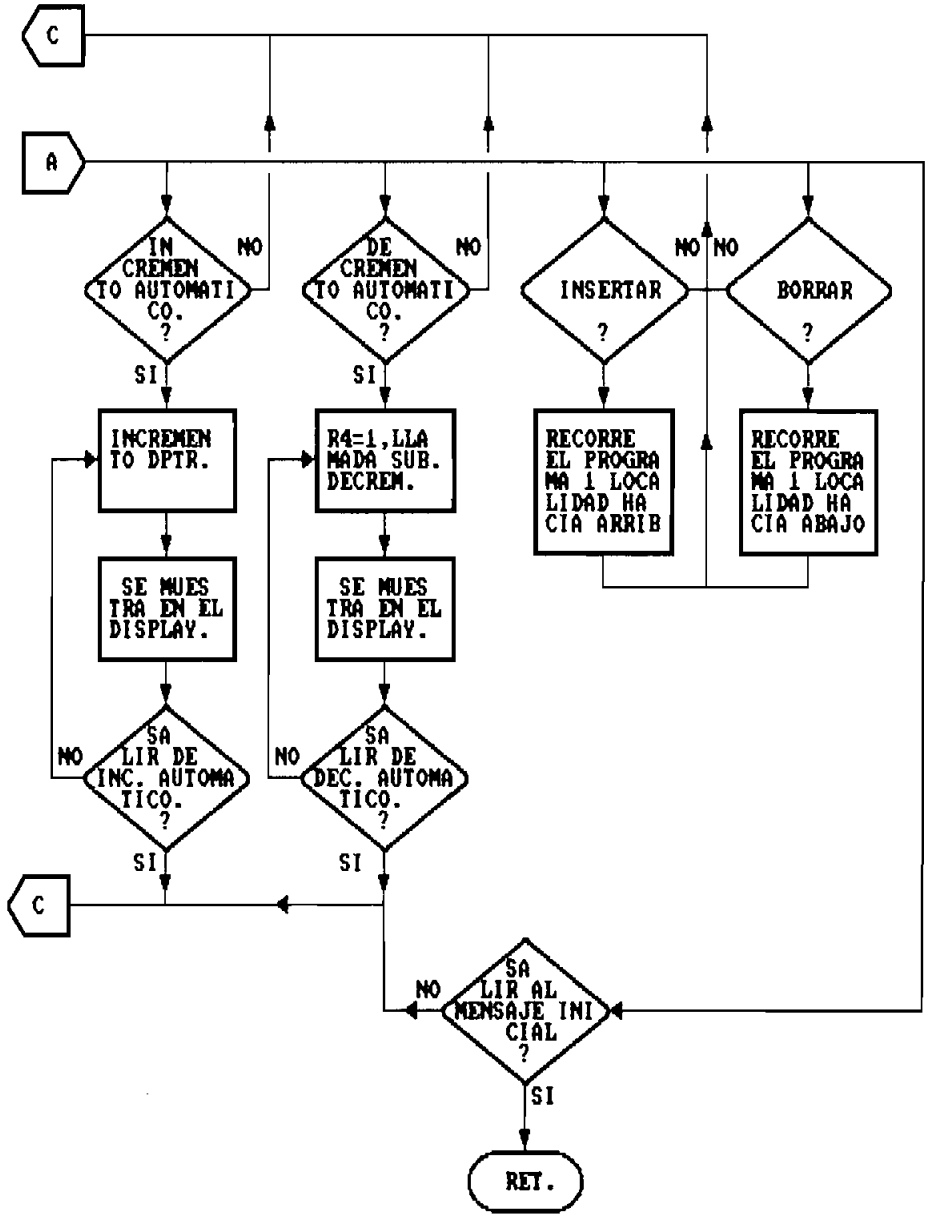
- Incrementar la localidad de memoria paso a paso.
- Decrementar la localidad de memoria paso a paso
- Incrementar la localidad de memoria automáticamente, es decir que esta incrementará su valor luego de un tiempo prudencial para poder observar el dato contenido.
- Decrementar la localidad automáticamente.
- Insertar una localidad de memoria, es decir el programa se traslada en una localidad hacia arriba; dejando la posibilidad de insertar un código de máquina.
- Borrar un código de máquina, es decir el programa se traslada una localidad hacia abajo.
- Cambio de la localidad a un valor definible por el usuario, o cambio del dato contenido en una localidad, para esto mediante una tecla se apunta en el display que dígito se quiere cambiar de valor (el dígito seleccionado se presenta con el punto decimal encendido), y luego presionando otra tecla se incrementa en uno el dígito seleccionado, este proceso se puede repetir hasta conseguir el valor deseado.

Esta función por su extensión y complejidad utiliza todos los registros de los bancos número dos y tres.

VER, CAMBIAR, INSERTAR O BORRAR LOCALIDAD DE MEMORIA.



VER, CAMBIAR, INSERTAR O BORRAR LOCALIDAD DE MEMORIA.
(CONTINUACION).



- Función tres (FUNC3, tecla # 4).

  Llenar un bloque con un valor determinado.

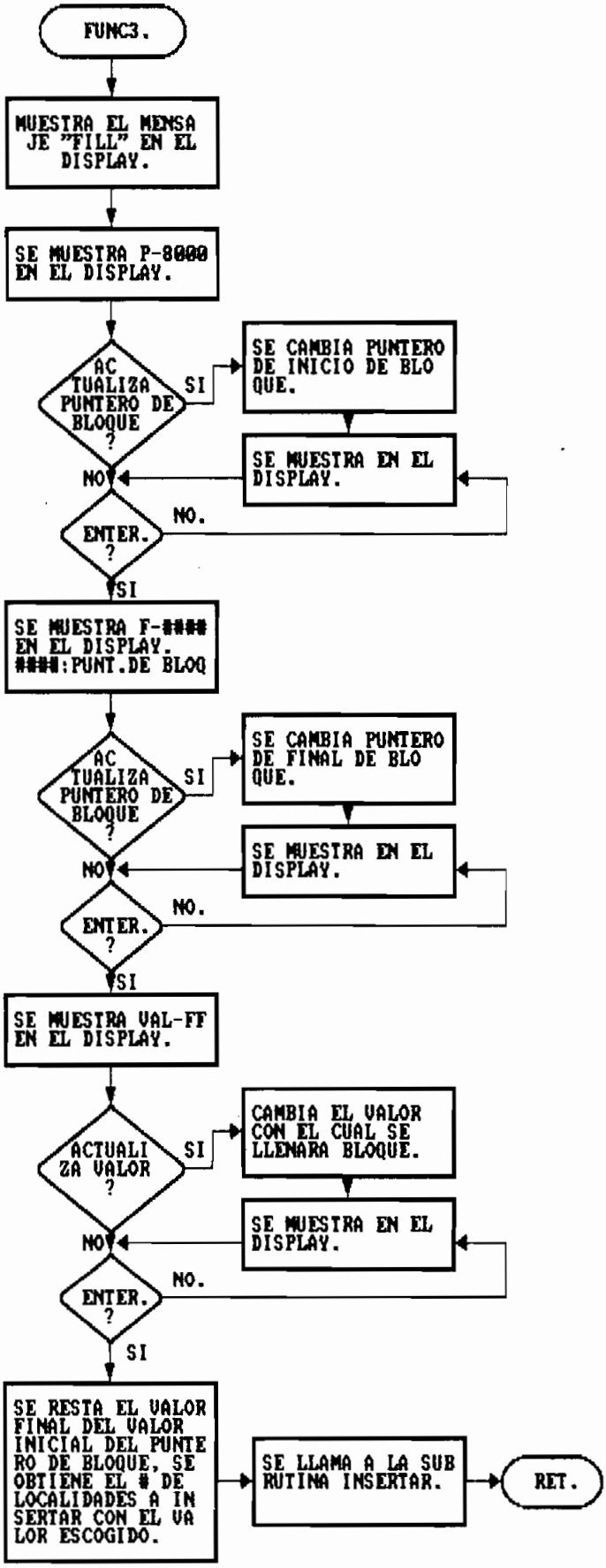
Al ingresar a esta función se muestra en el display el mensaje de FILL, a continuación se presenta el mensaje P-8000, como señal que el puntero de inicio del bloque se encuentra prefijado en la localidad 8000H, mediante el teclado se puede fijar el puntero de inicio del bloque en la localidad deseada; una vez escogido el inicio se presenta en el display F-8000 para de la misma manera anterior escoger el final del bloque, luego se presenta el mensaje VAL-FF, para escoger el valor con el cual se debe llenar el bloque,.

Para llenar el bloque se inserta el número de localidades correspondientes al valor final menos el valor inicial ingresado desde teclado para luego llenar este bloque con el valor escogido.

Esta función utiliza los registros R3, R4, R5, R6, R7, DPTR, ACC, y el registro B, del banco número tres de RAM interna del microcontrolador.

Una vez que se termina de llenar el bloque, se regresa al menú principal del programa.

LLENAR UN BLOQUE CON UN VALOR DETERMINADO.



- Función cuatro (FUNC4, tecla # 5).

  Puntos de parada.

Una vez que se ha escogido esta función se presenta en el displays el mensaje "Puntos".

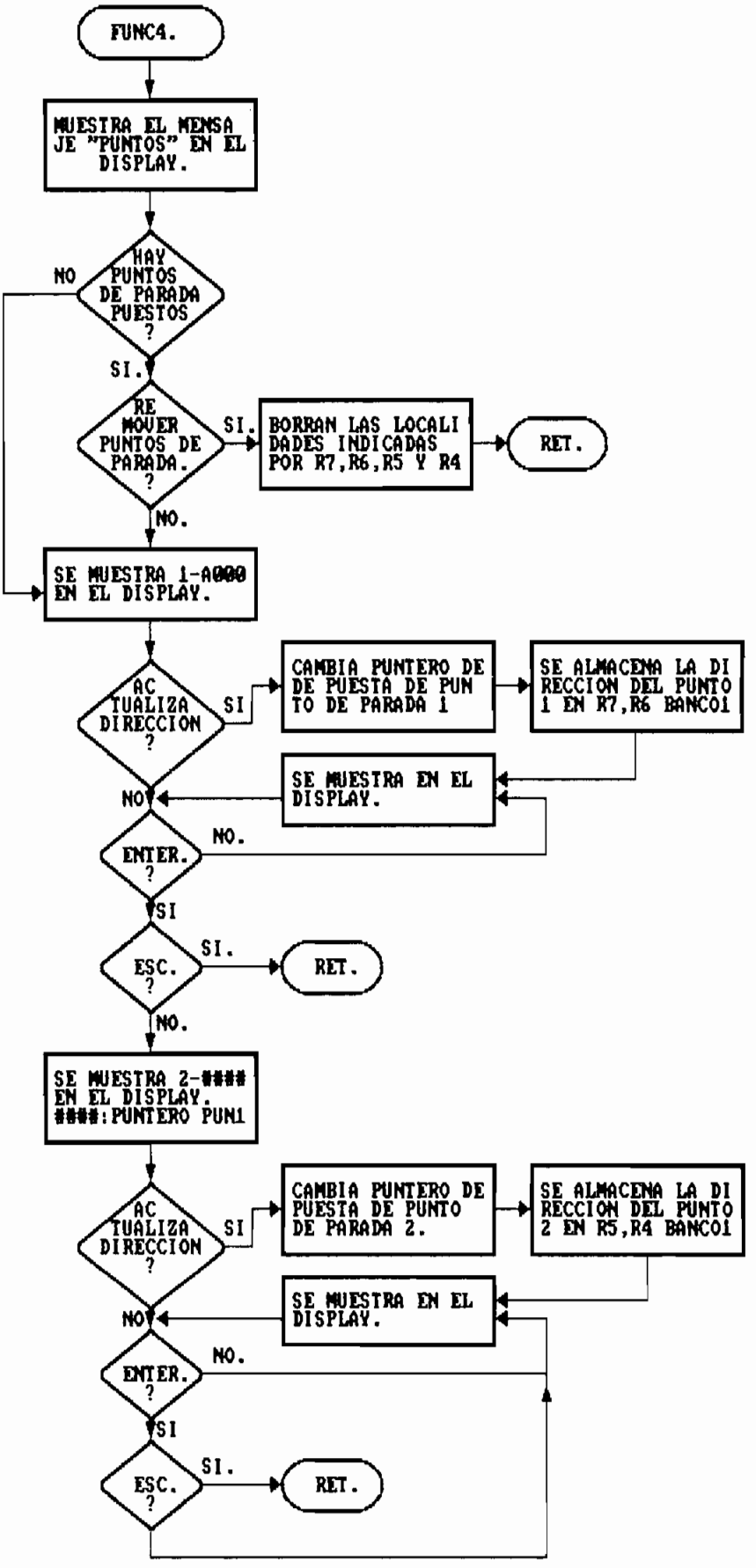
Se pueden poner un número indeterminado de puntos de parada en el programa, sin embargo se pueden remover automáticamente únicamente los dos últimos puntos de parada ingresados.

Por esta razón se recomienda colocar puntos de parada de dos en dos, es decir colocar dos puntos de parada si así fuera el caso, remover estos dos puntos de parada, y luego colocar otros dos puntos en localidades diferentes.

Cuando se tiene en el display el mensaje de 1-A000 , se puede colocar el primer puntos de parada en una localidad escogida desde teclado, luego se presenta 2-A002 , como señal que se espera por un segundo punto de parada, si no es necesario se puede salir al menú principal del programa, o caso contrario colocar este segundo punto de parada y salir.

Esta función utiliza los registros DPTR, B, ACC, R0, R3, R4, R5, R6, R7 del banco de registros tres, y R4, R5, R6, R7 del banco de registros uno.

COLOCAR PUNTOS DE PARADA.



- Subrutina Displays.

La Subrutina displays, hace un barrido de seis localidades de memoria interna, cada localidad de memoria representa un dígito del display.

Las localidades de memoria que usa esta subrutina son:

7AH: Dígito cero (dígito menos significativo).
7BH: Dígito uno
7CH: Dígito dos
7DH: Dígito tres
7EH: Dígito cuatro
7FH: Dígito cinco (dígito mas significativo).

El tiempo de barrido de estas localidades de memoria interna, esta determinado por el valor de dos contadores anidados R3 y R4, en donde el registro R4 es el contador interno o menos significativo, y R3 es el contador externo o mas significativo.

- Subrutina Coddis.

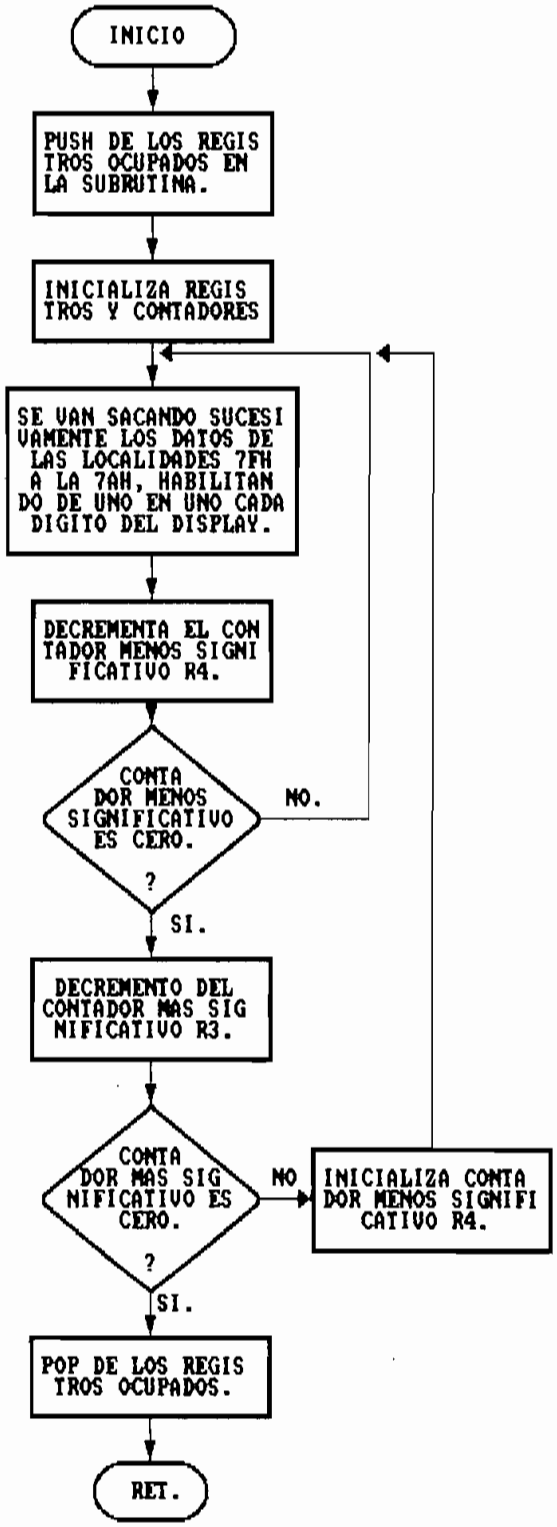
Para mostrar un dato contenido en una localidad de memoria, se necesitan dos dígitos del display.

Como la subrutina displays hace un barrido de las localidades de memoria 7AH hasta la 7FH en donde cada localidad representa un dígito del display; la subrutina Coddis parte de tres localidades de memoria codificándolas y almacenando los datos codificados en seis localidades, para que luego la subrutina displays se encargue de mostrar los datos.

Los datos que codifica esta subrutina se encuentran en la

localidad 7FH, 7DH, y 7BH; es decir se parte de la localidad 7FH, esta subrutina codifica el dato contenido en esta localidad almacenando los datos codificados para displays en las localidades 7FH y 7EH, luego la localidad 7DH, se codifica y se almacena en las localidades 7DH y 7CH, la 7BH, se codifica y almacena en la 7BH y 7AH; completando las seis localidades necesarias para cada dígito del display.

SUBRUTINA DISPLAYS.



- Subrutina Decrem.

Esta subrutina decrementa el puntero DPTR, el número de veces indicado por el registro R4.

- Subrutina Delay.

Esta subrutina introduce un pequeño retardo en el programa utilizando el timer1 del microcontrolador.
El tiempo de retardo esta determinado por el valor del registro "B", cuando B=1H el tiempo de retardo se encuentra en el orden de 15 msegundos.

- Subrutina Message.

Esta subrutina carga directamente al display seis códigos indicados por el registro DPTR.
Los códigos cargados en el display deben estar previamente codificados para mostrarse en los seis dígitos del display. Esta subrutina generalmente se utiliza para mostrar un mensaje en el display.
Una tabla de códigos, y su respectiva salida en el display se muestra en la tabla 3.6.

- Subrutina Writ.

Esta subrutina escribe en el display el mensaje de "OK."

- Subrutina Err.

Esta subrutina escribe en el display el mensaje de "Error".

Tabla 3.6: Precodificación de datos para ser mostrados por la subrutina message.

PRECODIFICACION PARA SALIDA A DISPLAYS.									
DP	G	F	E	D	C	B	A	# HEX.	SALIDA
Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0		
0	1	1	0	1	1	0	1	6D	5
0	0	0	0	0	1	0	0	04	'
0	1	0	1	1	1	1	0	5E	d
0	1	1	1	1	0	0	1	79	E
0	0	0	0	0	1	1	0	06	!
0	1	0	1	0	0	0	0	50	r
0	1	0	1	1	1	0	0	5C	o
1	0	0	0	0	0	0	0	80	.
0	0	0	1	1	1	0	0	1C	u
0	1	1	1	1	0	0	0	78	t
0	1	0	0	0	0	0	0	40	-
0	1	1	1	0	0	1	1	73	P
0	0	1	1	1	1	1	1	3F	O
1	1	1	1	0	0	1	0	F2	ñ.

3.4.2 Programa.

```
;      PROGRAMA DEL MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;                  MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:       ESCUELA POLITECNICA NACIONAL
;                    FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:    ELECTRONICA Y TELECOMUNICACIONES
;                    WASHINGTON GIOVANNI SIDEL TORRES
;                              1992-1993
;
$ALLPUBLIC
;   Area de inicialización de puntos de entrada de interrupciones.
;
;ETIQUETA:       OPC      OPERANDOS        ;COMENTARIOS

INICIO:            ;Definición de etiquetas
         ;Defino las direcciones de entrada/salida

INOUT2: EQU      4000H    ;4000-5FFF DAC OUT/buff-drip IN
INOUT3: EQU      6000H    ;6000-7FFF ADC IN
INOUT4: EQU      8000H    ;8000-9FFF latch OUT/buffer IN
INOUT7: EQU      0E000H   ;E000-FFFF latch OUT/buffer IN

INIRAM: EQU      8000H    ;Inicio de la RAM
FINRAM: EQU      87FFH    ;Fin de la RAM (2Kb)
EPROM:  EQU      0A000H   ;Direc de inicio de la SIM-EPROM
ENDEPROM:EQU     0A7FFH   ;Fin de la sim-eprom (4kb)
STACK:  EQU      4FH      ;Dirección del Stack en 50h
SCREEN5:EQU      7FH      ;Pantalla mas significativa #5
SCREEN4:EQU      7EH      ;Pantalla # 4
SCREEN3:EQU      7DH      ;Pantalla # 3
SCREEN2:EQU      7CH      ;Pantalla # 2
SCREEN1:EQU      7BH      ;Pantalla # 1
SCREEN0:EQU      7AH      ;Pantalla # 0
SALHAB: EQU      0000H    ;Habilitación displays de salida
SALDAT: EQU      2000H    ;Salida de datos hacia Displays
TECLAS: EQU      0000H    ;Entrada de datos de las teclas
BAUD:   EQU      0F8H     ;Valor de recarga 2400 baudios
ADSFR:  EQU      B0H      ;Dirección de los SFR
ADREG:  EQU      00H      ;Dirección de los registros
BANK0:  EQU      00H      ;Banco de registros 00-07
BANK1:  EQU      08H      ;Banco de registros 08-0F
BANK2:  EQU      10H      ;Banco de registros 10-17
BANK3:  EQU      18H      ;Banco de registros 18-1F

;        Definición de parámetros para la recepción serial
;             SCON Serial port control register

MOD0:   EQU      00H      ;Fosc/12
MOD1:   EQU      40H      ;8 bits UART variable
MOD2:   EQU      80H      ;9 bits UART fosc/64 o 32
MOD3:   EQU      0C0H     ;9 bits UART variable
SSMM22: EQU      20H      ;Enable RI
ENRX:   EQU      10H      ;Enable serial comunications
```

```
NOVBIT: EQU     08H     ;Noveno bit a trasmitirse
NOVBRX: EQU     04H     ;Noveno bit recibido
BANDTX: EQU     02H     ;Bandera de interrupción TX.
BANDRX: EQU     01H     :Bandera de interrupción RX.

;               TCON Timer counter control register

TTCON7: EQU     80H     ;Bandera timer 1 overflow
TTCON6: EQU     40H     ;On/Off timer 1
TTCON5: EQU     20H     ;Bandera timer 0 overflow
TTCON4: EQU     10H     ;On/Off timer 0
TTCON3: EQU     08H     ;Bandera de interrupción ext 1
TTCON2: EQU     04H     ;Control de interrupción ext 1
TTCON1: EQU     02H     ;Bandera de interrupción ext 0
TTCON0: EQU     01H     ;Control de interrupción ext 0

;               TMOD Timer/Counter mode control register

GGAT1:  EQU     80H     ;Control timer 1
CCTT1:  EQU     40h     ;Timer/counter selector 1
M1M001: EQU     00H     ;5 bits
M1M011: EQU     10H     ;16 bits
M1M021: EQU     20H     ;8 bits recargables
M1M031: EQU     30H     ;Control cascada timers
GGAT0:  EQU     08H     ;Control timer 0
CCTT0:  EQU     04H     ;Timer/Counter 0
M1M000: EQU     00H     ;5 bits 0
M1M010: EQU     01h     ;16 bits 0
M1M020: EQU     02H     ;8 Bits recargables 0
M1M030: EQU     03H     ;Control cascada timers 0

          ; Programa Principal

        MOV     SP,#STACK       ;Inicializar el Stack en 50h
        MOV     PSW,#BANK3      ;Cambiar al banco # 3.
        MOV     B,#06H          ;Encerar las localidades de
        MOV     A,#00H          ;memoria desde la 7fh(mas sig)
        MOV     R0,#SCREEN5     ;hasta la 7ah (menos sig)
INZERO: MOV     @R0,A
        DEC     R0
        DJNZ    B,INZERO

;    Presentar el nombre SIDEL. como señal de que el módulo esta en espera.

SHOW:   MOV     DPTR,#SIDEL     ;Cargar la dirección del nom
        LCALL   MESSAGE         ;bre Sidel y se muestra.

;     Barrido de las teclas de función, mientras no se aplaste
;       una de ellas el display seguirá mostrando el nombre SIDEL.

        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        CJNE    A,#0FFH,DECTECLA
        SJMP    SHOW
```

```
;           Decodificación de la tecla presionada

DECTECLA:CJNE   A,#0FEH,G00
         LCALL  FUNC0           ;Tecla men-sig TEST
         SJMP   SHOW
G00:     CJNE   A,#0FDH,G01
         LCALL  FUNC1           ;Recepción Serial
         SJMP   SHOW
G01:     CJNE   A,#0FBH,G02
         LCALL  FUNC2           ;Ver localidad de memoria
         SJMP   SHOW            ;insertar,borrar.
G02:     CJNE   A,#0F7H,G03     ;Fill
         LCALL  FUNC3
         SJMP   SHOW
G03:     CJNE   A,#0EFH,G04     ;Puntos de parada.
         LCALL  FUNC4
         SJMP   SHOW
G04:     CJNE   A,#0DFH,G05
         CALL   FUNC5
         SJMP   SHOW
G05:     CJNE   A,#0BFH,G06
         LCALL  FUNC6
         SJMP   SHOW
G06:     CJNE   A,#07FH,G07
         LCALL  FUNC7
         SJMP   SHOW
G07:     LCALL  ERR             ;muestra el mensaje de ERROR
         SJMP   SHOW

;  Subrutina que realiza un test de los displays y el teclado del módulo de
;           desarrollo para comprobar su buen funcionamiento

FUNC0:  MOV     B,#03H
OTHER1: MOV     DPTR,#TEST     ;Escribir la palabra TEST.
        LCALL   MESSAGE
        DJNZ    B,OTHER1       ;Llamar a que muestre en los displays.
        CLR     A
        MOV     DPTR,#NUMBERS
        MOV     R1,#10H        ;Contador del # de digitos hex.
NEWNUM: MOV     B,#06H         ;Contador del # de displays
        MOV     R0,#SCREEN5
        MOVC    A,@A+DPTR      ;Se carga los números hexa
NEWDIS: MOV     @R0,A          ;decimales desde el uno hasta la
        DEC     R0             ;efe en los displays
        DJNZ    B,NEWDIS
        LCALL   DISPLAYS
        INC     DPTR
        CLR     A
        DJNZ    R1,NEWNUM
        MOV     R1,#08H        ;Contador de número de teclas
LOAD1:  MOV     A,#40H         ;Cargar en el display "-------"
        MOV     R0,#SCREEN5    ;(40h), como señal de acabada la
        MOV     B,#06H         ;prueba de displays.
PASS:   MOV     @R0,A
        DEC     R0
```

```
        DJNZ    B,PASS
        CALL    DISPLAYS        ;Fín de la prueba de displays

         ;A continuación se realiza una prueba de las teclas

        CLR     A
        MOV     DPTR,#TECLAS    ;Probar cada una de las teclas
        MOVX    A,@DPTR         ;si el código de entrada de la
        CJNE    A,#0FFH,WRITOK  ;tecla es el correcto se presen
        SJMP    LOAD1           ;ta en los displays el mensaje
WRITOK: CJNE    A,#0FEH,WRIT0   ;de --OK-- caso contrario se
        CALL    WRIT            ;presenta el mensaje de ERROR
        SJMP    FINTEC          ;esto se repite ocho veces para
WRIT0:  CJNE    A,#0FDH,WRIT1   ;para las ocho teclas
        CALL    WRIT
        SJMP    FINTEC
WRIT1:  CJNE    A,#0FBH,WRIT2
        CALL    WRIT
        SJMP    FINTEC
WRIT2:  CJNE    A,#0F7H,WRIT3
        CALL    WRIT
        SJMP    FINTEC
WRIT3:  CJNE    A,#0EFH,WRIT4
        CALL    WRIT
        SJMP    FINTEC
WRIT4:  CJNE    A,#0DFH,WRIT5
        CALL    WRIT
        SJMP    FINTEC
WRIT5:  CJNE    A,#0BFH,WRIT6
        CALL    WRIT
        SJMP    FINTEC
WRIT6:  CJNE    A,#07FH,WRIT7
        CALL    WRIT
        SJMP    FINTEC
WRIT7:  LCALL   ERR
FINTEC: DJNZ    R1,LOAD1        ;Fín del test de las teclas.

         ; Realiza una prueba de la RAM y de la SIM-EPROM.

        CLR     71H
NEWTEST:MOV     B,#04H
OTHER2: JB      71H,TESEPRO
        MOV     DPTR,#RAMTEST   ;Escribir la palabra RAM
        SJMP    OTHER9          ;y se muestra en los displays..
TESEPRO:MOV     DPTR,#SIMEPRO
OTHER9: LCALL   MESSAGE
        DJNZ    B,OTHER2

        JB      71H,OTHER4
        MOV     DPTR,#INIRAM
        MOV     R1,#88H         ;cargar 2K. en los registros
        SJMP    OTHER5          ;07FFH.
OTHER4: MOV     DPTR,#EPROM
        MOV     R1,#0A8H
OTHER5: MOV     R2,#00H
```

```
PROXCEL:MOV     A,#00H          ;Cargar el ACC con 00h y
        MOVX    @DPTR,A         ;escribir en todas las
        INC     DPTR            ;celdas éste número
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,PROXCEL
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,PROXCEL

COMPBAR:JB      71H,OTHER6
        MOV     DPTR,#INIRAM
        SJMP    NEXTC1
OTHER6: MOV     DPTR,#EPROM     ;comprobar que en todas
NEXTC1: CLR     C               ;las celdas de memoria esté
        MOVX    A,@DPTR         ;el #00h, si en
        CJNE    A,#00H,BAD      ;todas las celdas se encuentra
        INC     DPTR            ;el #00h se completa esta parte
        CLR     C               ;del test caso contrario se
        MOV     A,DPH           ;muestra el mensaje de ERROR.
        SUBB    A,R1
        CJNE    A,#00H,NEXTC1
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXTC1
        JMP     COMP1
BAD:    LCALL   ERR
        JMP     ENDRAM

COMP1:  JB      71H,OTHER7
        MOV     DPTR,#INIRAM    ;escribir todas las celdas
        SJMP    PROX            ;con el número ffh.
OTHER7: MOV     DPTR,#EPROM
PROX:   MOV     A,#0FFH
        MOVX    @DPTR,A
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXT22
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXT22
        JMP     COMP22
NEXT22: SJMP    PROX
COMP22: JB      71H,OTHER8
        MOV     DPTR,#INIRAM    ;comprobar que en todas las cel
        SJMP    NEXTC2          ;das se encuentre el #ffh.
OTHER8: MOV     DPTR,#EPROM
NEXTC2: MOVX    A,@DPTR
        CJNE    A,#0FFH,BAD
```

```
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXTC2
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXTC2
        LCALL   WRIT            ;Si todo esta bien se muestra
        JB      71H,ENDRAM      ;el mensaje "OK"
        SETB    71H
        JMP     NEWTEST
ENDRAM: RET

;Subrutina que realiza la Recepción Serial desde el computador hacia el módulo
;                          de desarrollo

FUNC1:  MOV     DPTR,#EPROM     ;Escribe todas las celdas de la
        MOV     R1,#0A8H        ;sim-eprom con el número #ffh
        MOV     R2,#00H
PRX11:  MOV     A,#0FFH
        MOVX    @DPTR,A
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NE33
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NE33
        JMP     CP33
NE33:   SJMP    PRX11

CP33:   MOV     B,#03H
OTHER3: MOV     DPTR,#RECEP     ;Escribir la palabra RECEP.
        LCALL   MESSAGE         ;Llamar a que muestre en los displays.
        DJNZ    B,OTHER3
        LCALL   INITRX          ;inicializar el
        MOV     R3,#00H         ;pórtico serial
        MOV     R2,#00H         ;bandera de intercambio de nible
        MOV     R5,#00H         ;bandera de haber recibido el 3A
        MOV     DPTR,#INIRAM
WAIT1:  JNB     RI,WAIT1        ;Lazo de espera por un dato
        CLR     RI              ;Limpiar la bandera de RX
        MOV     A,SBUF          ;Cargar el dato al ACC
        CJNE    A,#3AH,ASC0     ;Comparar si el primer dato es 3A
        MOV     R5,#01H         ;e inicializar dos banderas
        MOV     R4,#01H
        JMP     LOAD            ;cargar a la memoria RAM
ASC0:   CJNE    R5,#00H,ASC1
        SJMP    WAIT1
ASC1:   CJNE    A,#30H,ASC2     ;a continuación se hace la transfor
        MOV     A,#00H          ;mación de ASCII a HEX.
```

```
        SJMP    LOAD
ASC2:   CJNE    A,#31H,ASC3     ;Cuando se recibe los dos puntos
        MOV     A,#01H          ;":" (3AH)se comienza el almace
        SJMP    LOAD            ;namiento de datos en la RAM,
ASC3:   CJNE    A,#32H,ASC4     ;previo la transformación del
        MOV     A,#02H          ;formato ASCII al formato HEX de
        SJMP    LOAD            ;acuerdo a la siguiente tabla:
ASC4:   CJNE    A,#33H,ASC5     ;      ASCII    HEX
        MOV     A,#03H          ;       30      00h
        SJMP    LOAD            ;       31      01h
ASC5:   CJNE    A,#34H,ASC6     ;       32      02h
        MOV     A,#04H          ;       33      03h
        SJMP    LOAD            ;       34      04h
ASC6:   CJNE    A,#35H,ASC7     ;       35      05h
        MOV     A,#05H          ;       36      06h
        SJMP    LOAD            ;       37      07h
ASC7:   CJNE    A,#36H,ASC8     ;       38      08h
        MOV     A,#06H          ;       39      09h
        SJMP    LOAD            ;       41      0Ah
ASC8:   CJNE    A,#37H,ASC9     ;       42      0Bh
        MOV     A,#07H          ;       43      0Ch
        SJMP    LOAD            ;       44      0Dh
ASC9:   CJNE    A,#38H,ASC10    ;       45      0Eh
        MOV     A,#08H          ;       46      0Fh
        SJMP    LOAD
ASC10:  CJNE    A,#39H,ASC11
        MOV     A,#09H
        SJMP    LOAD
ASC11:  CJNE    A,#41H,ASC12
        MOV     A,#0AH
        SJMP    LOAD
ASC12:  CJNE    A,#42H,ASC13
        MOV     A,#0BH
        SJMP    LOAD
ASC13:  CJNE    A,#43H,ASC14
        MOV     A,#0CH
        SJMP    LOAD
ASC14:  CJNE    A,#44H,ASC15
        MOV     A,#0DH
        SJMP    LOAD
ASC15:  CJNE    A,#45H,ASC16
        MOV     A,#0EH
        SJMP    LOAD
ASC16:  CJNE    A,#46H,ASC17
        MOV     A,#0FH
        SJMP    LOAD
ASC17:  JMP     WAIT1

LOAD:   CJNE    A,#3AH,LOAD0    ;A continuación y mientras
        SJMP    LOAD2           ;se van almacenando los datos
LOAD0:  CJNE    R2,#00H,LOAD3   ;se hace la comparación para
        SWAP    A               ;que cuando se haya recibido
        MOV     R0,A            ;la secuencia ":00"
        MOV     R2,#01H         ;se termine la recepción del
        JMP     WAIT1           ;buffer serial.
```

```
LOAD3:  ANL     A,#0FH
        ORL     A,R0
        MOV     R2,#00H
        CJNE    R4,#01H,LOAD2
        MOV     R4,#00H
        CJNE    A,#00H,LOAD2
        SJMP    ENDRECEP
LOAD2:  MOVX    @DPTR,A
        INC     DPTR
        JMP     WAIT1

ENDRECEP:       CLR     A       ;Deshabilitar el puerto Serial
        MOV     SCON,A
        MOV     DPTR,#INIRAM    ;Hacer la decodificación del
SIG:    MOV     R7,#00H         ;formato INTEL de Ocho bits
        MOVX    A,@DPTR         ;se comprueba la validez
        CJNE    A,#3AH,ER1      ;de los datos mediante el "CHEKSUM" y
        SJMP    A11             ;se almacena los datos hexadeci
ER1:    LCALL   ERR             ;males puros en la sim-eprom
        JMP     ENDINT
A11:    INC     DPTR
        MOVX    A,@DPTR
        MOV     R6,A
        ADD     A,R7
        MOV     R7,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R5,A            ;R5 es el dph para el paso
        ADD     A,R7            ;a la sim-eprom
        MOV     R7,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R4,A            ;R4 es el dpl para el paso
        ADD     A,R7
        MOV     R7,A
        INC     DPTR
        MOVX    A,@DPTR
        CJNE    A,#00H,ENDINT
        INC     DPTR
CONTINU:MOVX    A,@DPTR
        MOV     R3,A
        ADD     A,R7
        MOV     R7,A
        PUSH    DPH
        PUSH    DPL
        MOV     DPH,R5
        ORL     DPH,#0A0H
        MOV     DPL,R4
        MOV     A,R3
        MOVX    @DPTR,A
        INC     DPTR
        MOV     R5,DPH
        MOV     R4,DPL
        POP     DPL
        POP     DPH
```

```
        INC     DPTR
        DJNZ    R6,CONTINU
        MOV     A,R7
        CPL     A
        ADD     A,#01H
        MOV     R6,A
        MOVX    A,@DPTR
        CLR     C
        SUBB    A,R6
        CJNE    A,#00H,ENDINT
        INC     DPTR
        JMP     SIG
ENDINT: LCALL   WRIT            ;Escribir la palabra OK. y ter
        RET                     ;minar la recepción

; Subrutina que permite visualizar el contenido de una localidad de memoria

FUNC2:  MOV     DPTR,#VERCEL    ;Escribir la palabra "VER" y
        LCALL   MESSAGE         ;espera por una tecla de función
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        CLR     7FH             ;Bandera de SFRS
        CLR     7EH             ;Bandera de subida automática
        CLR     7DH             ;Bandera de bajada automática
        CLR     7CH             ;Bandera de RAM INTERNA
        CLR     7BH             ;Badera de ingreso al modo de cambio datos
        CLR     7AH
        MOV     PSW,#BANK2
        MOV     R0,#SCREEN0-1H
        MOV     PSW,#BANK3
        CJNE    A,#0FFH,VERLOC
        SJMP    FUNC2

VERLOC: CJNE    A,#0FEH,NEXT0
        SJMP    EIGHTH
NEXT0:  CJNE    A,#0FDH,NEXT1
        SJMP    A000H
NEXT1:  CJNE    A,#0FBH,NEXT2
        SJMP    REG
NEXT2:  CJNE    A,#0F7H,NEXT3
        SJMP    SFRS
NEXT3:  JMP     DECTE8

SFRS:   MOV     B,#04H
SFRS0:  MOV     DPTR,#SFR
        LCALL   MESSAGE
        DJNZ    B,SFRS0
        SETB    7CH
        SETB    7FH
        MOV     DPTR,#ADSFR
        SJMP    VER

REG:    MOV     B,#04H
REG0:   MOV     DPTR,#REGISTROS ;Escribir la palabra "REGIST"
        LCALL   MESSAGE         ;y se muestra en los displays.
```

```
        DJNZ    B,REG0
        MOV     DPTR,#ADREG
        SETB    7CH
        SJMP    VER

A000H:  MOV     B,#04H
A0000:  MOV     DPTR,#SIMEPRO   ;Escribir la palabra EPROM
        LCALL   MESSAGE         ;y se muestra en los displays.
        DJNZ    B,A0000
        MOV     DPTR,#EPROM
        SJMP    VER

EIGHTH: MOV     B,#04H
EIGHT0: MOV     DPTR,#RAMTEST   ;Escribir la palabra RAM
        LCALL   MESSAGE         ;y se muestra en los displays.
        DJNZ    B,EIGHT0
        MOV     DPTR,#INIRAM
VER:    JNB     7CH,EXTER
        MOV     R1,DPL
        MOV     DPH,#00H
        JB      7FH,SPECIAL
        SJMP    AHEAD
SPECIAL:CLR     73H
        LCALL   ESPFUNC
        JBC     0,INTER
        SJMP    INTER
AHEAD:  MOV     A,@R1
        SJMP    INTER
EXTER:  MOVX    A,@DPTR
INTER:  MOV     R0,#SCREEN5
        MOV     @R0,A
        MOV     A,DPL
        DEC     R0
        DEC     R0
        MOV     @R0,A
        MOV     A,DPH
        DEC     R0
        DEC     R0
        MOV     @R0,A
        LCALL   CODDIS
        JNB     7CH,VER1
        MOV     SCREEN0,#00H
        MOV     SCREEN1,#00H
VER1:   LCALL   DISPLAYS
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        CJNE    A,#0FFH,DECTE
        JB      7EH,ONEUP       ;Saltar al incremento automático
        JB      7DH,ONEDOWN     ;Saltar al decremento automático
        MOV     PSW,#BANK2
        SJMP    COMP
```

```
LOOK9:  JMP     LOOK5
COMP:   JB      7BH,LOOK9
        MOV     PSW,#BANK3
        JMP     VER

;    A continuación se realiza la decodificación de la tecla presionada

DECTE:  CJNE    A,#7FH,LOOK     ;tec#8 subida paso a paso
        CLR     7EH
        CLR     7DH
        CLR     7BH
        MOV     PSW,#BANK2
        MOV     R0,#SCREEN0-1H
        MOV     PSW,#BANK3
ONEUP:  INC     DPTR            ;Carga la próxima localidad
        JNB     7FH,SALT0
        PUSH    ACC
        MOV     A,DPH
        CJNE    A,#01H,SALT1
        MOV     DPTR,#ADSFR
SALT1:  POP     ACC
SALT0:  JMP     VER             ;push la tecla mas sign #8

LOOK:   CJNE    A,#0BFH,LOOK1   ;tec#7 bajada paso a paso
        CLR     7EH
        CLR     7DH
        CLR     7BH
        MOV     PSW,#BANK2
        MOV     R0,#SCREEN0-1H
        MOV     PSW,#BANK3
ONEDOWN:MOV     R4,#01H
        LCALL   DECREM          ;Carga la localidad anterior
        JNB     7FH,SALT0
        PUSH    ACC
        MOV     A,DPL
        CJNE    A,#7FH,SALT2
        MOV     DPTR,#0FFH
SALT2:  POP     ACC
SALT0:  JMP     VER             ;push la tecla #7

LOOK1:  CJNE    A,#0FEH,LOOK2   ;tec#1 subida automática
        SETB    7EH
        JMP     VER

LOOK2:  CJNE    A,#0FDH,LOOK3   ;tec#2 bajada automática
        CLR     7EH
        SETB    7DH
        CLR     7BH
        JMP     VER

;Con la tecla #6 se cambia los datos contenidos en la memoria RAM externa
;o interna.

LOOK3:  CJNE    A,#0DFH,LOOK4   ;Tec#6 cambio de datos
        SETB    7AH
```

```
        SETB    7BH
        MOV     PSW,#BANK2
        JNB     7AH,LOOK5
        INC     R0              ;R0 contiene la dirección del
        CJNE    R0,#80H,LOOK5   ;display que no puede ser mayor
        DEC     R0              ; a 7fh
        MOV     A,@R0
        ANL     A,#7FH
        MOV     @R0,A
        MOV     R0,#SCREEN0-1H
LOOK5:  MOV     A,@R0
        ORL     A,#80H          ;se introduce el punto decimal
        MOV     @R0,A           ;para saber en que display se
        DEC     R0              ;va a hacer el cambio
        MOV     A,@R0
        ANL     A,#7FH
        MOV     @R0,A
        INC     R0
        MOV     PSW,#BANK3
        JMP     VER1            ;saltar a mostrar en los displays

DEC8:   JMP     DECTE8          ;línea de salto auxiliar.
LOOK4:  CJNE    A,#0FBH,DEC8    ;Si se presiona la Tec#3 se incrementa
        MOV     PSW,#BANK2      ;el valor señalado por el punto
        CJNE    R0,#7AH,CHAN0   ;decimal en el display
        MOV     A,DPH
        CLR     C
        ADD     A,#10H
        MOV     DPH,A
        MOV     PSW,#BANK3
        JMP     VER
CHAN0:  CJNE    R0,#7BH,CHAN1
        MOV     A,DPH
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPH,A
        MOV     PSW,#BANK3
        JMP     VER
CHAN1:  CJNE    R0,#7CH,CHAN2
        MOV     A,DPL
        CLR     C
        ADD     A,#10H
        MOV     DPL,A
        MOV     PSW,#BANK3
        JMP     VER
CHAN2:  CJNE    R0,#7DH,CHAN3
        MOV     A, DPL
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
```

```
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPL,A
        MOV     PSW,#BANK3
        JMP     VER

CHAN3:  CJNE    R0,#7EH,CHAN4
        JNB     7CH,EXTE11
        MOV     PSW,#BANK3
        MOV     R1,DPL
        JNB     7FH,NOESP
        CLR     73H
        LCALL   ESPFUNC
        JBC     0,INTER11
NOESP:  MOV     A,@R1
        MOV     PSW,#BANK2
        SJMP    INTER11
EXTE11: MOVX    A,@DPTR
INTER11:CLR     C
        ADD     A,#10H
        JNB     7CH,EXTE22
        MOV     PSW,#BANK3
        MOV     R1,DPL
        JNB     7FH,NOESP1
        SETB    73H
        LCALL   ESPFUNC
        JBC     0,SMJUM
NOESP1: MOV     @R1,A
SMJUM:  MOV     PSW,#BANK2
        SJMP    INTER22
EXTE22: MOVX    @DPTR,A
INTER22:MOV     PSW,#BANK3
        JMP     VER

CHAN4:  CJNE    R0,#7FH,CHAN5
        JNB     7CH,EXTE33
        MOV     PSW,#BANK3
        MOV     R1,DPL
        JNB     7FH,NOESP2
        CLR     73H
        LCALL   ESPFUNC
        JBC     0,INTER33
NOESP2: MOV     A,@R1
        MOV     PSW,#BANK2
        SJMP    INTER33
EXTE33: MOVX    A,@DPTR
INTER33:CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
```

```
        JNB     7CH,EXTE44
        MOV     PSW,#BANK3
        MOV     R1,DPL
        JNB     7FH,NOESP3
        SETB    73H
        LCALL   ESPFUNC
        JBC     0,SMJUM1
NOESP3: MOV     @R1,A
SMJUM1: MOV     PSW,#BANK2
        SJMP    INTER44
EXTE44: MOVX    @DPTR,A
INTER44:MOV     PSW,#BANK3
CHAN5:  JMP     VER

            ;Inserción de una localidad de memoria Tec#4

DECTE8: CJNE    A,#0F7H,LOOK6
        PUSH    DPH
        PUSH    DPL
        PUSH    B
        MOV     B,#04H
INSER0: MOV     DPTR,#INSERTE
        LCALL   MESSAGE
        DJNZ    B,INSER0
        POP     B
        POP     DPL
        POP     DPH
        CLR     76H
FILBLOQ:PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    19H
        PUSH    1AH
        PUSH    1CH
        PUSH    1DH
        MOV     R1,#08H
        MOV     R5,#00H
        MOVX    A,@DPTR
        MOV     R4,A
INSER:  INC     DPTR
        MOVX    A,@DPTR
        MOV     R2,A
        MOV     A,R4
        MOVX    @DPTR,A
        MOV     A,R2
        MOV     R4,A
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CLR     C
        ANL     A,#0FH
        CJNE    A,#00H,INSER
        MOV     A,DPL
        CLR     C
        SUBB    A,R5
```

```
        CJNE    A,#00H,INSER
        POP     1DH
        POP     1CH
        POP     1AH
        POP     19H
        POP     DPL
        POP     DPH
        JB      76H,BLOQ
        MOV     A,#0FFH
        SJMP    NOBLOQ
BLOQ:   MOV     A,R6
NOBLOQ: MOVX    @DPTR,A
        POP     ACC
        JB      76H,BLOQ1
        JMP     VER
BLOQ1:  RET

                ;Borrado de una localidad de memoria

LOOK6:  CJNE    A,#0EFH,LOOK7
        CLR     77H
        PUSH    B
        PUSH    DPH
        PUSH    DPL
        MOV     B,#04H
DEL0:   MOV     DPTR,#DEL
        LCALL   MESSAGE
        DJNZ    B,DEL0
        POP     DPL
        POP     DPH
        POP     B
DELPUNTS:PUSH   ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    19H
        PUSH    1DH
        PUSH    0FH
        PUSH    0EH
        PUSH    0DH
        PUSH    0CH
        PUSH    0BH
        MOV     R1,#08H
        MOV     R5,#00H
DELETE: INC     DPTR
        MOVX    A,@DPTR
        MOV     R4,#01H
        LCALL   DECREM
        MOVX    @DPTR,A
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CLR     C
        ANL     A,#0FH
        CJNE    A,#00H,DELETE
```

```
        MOV     A,DPL
        CLR     C
        SUBB    A,R5
        CJNE    A,#00H,DELETE
        POP     0BH
        POP     0CH
        POP     0DH
        POP     0EH
        POP     0FH
        POP     1DH
        POP     19H
        POP     DPL
        POP     DPH
        POP     ACC
        JB      77H,LOOK7
        JMP     VER

LOOK7:  RET

;Subrutina que permite llenar un bloque con un valor determinado, insertar o
;                       eliminar una localidad de memoria

FUNC3:  MOV     B,#04H
FUNC30: MOV     DPTR,#FILL      ;Escribir la palabra FILL y
        LCALL   MESSAGE         ;esperar por una tecla de función
        DJNZ    B,FUNC30
        CLR     79H
        CLR     7DH
        CLR     78H
        CLR     75H
        CLR     76H
        MOV     DPTR,#INIRAM

FIL00:  MOV     R0,#SCREEN5
        MOV     A,DPL
        MOV     @R0,A
        MOV     A,DPH
        DEC     R0
        DEC     R0
        MOV     @R0,A
        LCALL   CODDIS

        JNB     7DH,SHOWDAT
        MOV     SCREEN0,#1CH
        MOV     SCREEN1,#77H
        MOV     SCREEN2,#06H
        MOV     SCREEN3,#40H
        SETB    78H
        SJMP    SHOWVAL
SHOWDAT:JB      79H,SHOWF
        MOV     SCREEN0,#73H
        SJMP    SHOW40
SHOWF:  MOV     SCREEN0,#71H
SHOW40: MOV     SCREEN1,#40H
SHOWVAL:LCALL   DISPLAYS
```

```
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        CJNE    A,#0FFH,CHANGE
        JMP     FIL00

CHANGE: JB      78H,CHANGE1
        CJNE    A,#0EFH,CHANGE0
        MOV     A,DPH
        CLR     C
        ADD     A,#10H
        MOV     DPH,A
        SJMP    FIL00

CHANGE0:CJNE    A,#0F7H,CHANGE1
        MOV     A,DPH
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPH,A
        JMP     FIL00

CHANGE1:CJNE    A,#0FBH,CHANGE2
        MOV     A,DPL
        CLR     C
        ADD     A,#10H
        MOV     DPL,A
        JMP     FIL00

CHANGE2:CJNE    A,#0FDH,CHANGE3
        MOV     A, DPL
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPL,A
        JMP     FIL00

CHANGE3:CJNE    A,#0FEH,ENTER
        JB      78H,FIL22
        JB      79H,FIL11
        MOV     R7,DPH          ;Inicio del bloque dph
        MOV     R5,DPL          ;inicio del bloque dpl
        SETB    79H
        JMP     FIL00
```

```
FIL11:  INC     DPTR
        MOV     R4,DPH
        MOV     R3,DPL
        SETB    7DH
        MOV     DPL,#0FFH
        JMP     FIL00
FIL22:  MOV     R6,DPL          ;En R6 esta el valor a llenar
        CLR     C
        MOV     A,R3
        SUBB    A,R5
        MOV     R3,A            ;En R3 esta la cifra menos signi
        MOV     A,R4
        SUBB    A,R7
        MOV     R4,A            ;En R4 esta la cifra mas signifi
        MOV     DPH,R7
        MOV     DPL,R5
        SETB    76H             ;Bit de llamada a insertar bloq
        CJNE    R3,#00H,BLOQ9
        CJNE    R4,#00H,BLOQ9
        SJMP    ENTER
BLOQ9:  INC     R3
        INC     R4
        SJMP    LOOP
BLOQ8:  LCALL   FILBLOQ
LOOP:   DJNZ    R3,BLOQ8
        DJNZ    R4,BLOQ8
ENTER:  RET

;Subrutina que permite poner dos puntos de parada consecutivos
;en el programa, con remoción automática.

FUNC4:  PUSH    ACC
        PUSH    B
        PUSH    1EH
        CLR     76H
        CLR     7CH
        CLR     77H
        MOV     B,#04H
PARAD0: MOV     DPTR,#PARADA
        LCALL   MESSAGE
        DJNZ    B,PARAD0
        MOV     DPTR,#EPROM

PARAD1: MOV     R0,#SCREEN5
        MOV     A,DPL
        MOV     @R0,A
        MOV     A,DPH
        DEC     R0
        DEC     R0
        MOV     @R0,A
        LCALL   CODDIS
        JB      7CH,PUNT2
        MOV     SCREEN0,#06H
        SJMP    PUNT3
PUNT2:  MOV     SCREEN0,#5BH
```

```
PUNT3:  MOV     SCREEN1,#40H
        LCALL   DISPLAYS
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        CJNE    A,#0FFH,MARK
        JMP     PARAD1

MARK:   CJNE    A,#0EFH,MARK0
        MOV     A,DPH
        CLR     C
        ADD     A,#10H
        MOV     DPH,A
        SJMP    PARAD1

MARK0:  CJNE    A,#0F7H,MARK1
        MOV     A,DPH
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPH,A
        JMP     PARAD1

MARK1:  CJNE    A,#0FBH,MARK2
        MOV     A,DPL
        CLR     C
        ADD     A,#10H
        MOV     DPL,A
        JMP     PARAD1

MARK2:  CJNE    A,#0FDH,MARK3
        MOV     A, DPL
        CLR     C
        MOV     R6,A
        ANL     A,#0F0H
        XCH     A,R6
        ADD     A,#01H
        ANL     A,#0FH
        ORL     A,R6
        MOV     DPL,A
        JMP     PARAD1

MARK3:  CJNE    A,#7FH,MARK4
        MOV     PSW,#BANK1
        JB      7CH,PUNT4
        MOV     R7,#01H
        MOV     R6,DPH
        MOV     R5,DPL
```

```
        SJMP    PUNT5
PUNT4:  CJNE    R7,#02H,PUNT6
        SJMP    MARK4
PUNT6:  MOV     R4,DPH
        MOV     R3,DPL
        MOV     R7,#02H
PUNT5:  MOV     PSW,#BANK3
        MOV     B,#02H
        SETB    76H
NEXCOD: LCALL   FILBLOQ
        DJNZ    B,NEXCOD
        MOV     A,#80H
        MOVX    @DPTR,A
        MOV     A,#0FEH
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        SETB    7CH
PUNT7:  JMP     PARAD1

MARK4:  CJNE    A,#0BFH,MARK5
        MOV     PSW,#BANK1
        CJNE    R7,#01H,NOREMOV
        MOV     PSW,#BANK3
        SJMP    SIREMOV
NOREMOV:CJNE    R7,#02H,PUNT7

SIREMOV:MOV     PSW,#BANK3
        MOV     B,#03H
REMOV0: MOV     DPTR,#REMOV
        LCALL   MESSAGE
        DJNZ    B,REMOV0
        MOV     PSW,#BANK1
        CJNE    R7,#02H,REMOV1
        MOV     DPL,R3
        MOV     DPH,R4
        SETB    77H
        LCALL   DELPUNTS
        LCALL   DELPUNTS
        MOV     R7,#01H
REMOV1: CJNE    R7,#01H,MARK5
        MOV     DPH,R6
        MOV     DPL,R5
        SETB    77H
        LCALL   DELPUNTS
        LCALL   DELPUNTS
        MOV     R7,#00H
        SJMP    MARK6

MARK5:  CJNE    A,#0FEH,PUNT7
        SJMP    MARK6

MARK6:  MOV     PSW,#BANK3
        POP     1EH
        POP     B
```

```
        POP     ACC
        RET

FUNC5:  NOP             ;Tecla de función disponible
        RET

FUNC6:  NOP             ;Tecla de función disponible
        RET

FUNC7:  NOP             ;Tecla de función disponible
        RET

;Subrutina que permite visualizar o escribir en los Registros de Funciones
;Especiales (Accesibles únicamente con direccionamiento directo.)

ESPFUNC:CLR     72H
        CJNE    R1,#80H,JUMP00
        JBC     1,LD0
        MOV     A,80H
        SJMP    PON0
LD0:    MOV     P0,A
PON0:   SETB    72H
        JMP     JUMP14

JUMP00: CJNE    R1,#81H,JUMP01
        JBC     1,LD1
        MOV     A,81H
        SJMP    PON1
LD1:    MOV     SP,A
PON1:   SETB    72H
        JMP     JUMP14

JUMP01: CJNE    R1,#82H,JUMP02
        JBC     1,LD2
        MOV     A,82H
        SJMP    PON2
LD2:    MOV     DPL,A
PON2:   SETB    72H
        JMP     JUMP14

JUMP02: CJNE    R1,#83H,JUMP03
        JBC     1,LD3
        MOV     A,83H
        SJMP    PON3
LD3:    MOV     DPH,A
PON3:   SETB    72H
        JMP     JUMP14

JUMP03: CJNE    R1,#87H,JUMP04
        JBC     1,LD4
        MOV     A,87H
        SJMP    PON4
LD4:    MOV     PCON,A
PON4:   SETB    72H
        JMP     JUMP14
```

```
JUMP04: CJNE    R1,#88H,JUMP05
        JBC     1,LD5
        MOV     A,88H
        SJMP    PON5
LD5:    MOV     TCON,A
PON5:   SETB    72H
        JMP     JUMP14

JUMP05: CJNE    R1,#89H,JUMP06
        JBC     1,LD6
        MOV     A,89H
        SJMP    PON6
LD6:    MOV     TMOD,A
PON6:   SETB    72H
        JMP     JUMP14

JUMP06: CJNE    R1,#8AH,JUMP07
        JBC     1,LD7
        MOV     A,8AH
        SJMP    PON7
LD7:    MOV     TL0,A
PON7:   SETB    72H
        JMP     JUMP14

JUMP07: CJNE    R1,#8BH,JUMP08
        JBC     1,LD8
        MOV     A,8BH
        SJMP    PON8
LD8:    MOV     TL1,A
PON8:   SETB    72H
        JMP     JUMP14

JUMP08: CJNE    R1,#8CH,JUMP09
        JBC     1,LD9
        MOV     A,8CH
        SJMP    PON9
LD9:    MOV     TH0,A
PON9:   SETB    72H
        JMP     JUMP14

JUMP09: CJNE    R1,#8DH,JUMP0A
        JBC     1,LDA
        MOV     A,8DH
        SJMP    PONA
LDA:    MOV     TH1,A
PONA:   SETB    72H
        JMP     JUMP14

JUMP0A: CJNE    R1,#90H,JUMP0B
        JBC     1,LDB
        MOV     A,90H
        SJMP    PONB
LDB:    MOV     P1,A
PONB:   SETB    72H
```

```
        JMP     JUMP14

JUMP0B: CJNE    R1,#98H,JUMP0C
        JBC     1,LDC
        MOV     A,98H
        SJMP    PONC
LDC:    MOV     SCON,A
PONC:   SETB    72H
        JMP     JUMP14

JUMP0C: CJNE    R1,#99H,JUMP0D
        JBC     1,LDD
        MOV     A,99H
        SJMP    POND
LDD:    MOV     SBUF,A
POND:   SETB    72H
        JMP     JUMP14

JUMP0D: CJNE    R1,#0A0H,JUMP0E
        JBC     1,LDE
        MOV     A,0A0H
        SJMP    PONE
LDE:    MOV     P2,A
PONE:   SETB    72H
        JMP     JUMP14

JUMP0E: CJNE    R1,#0A8H,JUMP0F
        JBC     1,LDF
        MOV     A,0A8H
        SJMP    PONF
LDF:    MOV     IE,A
PONF:   SETB    72H
        JMP     JUMP14

JUMP0F: CJNE    R1,#0B0H,JUMP10
        JBC     1,LD10
        MOV     A,0B0H
        SJMP    PON10
LD10:   MOV     P3,A
PON10:  SETB    72H
        JMP     JUMP14

JUMP10: CJNE    R1,#0BBH,JUMP11
        JBC     1,LD11
        MOV     A,0B8H
        SJMP    PON11
LD11:   MOV     IP,A
PON11:  SETB    72H
        JMP     JUMP14

JUMP11: CJNE    R1,#0D0H,JUMP12
        JBC     1,LD12
        MOV     A,0D0H
        SJMP    PON12
LD12:   MOV     PSW,A
```

```
PON12:  SETB    72H
        JMP     JUMP14

JUMP12: CJNE    R1,#0E0H,JUMP13
        JBC     1,LD13
        MOV     A,0E0H
        SJMP    PON13
LD13:   MOV     0E0H,A
PON13:  SETB    72H
        JMP     JUMP14

JUMP13: CJNE    R1,#0F0H,JUMP14
        JBC     1,LD14
        MOV     A,0F0H
        SJMP    PON14
LD14:   MOV     B,A
PON14:  SETB    72H

JUMP14: RET

;       Subrutina que codifica un dato para ser mostrado en los displays

CODDIS: PUSH    ACC
        PUSH    B
        PUSH    DPH
        PUSH    DPL
        PUSH    18H
        PUSH    19H
        PUSH    1CH
        MOV     B,#04H
        MOV     R0,#SCREEN5
SIGDIS: MOV     A,@R0
        ANL     A,#0F0H
        DEC     R0
        SWAP    A
        MOV     @R0,A
        INC     R0
        MOV     A,@R0
        ANL     A,#0FH
        MOV     @R0,A
        DEC     R0
        DEC     R0
        DJNZ    B,SIGDIS
        MOV     R4,#06H
        MOV     R0,#SCREEN5
NEXDIS: MOV     A,@R0
        MOV     R1,A
        INC     R1
        MOV     DPTR,#NUMBERS
NEXNUM: CLR     A
        MOVC    A,@A+DPTR
        INC     DPTR
        DJNZ    R1,NEXNUM
        MOV     @R0,A
        DEC     R0
```

```
        DJNZ    R4,NEXDIS
        POP     1CH
        POP     19H
        POP     18H
        POP     DPL
        POP     DPH
        POP     B
        POP     ACC
        RET

; Subrutina que decrementa el puntero DPTR un número determinado de veces
; indicado por el registro R4.

DECREM: PUSH    ACC
        MOV     A,R4
        PUSH    ACC
DEC:    CLR     C
        MOV     A,DPL
        SUBB    A,#01H
        MOV     DPL,A
        JC      DEC1
        SJMP    DEC2
DEC1:   CLR     C
        MOV     A,DPH
        SUBB    A,#01H
        MOV     DPH,A
DEC2:   DJNZ    R4,DEC
        POP     ACC
        MOV     R4,A
        POP     ACC
        RET

;Subrutina que muestra en los displays los datos contenidos en las localidades
;de memoria 07AH hasta la 07FH, teniendo como dato menos significativo el de la
;dirección 07AH

DISPLAYS:
        PUSH    ACC             ;Se guarda los registros ocupados
        PUSH    B               ;para no alterar los resulta
        PUSH    DPH             ;dos al llamar a la subrutina
        PUSH    DPL
        PUSH    1BH
        PUSH    1BH
        PUSH    1CH
        CLR     A
        MOV     R3,#05H         ;Contador del # de barridas externas
NEW2:   MOV     R4,#08H         ;contador menos significativo
NEW1:   MOV     R0,#SCREEN5     ;Cargar R0 con la dirección de Screen-5
        MOV     B,#06H          ;contador del número de displays
        MOV     DPTR,#HABDISPLAYS;Carga el # de habilitación de Screen-5
NEW:    PUSH    DPH
        PUSH    DPL
        MOVC    A,@A+DPTR       ;Cargar al ACC la hab. de SC-5
        MOV     DPTR,#SALHAB
        MOVX    @DPTR,A         ;Sacar al latch de Salida de hab.
```

```
        MOV     DPTR,#SALDAT    ;Cargar dptr con # del latch de datos
        MOV     A,@R0
        MOVX    @DPTR,A         ;sacar el dato al latch de datos
        LCALL   DELAY           ;Mantener el dato un momento
        CLR     A               ;Limpiar el dato para evitar
        MOVX    @DPTR,A         ;superposición de datos.
        POP     DPL
        POP     DPH
        CLR     A
        INC     DPTR            ;incrementar dptr,y r0 para seguir
        DEC     R0              ;haciendo un barrido de las loca
        DJNZ    B,NEW           ;lidades de memoria de displays
        DJNZ    R4,NEW1
        DJNZ    R3,NEW2
        POP     1CH
        POP     1BH
        POP     18H
        POP     DPL
        POP     DPH
        POP     B
        POP     ACC
        RET

;Subrutina que introduce un pequeño retardo utilizando el timer1

DELAY:  PUSH    ACC
        PUSH    B
        PUSH    DPH
        PUSH    DPL
        MOV     B,#02H
REPIT:  MOV     TCON,#00H
        MOV     TMOD,#00100000B
        MOV     TH1,#0FDH
        SETB    TCON.6
WAIT:   JNB     TCON.7,WAIT
        CLR     TCON.7
        CLR     TCON.6
        MOV     TL1,#00H
        DJNZ    B,REPIT
        POP     DPL
        POP     DPH
        POP     B
        POP     ACC
        RET

;  Subrutina que escribe el mensaje de "OK" en los displays.

WRIT:   PUSH    B
        MOV     B,#02H
WRIT01: MOV     DPTR,#OK
        CALL    MESSAGE
        DJNZ    B,WRIT01
        POP     B
        RET
```

```
;  Subrutina que escribe el mensaje de "Error" en los displays

ERR:    PUSH    B
        MOV     B,#02H
ERR1:   MOV     DPTR,#ERROR
        CALL    MESSAGE
        DJNZ    B,ERR1
        POP     B
        RET

;    Subrutina que muestra un mensaje directamente en los displays,
;    de acuerdo a la dirección que tenga el DPTR.

MESSAGE:PUSH    B
        PUSH    18H
        PUSH    ACC
        MOV     A,#00H
        MOV     B,#06H
        MOV     R0,#SCREEN5
CARGA:  MOVC    A,@A+DPTR
        MOV     @R0,A
        INC     DPTR
        DEC     R0
        CLR     A
        DJNZ    B,CARGA
        LCALL   DISPLAYS
        POP     ACC
        POP     18H
         POP    B
        RET

;       Subrutina que inicializa el pórtico de recepción serial

INITRX: MOV     DPTR,#MODO
        MOV     A,#00H
        MOVC    A,@A+DPTR
        MOV     TMOD,A
        CLR     A
        INC     DPTR
        MOVC    A,@A+DPTR
        MOV     TCON,A
        MOV     TH1,#BAUD
        MOV     TL1,#BAUD
        SETB    TR1
        INC     DPTR
        CLR     A
        MOVC    A,@A+DPTR
        MOV     SCON,A
        RET

MODO:   DB      M1M021
        DB      TTCON0
        DB      MOD1+SSMM22+ENRX

REMOV:  DB      80H,1CH,5CH,54H,79H,50H
```

```
PARADA: DB      0DCH,5EH,0DCH,50H,0DCH,73H
RECEP:  DB      80H,73H,79H,39H,79H,50H
OK:     DB      40H,40H,0F2H,5CH,40H,40H
TEST:   DB      00H,80H,78H,6DH,79H,78H
ERROR:  DB      80H,50H,5CH,50H,50H,79H
SIDEL:  DB      80H,06H,79H,5EH,04H,6DH
RAMTEST:DB      40H,80H,54H,77H,50H,40H
SIMEPRO:DB      54H,54H,5CH,50H,73H,79H
SFR:    DB      40H,80H,50H,71H,6DH,40H
REGISTROS:DB    78H,6DH,04H,6FH,79H,50H
VERCEL: DB      40H,80H,50H,79H,1CH,40H
BLOQUE: DB      79H,3EH,0DCH,5CH,6H,7CH
FILL:   DB      40H,06H,06H,04H,71H,40H
INSERTE:DB      78H,50H,79H,6DH,54H,04H
DEL:    DB      50H,77H,50H,50H,5CH,7CH

          ;  Códigos de habilitación de los displays

HABDISPLAYS:    DB      01H,02H,04H,08H,10H,20H  ;men-sig,...MAS-sig

      ; Codificación de los números hexadecimales para los displays

NUMBERS:D8      3FH     ;Cero
        DB      06H     ;Uno
        DB      5BH     ;Dos
        DB      4FH     ;Tres
        DB      66H     ;Cuatro
        DB      6DH     ;Cinco
        DB      7DH     ;Seis
        DB      07H     ;Siete
        DB      7FH     ;Ocho
        DB      6FH     ;Nueve
        DB      77H     ;A h
        D8      7CH     ;B h
        D8      39H     ;C h
        DB      5EH     ;D h
        DB      79H     ;E h
        DB      71H     ;F h

        END
```

3.4.3 Subrutinas del Usuario.

El módulo de desarrollo ofrece la facilidad al usuario de poder utilizar varias subrutinas de propósito general que se encuentran contempladas dentro de él.

- Subrutina Displays.

La subrutina displays realiza un barrido de las localidades 7AH hasta la 7FH de la RAM interna del microcontrolador, mostrando en el display los datos contenidos en estas localidades de memoria; la localidad 7AH corresponde al dígitos menos significativo y consecutivamente la localidad 7FH al dígito mas significativo del display. Esta subrutina se presenta al usuario en tres modalidades:

- Displays. Se encuentra a partir de la localidad 1000H y esta tiene la particularidad de tener un tiempo de duración de 0.8 segundos aproximadamente.

Es decir que cualquier actualización de datos que se quiera hacer en los displays tendrá un retardo de 0.8 segundos, que es el tiempo que se demora esta subrutina en completar su barrido.

- Displays1. Se encuentra organizada a partir de la localidad 1100H, con un tiempo de barrido de 0.5 segundos aproximadamente.
- Displays2. Se encuentra organizada a partir de la localidad 1200H, con un tiempo de barrido de 0.1 segundos aproximadamente.

De acuerdo a la necesidad del usuario este puede usar una de las tres variedades de la presente subrutina.

- Subrutina Coddis.

Esta subrutina codifica los datos contenidos en las localidades 7FH, 7DH, y 7BH tarnsformándolos a código de siete segmentos.
Los dos códigos de siete segmentos resultantes del dato contenido inicialmente en esta localidad se almacenan nuevamente en las localidades 7FH y 7EH; la localidad 7DH, se almacena en la localidad 7DH y 7CH; la localidad 7BH en las localidades 7BH y 7AH.
Esta subrutina se encuentra organizada partir de la localidad 1300H.

- Subrutina Decrem.

Se encuentra organizada a partir de la localidad 1400H.
Esta decrementa el puntero DPTR, el número de veces que indique el registro R4.

- Subrutina Delay.

Se encuentra organizada a partir de la localidad 1500H, e introduce un retardo de aproximadamente 15 mseg. utilizando el Timer1 del microcontrolador.

- Subrutina Message.

Se organiza en la localidad 1600H.

Esta carga a las localidades de memoria ocupadas por el display seis códigos a partir de la dirección que indique el registro DPTR, para luego mostrarlos en el display.

- Subrutina Writ.

Muestra en el display el mensaje de "OK."
Se encuentra en la localidad 1700H

- Subrutina Err.

Muestra el mensaje de "Error" en el display. Se encuentra en la localidad 1750H.
Todas las subrutinas almacenan mediante la instrucción PUSH el valor de los registros que estas ocupan, para recuperarlos a la salida de la misma. De esta manera no se altera cualquier valor que el usuario tenga en estos registros. Adicionalmente el usuario puede trabajar en cualquier banco de registros de la RAM interna y llamar a estas subrutinas, sin que se alteren los valores que se tenga al inicio de la llamada.

CAPITULO IV

CONSTRUCCION Y PRUEBAS

- Construcción y pruebas del módulo de desarrollo
- Modularidad y facilidades de ampliación del módulo de desarrollo.
- Notas de aplicación referentes a la familia MCS-51.
- Conclusiones y recomendaciones.

4.1 CONSTRUCCION Y PRUEBAS DEL MODULO DE DESARROLLO

El hardware del módulo de desarrollo es muy sencillo realizarlo, basta con seguir cuidadosamente el diagrama de conexiones graficado en el capítulo anterior.

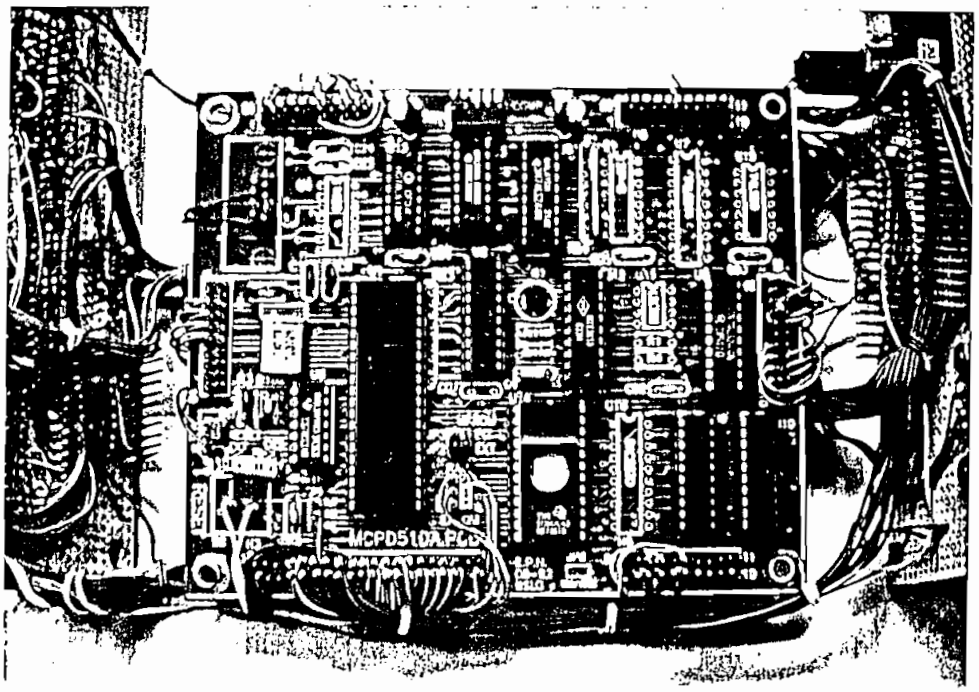
Como proyecto en desarrollo se recomienda realizar la interconexión de los elementos mediante zócalos para entorchar alambres (Wirewrap), debido a la facilidad que presentan estos elementos para realizar cualquier conexión o desconexión de algún terminal del circuito.

Además la herramienta de entorchar es muy fácil de usar, con lo que en primera instancia se evita daños en los elementos debido al calentamiento de elementos por suelda, por otra parte corregir una mala conexión es más fácil en wirewrap antes que realizarlo en elementos soldados.

La parte central del circuito del módulo está realizado sobre la tarjeta base MCPD51DA realizada y probada en el Laboratorio de Electrónica de Potencia.

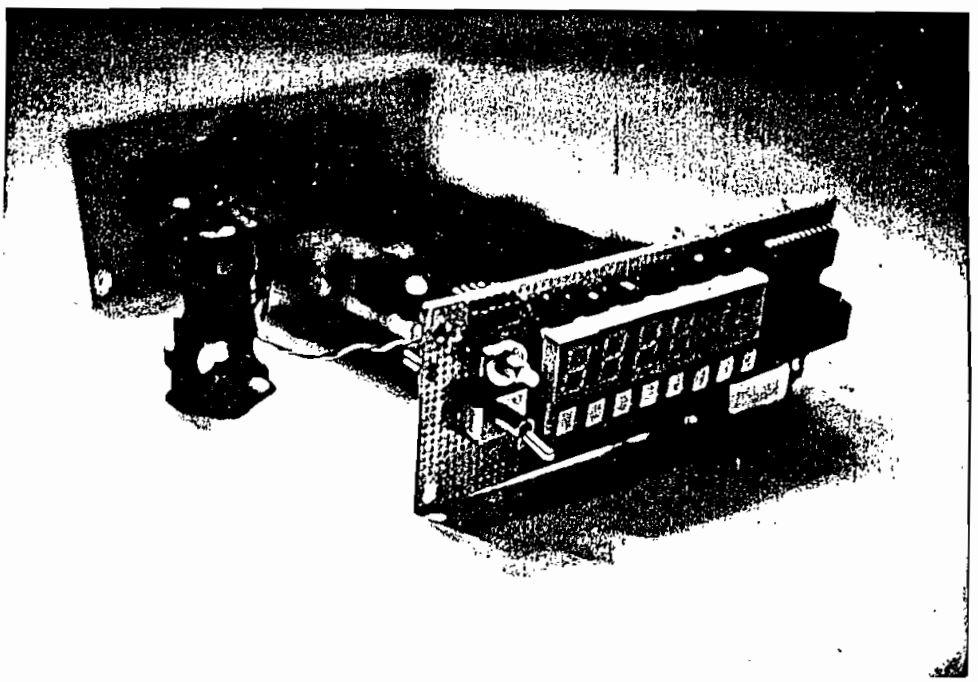
En esta tarjeta se encuentra el circuito básico de funcionamiento del microcontrolador. Esta parte del circuito se encuentra montada sobre zócalos soldados sobre la tarjeta, el detalle de esta tarjeta se la puede encontrar en el Anexo 1.

Fotografía 1: Tarjeta MCPD51DA.



En lo que se refiere a todos los periféricos que se encuentran en el módulo, estos están realizados en la técnica de wirewrap montados sobre baquelita perforada.

Fotografía 2: Elementos montados sobre baquelita perforada, vista frontal.



En la construcción del módulo y de cualquier hardware en

general es recomendable ir avanzando de manera modular; es decir, realizar una parte del mismo, probar el adecuado funcionamiento de esta parte y luego seguir al módulo siguiente, de esta manera al realizar las pruebas correspondientes del segundo módulo se puede estar seguro que si se tiene alguna falla en los resultados esperados esta se centra en el segundo bloque.

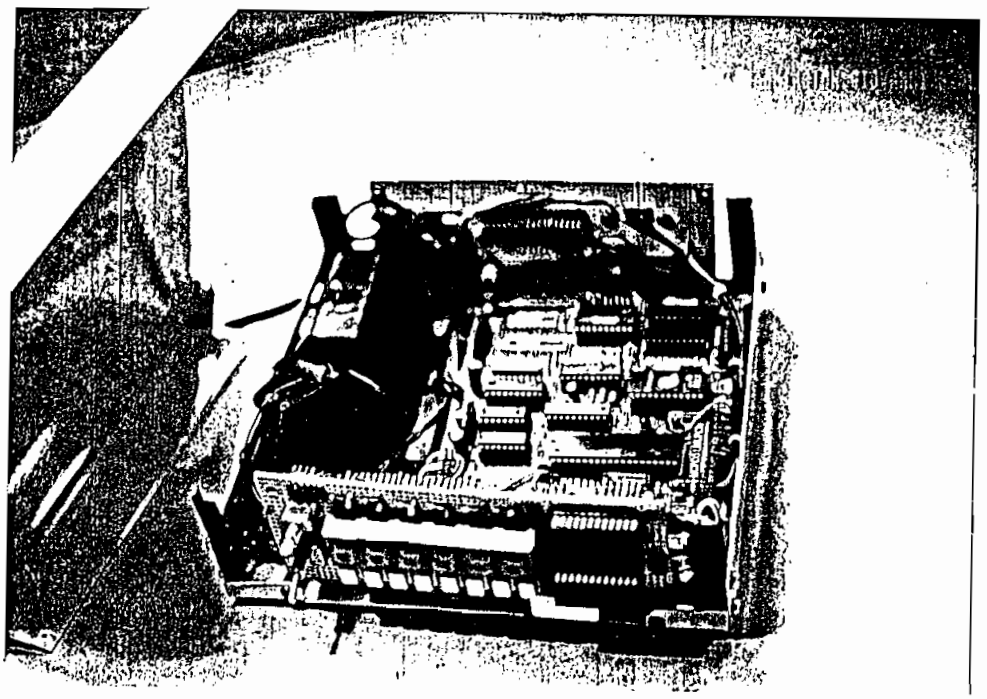
La primera parte realizada en el módulo de desarrollo es lo que se refiere a todo el circuito básico, luego se realiza la parte correspondiente a la salida de datos hacia displays y la entrada de datos desde teclado, de esta forma se puede visualizar cual es el error y corregirlo de manera rápida y eficaz.

Para realizar la primera prueba del módulo se realizó un pequeño programa cuyo objetivo básico es el comprobar el buen funcionamiento de las pantallas.

El programa saca a todas las pantallas los dígitos hexadecimales del "0" a la "F" consecutivamente en un lazo infinito.

El programa en mención se lo lista a continuación, bajo el nombre de prueba # 1.

Fotografía 3: Montaje de tarjetas y fuente de alimentación. Vista superior.



```
;          MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;                MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:      ESCUELA POLITECNICA NACIONAL
;                   FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:    ELECTRONICA Y TELECOMUNICACIONES
;                   WASHINGTON GIOVANNI SIDEL TORRES
;                            1992-1993
;
;                           PRUEBA # 1
;
$ALLPUBLIC
;   Area de inicialización de puntos de entrada de interrupciones.
;   Sacar de 0 a F consecutivamente en Disp
;ETIQUETA:      OPC     OPERANDOS       ;COMENTARIOS

INICIO:           ;Definición de etiquetas
        ;Definición de las direcciones de entrada/salida

STACK:  EQU     4FH     ;Dirección del Stack en 50h
SCREEN5:EQU     7FH     ;Pantalla mas significativa #5
SCREEN4:EQU     7EH     ;Pantalla # 4
SCREEN3:EQU     7DH     ;Pantalla # 3
SCREEN2:EQU     7CH     ;Pantalla # 2
SCREEN1:EQU     7BH     ;Pantalla # 1
SCREEN0:EQU     7AH     ;Pantalla # 0
SALHAB: EQU     0000H   ;Habilitación displays de salida
SALDAT: EQU     2000H   ;Salida de datos hacia Displays
TECLAS: EQU     0000H   ;Entrada de datos de las teclas
DISPLAYS:EQU    1000H   ;Dirección de la subrutina.
```

```
          ; Programa Principal

        MOV     SP,#STACK       ;Inicializar el Stack en 50h
START:  MOV     DPTR,#NUMBERS
        MOV     R1,#10H         ;Contador del # de digitos hex.
NEWNUM: MOV     B,#06H          ;Contador del # de displays
        MOV     R0,#SCREEN5
        MOVC    A,@A+DPTR       ;Se cargan los números hexa
NEWDIS: MOV     @R0,A           ;decimales desde el uno hasta la
        DEC     R0              ;efe en los displays
        DJNZ    B,NEWDIS
        LCALL   DISPLAYS
        INC     DPTR
        CLR     A
        DJNZ    R1,NEWNUM
        MOV     R1,#08H         ;Contador de número de teclas
LOAD1:  MOV     A,#40H          ;Se carga ------ en los displays
        MOV     R0,#SCREEN5     ;(40h), como señal de acabada la
        MOV     B,#06H          ;prueba de displays.
PASS:   MOV     @R0,A
        DEC     R0
        DJNZ    B,PASS
        LCALL   DISPLAYS        ;Fin de la prueba de displays
        MOV     DPTR,#SALHAB
        MOV     A,#00H
        MOVX    @DPTR,A
        SJMP    START


NUMBERS:DB      3FH     ;Cero
        DB      06H     ;Uno
        DB      5BH     ;Dos
        DB      4FH     ;Tres
        DB      66H     ;Cuatro
        DB      6DH     ;Cinco
        DB      7DH     ;Seis
        DB      07H     ;Siete
        DB      7FH     ;Ocho
        DB      6FH     ;Nueve
        DB      77H     ;A h
        DB      7CH     ;B h
        DB      39H     ;C h
        DB      5EH     ;D h
        DB      79H     ;E h
        DB      71H     ;F h

        END
```

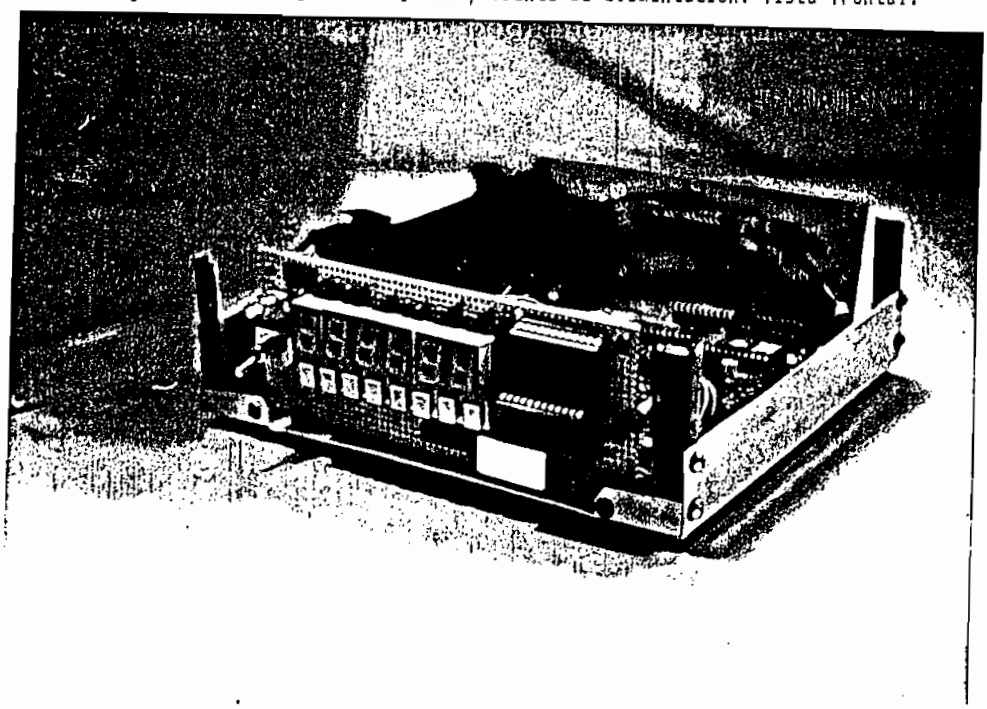
Para la prueba del teclado, de igual manera se vale de un programa el cual, básicamente realiza un barrido del teclado en espera de un dato, este se compara

sucesivamente con la tabla de códigos válidos del teclado, si el código es el correspondiente a una de las ocho teclas sale en pantalla el mensaje de "OK" , caso contrario se observa el mensaje de "Error" . Como en el programa no se discrimina el orden en que se presionen las teclas; lo mas conveniente es presionar cada una de las teclas una sola vez para probar todas y cada una de ellas. A continuación se puede ver dicho programa bajo el nombre de prueba # 2.

```
;            MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;              MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:      ESCUELA POLITECNICA NACIONAL
;                   FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:    ELECTRONICA Y TELECOMUNICACIONES
;                   WASHINGTON GIOVANNI SIDEL TORRES
;                            1992-1993
;                      PRUEBA # 2
;
$ALLPUBLIC
;   Area de inicialización de puntos de entrada de interrupciones.
;
;ETIQUETA:      OPC     OPERANDOS       ;COMENTARIOS

INICIO:           ;Definición de etiquetas
         ;Definición de las direcciones de entrada/salida

STACK:  EQU     4FH     ;Dirección del Stack en 50h
SCREEN5:EQU     7FH     ;Pantalla mas significativa #5
SCREEN4:EQU     7EH     ;Pantalla # 4
SCREEN3:EQU     7DH     ;Pantalla # 3
SCREEN2:EQU     7CH     ;Pantalla # 2
SCREEN1:EQU     7BH     ;Pantalla # 1
SCREEN0:EQU     7AH     ;Pantalla # 0
SALHAB: EQU     0000H   ;Habilitación displays de salida
SALDAT: EQU     2000H   ;Salida de datos hacia Displays
TECLAS: EQU     0000H   ;Entrada de datos de las teclas
DISPLAYS:EQU    1000H   ;Direcciones de las subrutinas
DELAY:  EQU     1500H   ;de uso por el usuario.
WRIT:   EQU     1700H
ERR:    EQU     1750H
MESSAGE:EQU     1600H
         ; Programa Principal

      MOV     SP,#STACK       ;Inicializar el Stack en 50h
START:  MOV     B,#06H          ;Encerar las localidades de
```

```
        MOV     A,#00H          ;memoria desde la 7fh(mas sig)
        MOV     R0,#SCREEN5     ;hasta la 7ah (menos sig)
INZERO: MOV     @R0,A
        DEC     R0
        DJNZ    B,INZERO
LOAD1:  MOV     DPTR,#MENOS
        LCALL   MESSAGE

         ;A continuación se realiza una prueba de las teclas

        CLR     A
        MOV     DPTR,#TECLAS
        MOV     R1,#08H
LOAD2:  LCALL   DELAY
        DJNZ    R1,LOAD2
                                ;Se prueba cada una de las teclas
        MOVX    A,@DPTR         ;si el código de entrada de la
        CJNE    A,#0FFH,WRITOK  ;tecla es el correcto se presen
        SJMP    LOAD1           ;ta en los displays el mensaje
WRITOK: CJNE    A,#0FEH,WRIT0   ;de --OK-- caso contrario se
        LCALL   WRIT            ;presenta el mensaje de ERROR
        SJMP    FINTEC          ;esto se repite ocho veces para
WRIT0:  CJNE    A,#0FDH,WRIT1   ;para las ocho teclas
        LCALL   WRIT
        SJMP    FINTEC
WRIT1:  CJNE    A,#0FBH,WRIT2
        LCALL   WRIT
        SJMP    FINTEC
WRIT2:  CJNE    A,#0F7H,WRIT3
        LCALL   WRIT
        SJMP    FINTEC
WRIT3:  CJNE    A,#0EFH,WRIT4
        LCALL   WRIT
        SJMP    FINTEC
WRIT4:  CJNE    A,#0DFH,WRIT5
        LCALL   WRIT
        SJMP    FINTEC
WRIT5:  CJNE    A,#0BFH,WRIT6
        LCALL   WRIT
        SJMP    FINTEC
WRIT6:  CJNE    A,#07FH,WRIT7
        LCALL   WRIT
        SJMP    FINTEC
WRIT7:  LCALL   ERR
FINTEC: JMP     START           ;Lazo de repetición.

MENOS:  DB      40H,40H,40H,40H,40H,40H

        END
```

Fotografía 4: Montaje de tarjetas y fuente de alimentación. Vista frontal.



Con la prueba de funcionamiento de la salida a displays, y con la prueba de entrada de datos desde teclado, ya se pueden intentar hacer varios programas de prueba y de funcionamiento general del sistema, ya que estas ayudas son una de las mas importantes en el desarrollo general de un programa. Debido a que se tiene la posibilidad de visualizar cualquier resultado que se desee obtener y por otra parte se pueden ingresar datos arbitrarios a conveniencia del usuario.

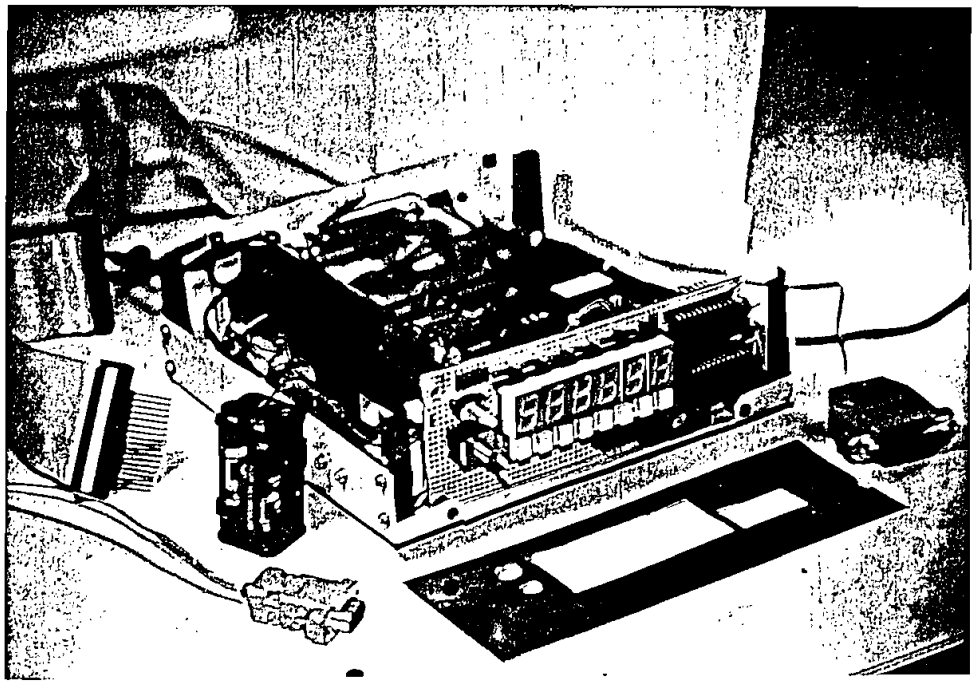
Otro módulo que necesariamente requiere un programa de prueba es el referente a la memoria de datos; ya que por una parte se debe comprobar el buen funcionamiento de la memoria y por otra se debe estar seguro que el hardware implementado sea el correcto.

Para el caso de la memoria en el sistema del módulo de desarrollo, se realiza un pequeño programa en el cual se

escribe todas las localidades de memoria (2K) con el número "00H", y a continuación se realiza un proceso de lectura de todas las localidades, comparando en primera instancia que el dato leido sea 00H, a continuación se realiza un proceso de escritura pero ahora con el dato "FFH" y nuevamente se realiza la comparación, si todo este proceso se cumple se escribe en las pantallas el mensaje de "OK" caso contrario se obtiene el mensaje de "ERROR".

Mientras se realiza el proceso de lectura de las localidades de memoria se puede apreciar en las pantallas tanto la localidad que esta siendo accesada como el dato que esta presente en dicha localidad. El programa que realiza esta verificación se encuentra a continuación.

Fotografía 5: Módulo de Desarrollo y todos sus elementos.



```
;           MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;                MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:      ESCUELA POLITECNICA NACIONAL
;                   FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:    ELECTRONICA Y TELECOMUNICACIONES
;                   WASHINGTON GIOVANNI SIDEL TORRES
;                            1992-1993
;
;                          PRUEBA # 3
;
$ALLPUBLIC
;   Area de inicialización de puntos de entrada de interrupciones.
;
;ETIQUETA:      OPC     OPERANDOS       ;COMENTARIOS

INICIO:           ;Definición de etiquetas
         ;Definición de las direcciones de entrada/salida


INIRAM: EQU     8000H   ;Inicio de la RAM
FINRAM: EQU     87FFH   ;Fin de la RAM (2Kb)
EPROM:  EQU     0A000H  ;Direc de inicio de la SIM-EPROM
ENDEPROM:EQU    0A7FFH  ;Fin de la sim-eprom (4kb)
STACK:  EQU     4FH     ;Dirección del Stack en 50h
SCREEN5:EQU     7FH     ;Pantalla mas significativa #5
SCREEN4:EQU     7EH     ;Pantalla # 4
SCREEN3:EQU     7DH     ;Pantalla # 3
SCREEN2:EQU     7CH     ;Pantalla # 2
SCREEN1:EQU     7BH     ;Pantalla # 1
SCREEN0:EQU     7AH     ;Pantalla # 0
SALHAB: EQU     0000H   ;Habilitación displays de salida
SALDAT: EQU     2000H   ;Salida de datos hacia Displays
TECLAS: EQU     0000H   ;Entrada de datos de las teclas
MESSAGE:EQU     1600H   ;Dirección de subrutinas de uso
DISPLAYS2:EQU   1200H   ;del usuario.
CODDIS: EQU     1300H
WRIT:   EQU     1700H
ERR:    EQU     1750H

         ; Programa Principal

       MOV      SP,#STACK       ;Inicializar el Stack en 50h
       MOV      B,#06H          ;Encerar las localidades de
       MOV      A,#00H          ;memoria desde la 7fh(mas sig)
       MOV      R0,#SCREEN5     ;hasta la 7ah (menos sig)
INZERO: MOV     @R0,A
       DEC      R0
       DJNZ     B,INZERO
       CLR      0H

            ;     Test de la meoria RAM

START:  MOV     B,#02H
OTHER2: MOV     DPTR,#RAMTEST   ;Escribir la palabra RAM
       LCALL    MESSAGE         ;y se muestra en los displays.
```

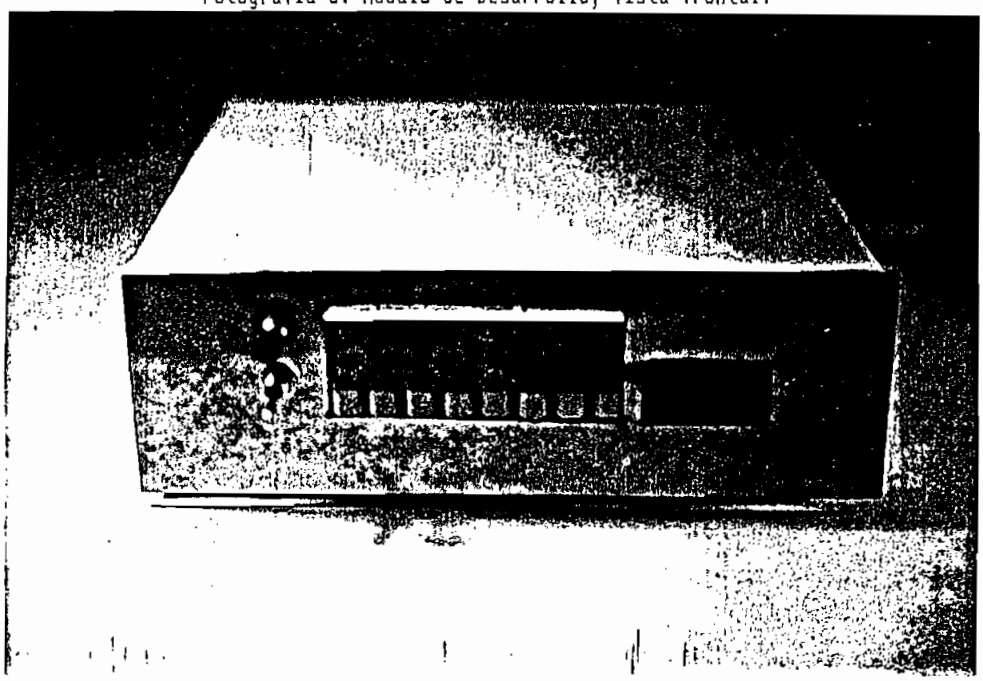
```
        DJNZ    B,OTHER2
        MOV     DPTR,#INIRAM
        MOV     R1,#87H         ;cargar 2K. en los registros
        MOV     R2,#0FFH        ;07FFH.
PROXCEL:MOV     A,#00H          ;Cargar el ACC con 00h y
        MOVX    @DPTR,A         ;escribir en todas las
        INC     DPTR            ;celdas éste número
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXTCEL
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXTCEL
        JMP     COMPBAR         ;se comprueba que en todas
NEXTCEL:SJMP    PROXCEL         ;las celdas de memoria esté
                                ;escrito el número 00h, si en
COMPBAR:MOV     DPTR,#INIRAM    ;todas las celdas se encuentra
NEXTC1: CLR     C               ;el #00h se completa esta parte
        MOVX    A,@DPTR         ;del test caso contrario se
                                ;se muestra el mensaje de
        PUSH    ACC             ;"Error".
        PUSH    DPL
        PUSH    DPH
        MOV     R0,#SCREEN5
        MOV     @R0,A
        MOV     A,DPL
        DEC     R0
        DEC     R0
        MOV     @R0,A
        MOV     A,DPH
        DEC     R0
        DEC     R0
        MOV     @R0,A
        LCALL   CODDIS
        LCALL   DISPLAYS2
        POP     DPH
        POP     DPL
        POP     ACC

        LCALL   DISPLAYS2
        CJNE    A,#00H,BAD      ;muestra el mensaje de ERROR
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXTC1
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXTC1
        JMP     COMP1
BAD:    LCALL   ERR
        JMP     ENDRAM
```

```
COMP1:  MOV     DPTR,#INIRAM    ;se escribe todas las celdas
PROX:   MOV     A,#0FFH         ;con el número ffh.
        MOVX    @DPTR,A
        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXT22
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXT22
        JMP     COMP22
NEXT22: SJMP    PROX
COMP22: MOV     DPTR,#INIRAM    ;compruebar que en todas las cel
NEXTC2: MOVX    A,@DPTR         ;das se encuentre el #ffh.
        CJNE    A,#0FFH,BAD

        PUSH    ACC
        PUSH    DPL
        PUSH    DPH
        MOV     R0,#SCREEN5
        MOV     @R0,A
        MOV     A,DPL
        DEC     R0
        DEC     R0
        MOV     @R0,A
        MOV     A,DPH
        DEC     R0
        DEC     R0
        MOV     @R0,A
        LCALL   CODDIS
        LCALL   DISPLAYS2
        POP     DPH
        POP     DPL
        POP     ACC

        INC     DPTR
        CLR     C
        MOV     A,DPH
        SUBB    A,R1
        CJNE    A,#00H,NEXTC2
        MOV     A,DPL
        CLR     C
        SUBB    A,R2
        CJNE    A,#00H,NEXTC2

        MOV     R1,#04H
REP:    LCALL   WRIT            ;Si todo esta bien se muestra
        DJNZ    R1,REP          ;el mensaje de OK.
ENDRAM: SETB    0H
        JMP     START
```

```
RAMTEST:DB      40H,80H,54H,77H,50H,40H
SIMEPRO:DB      54H,54H,5CH,50H,73H,79H

        END
```

Nótese que con estas pruebas básicas, y que no son complicadas en su concepción se llega a la certeza que los pórticos de entrada de datos desde teclado, salida hacia pantallas y el banco de almacenamiento de datos se encuentran en perfectas condiciones de funcionamiento y por otro lado que el circuito implementado es el correcto.
Una vez que se tiene esta seguridad se comienza a trabajar y a desarrollar el programa monitor de control del módulo de desarrollo.
De tal suerte que cuando se va probando parte por parte el programa desarrollado, si algún resultado no es el correcto se descarta de inmediato fallas debido al hardware del sistema, y únicamente se centra las correcciones necesarias dentro del programa.

Fotografía 6: Módulo de Desarrollo, vista frontal.



4.2 MODULARIDAD Y FACILIDADES DE AMPLIACION DEL MODULO DE DESARROLLO.

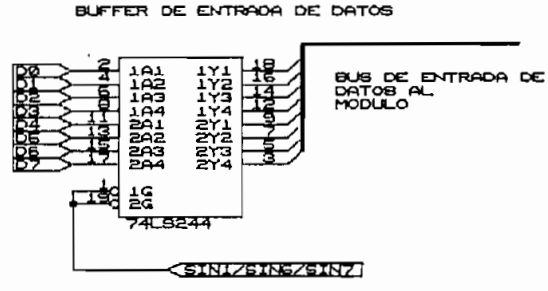
El módulo de desarrollo esta diseñado, y programado de manera modular; es decir que este se puede fácilmente acoplar a otro sistema por diferente que sea, con el fin de ampliar sus capacidades individuales.

Para este efecto el hardware del módulo consta de dos salidas ubicadas en la parte posterior del mismo. Una de ellas es un conector de 40 postes en la cual se han ubicado varias señales mediante las cuales se pueden realizar la interconexión con otro sistema dado.

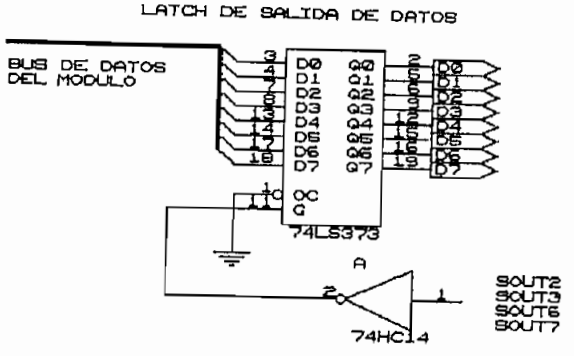
Asi mismo, aquí se encuentran ubicadas las señales de habilitación para entrada de datos SIN1, SIN6, y SIN7, mediante las cuales y por intermedio de un buffer octal se puede realizar el ingreso de datos hacia el microcontrolador central.

La manera de hacerlo es igual que cualquier ingreso de datos realizado en anteriores ocasiones. Conviene revisar las direcciones en las cuales están ubicados los pórticos para el usuario.

DIRECCION DE PORTICOS			
DIRECC. HEX.	PORT.#	LIN. DE SELECCION	
		READ	WRITE
2000-3FFF	1	SIN1	
4000-5FFF	2		SOUT2
6000-7FFF	3		SOUT3
C000-DFFF	6	SIN6	SOUT6
E000-FFFF	7	SIN7	SOUT7



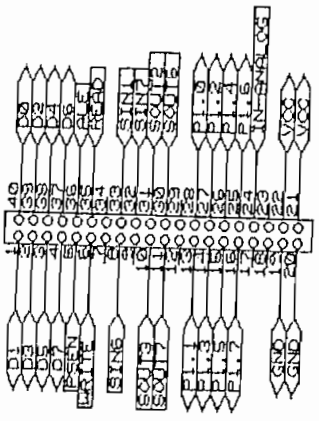
De igual manera en los postes también se tiene las líneas de habilitación para salida de datos SOUT2, SOUT3, SOUT6, SOUT7, las cuales por intermedio de un latch se tiene la posibilidad de sacar un dato dado hacia un módulo cualquiera. La manera de realizar esta salida es igual al hardware hecho anteriormente.



Este es sólo un ejemplo de algunas interconexiones que se pueden realizar desde y hacia el módulo de desarrollo; ya que dentro del conector de 40 postes también se puede encontrar las líneas de RD, WR, ALE, PSEN, con las cuales se pueden hacer infinidad de combinaciones tanto para entrada como para salida de datos desde y hacia el módulo.

Otras líneas que pueden ser muy útiles son las correspondientes al pórtico uno, dentro del sistema del módulo, estas líneas no se las usa de ninguna manera por lo tanto pueden ser usadas por el usuario como líneas de propósito general configuradas de acuerdo a la necesidad que este tenga.

La distribución completa de los postes la presentamos nuevamente a continuación.

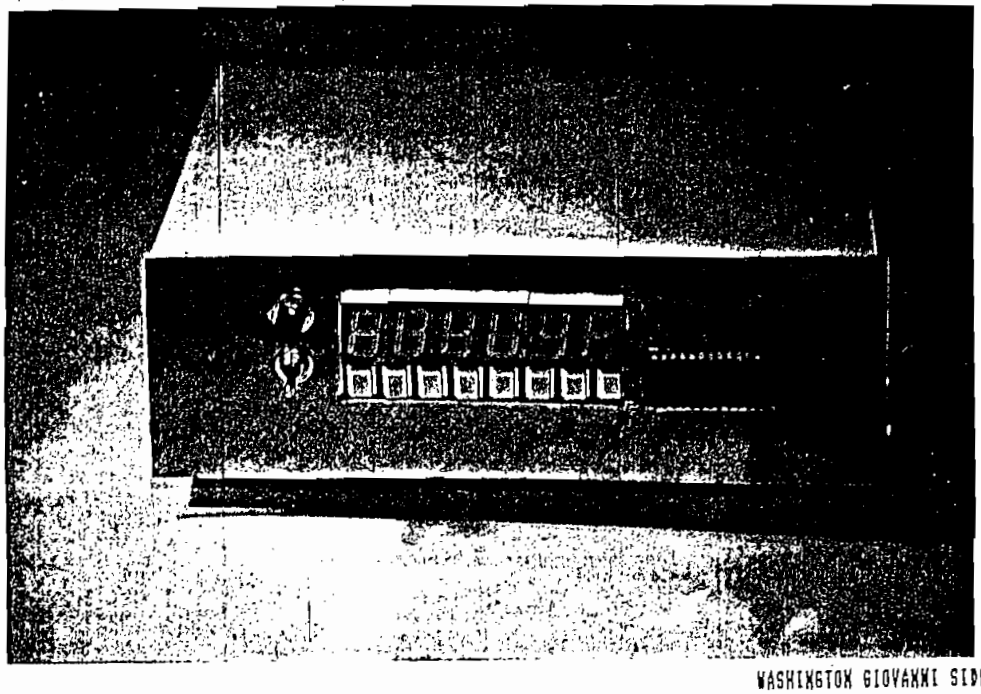


Las salidas de VCC y GND que se encuentran presentes son un paralelo directo de la fuente de alimentación, por lo tanto estas pueden soportar gran corriente.

La línea IN-ANALOG, es la entrada de datos analógicos hacia el conversor presente en el módulo.

Para tener una mayor facilidad para lo que puede ser un monitoreo o si se desea realizar algún módulo o ampliación que use unas líneas del microcontrolador que no estén contempladas en la salida de 40 postes; se cuenta asi mismo con un zócalo de 40 pines, localizado en la misma parte posterior del módulo en el cual se tiene un paralelo directo con el microcontrolador pin a pin.

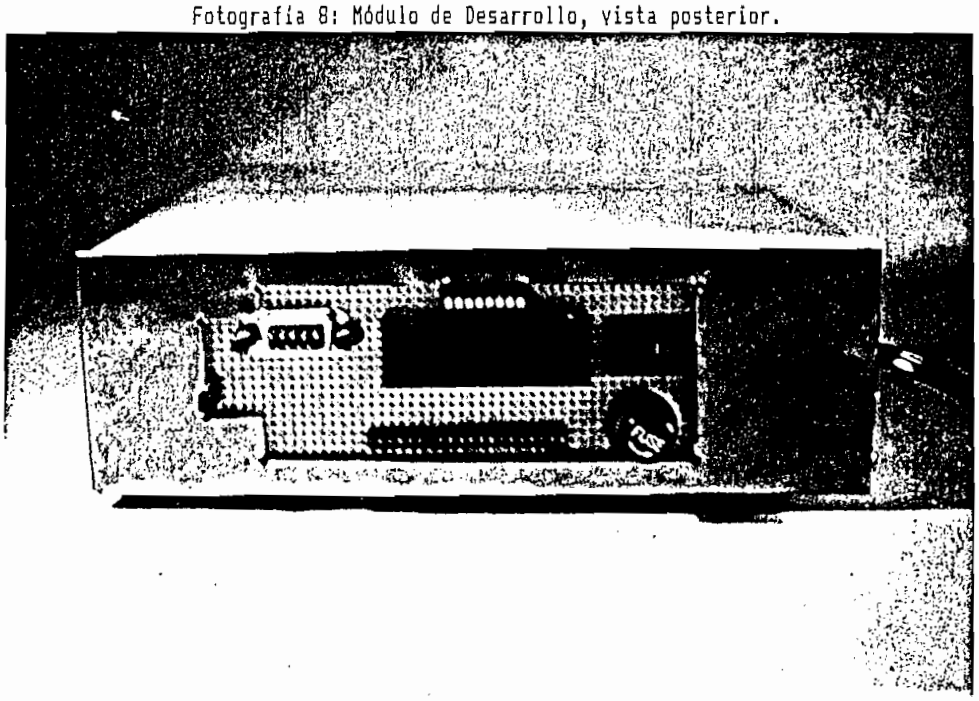
Fotografía 7: Módulo de Desarrollo operando en la función de visualización de la RAM, vista frontal.



Esto en cuanto se refiere al hardware del sistema, ahora por otra parte el software del módulo también esta realizado de manera estructurada (como se puede apreciar en el capítulo anterior), es decir que cada función específica tiene un lugar determinado dentro del programa, y funciona o realiza su tarea independientemente de las otras funciones, es decir que el programa total es un conjunto de bloques o subprogramas que si bien es cierto trabajan dentro de la misma unidad, pero no por ello pueden trabajar independientemente o en un orden totalmente distinto al presentado.

Por esta razón es que aún el programa monitor del módulo de desarrollo, puede ser optimizado, mejorado o ampliado en lo posterior, de esta manera se ofrece al usuario del sistema total libertad y facilidad para poder ampliar modularmente en hardware o software al módulo de desarrollo.

Fotografía 8: Módulo de Desarrollo, vista posterior.



4.3 NOTAS DE APLICACIÓN REFERENTES A LA FAMILIA MCS-51.

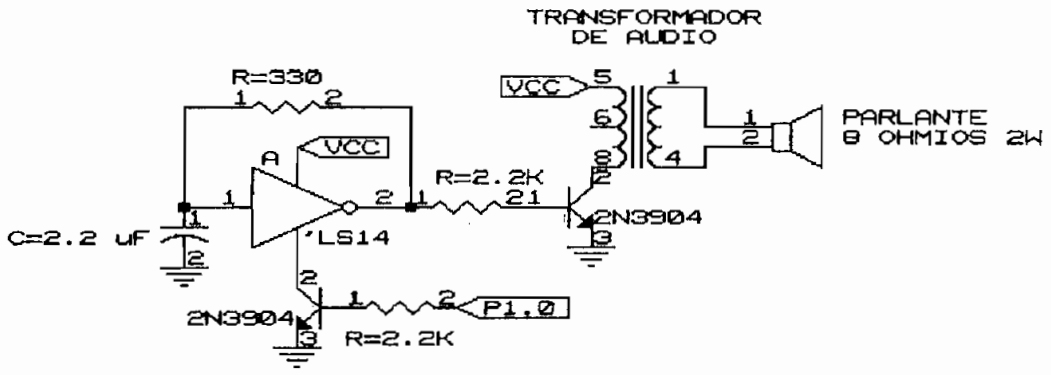
Para trabajar de una manera práctica con el módulo de desarrollo, se plantean algunos problemas de diseño muy comunes, que un usuario puede encontrar frecuentemente, y se observa la manera de resolverlo con la ayuda diseñada.

4.3.1 Reloj electrónico digital.

Este ejemplo de aplicación es uno de los problemas que se plantea dentro del Laboratorio de Sistemas Microprocesados Ahora se realiza el programa dentro del módulo para comprobar su versatilidad y apoyo al diseñador que en este caso pueden ser los estudiantes de esta materia, y adicionalmente para observar la modularidad del sistema, se realiza el programa del reloj con la condición de que este tenga alarma.

Como el módulo de desarrollo no cuenta con un elemento de que se pueda utilizar como alarma; esta se realiza en un pequeño módulo adicional. Básicamente la alarma de un reloj no es mas que un tono que se lo escucha en un pequeño parlante. Para este caso se proyecta un oscilador acoplado a un pequeño amplificador que se controle desde cualquiera de las salidas que se tiene en el módulo de desarrollo.

El esquema de la alarma proyectada se encuentra en la siguiente figura.



El funcionamiento del circuito es muy, el CI.74LS14 es un inversor Schmith Trigger actuando como oscilador, cuya salida exita al transistor amplificador dando señal al transformador de audio el cual a su vez se acopla con el parlante. Como este circuito es únicamente demostrativo no se ha incluido otras etapas de amplificación para obtener mayor señal auditiva a la salida. Para este caso la salida que controla la alarma se encuentra en el pórtico uno, y como se necesita únicamente un bit de control se escogió utitlizar el bit # 0 (P1.0). Cuando se coloca un 1L en este bit, el transistor pone la señal de GND al oscilador el cual comienza a funcionar dando señal audible en el parlante, cuando se coloca un 0L en este bit, el transistor se abre, sacando de funcionamiento al oscilador y por ende no se tendrá el tono de alarma.

El programa de ejemplo presenta en las pantallas el formato convencional de los relojes digitales en donde se puede apreciar desde la pantalla mas significativa a la menos significativa primero el punto decimal encendido que indica que se encuentra en horario de la tarde (PM), a

continuación se tiene dos dígitos que muestran las horas, la siguiente pantalla indica los segundos por lo cual esta se encuentra titilando permanentemente, en las dos pantallas siguientes se tiene la indicación de los minutos.

Para permitir igualar el reloj, se utilizan las tres primeras teclas del módulo, la tecla menos significativa es para igualar las horas; la siguiente iguala los minutos; cuando se presiona la tercera tecla se entra en el modo de igualación del reloj, y si a continuación si se presiona la tecla de los minutos o de las horas, de acuerdo a la tecla presionada el valor indicado en las pantallas, se irá incrementando hasta que se deje de presionar la tecla.

Para poner la hora de la alarma, se presiona la tecla número cuatro, cuando se presiona esta tecla nótese que el segundero del reloj desaparece y en las pantallas únicamente queda la indicación de la hora y minutos en la que se activará la alarma. Si se quisiera cambiar la hora indicada, se presiona la tecla de los minutos (tecla # 2) o de las horas (tecla # 1), hasta obtener la hora deseada.

Cuando se activa la alarma del reloj, esta al igual que la alarma de cualquier reloj suena por un determinado tiempo (un minuto para este caso), para luego callarse.

```
;            MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;                 MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:       ESCUELA POLITECNICA NACIONAL
;                    FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:     ELECTRONICA Y TELECOMUNICACIONES
;                    WASHINGTON GIOVANNI SIDEL TORRES
;                              1992-1993
;
;       PROGRAMA DE PRUEBA:  RELOJ DIGITAL CON ALARMA
;
$ALLPUBLIC
;
;   Area de inicialización de puntos de entrada de interrupciones.
;
;ETIQUETA:       OPC      OPERANDOS        ;COMENTARIOS


INICIO:            ;Definición de etiquetas
         ;Definir las direcciones de entrada/salida

STACK:  EQU      4FH      ;Dirección del Stack en 50h
SCREEN5:EQU      7FH      ;Pantalla mas significativa #5
SCREEN4:EQU      7EH      ;Pantalla # 4
SCREEN3:EQU      7DH      ;Pantalla # 3
SCREEN2:EQU      7CH      ;Pantalla # 2
SCREEN1:EQU      7BH      ;Pantalla # 1
SCREEN0:EQU      7AH      ;Pantalla # 0
SALHAB: EQU      0000H    ;Habilitación displays de salida
SALDAT: EQU      2000H    ;Salida de datos hacia Displays
TECLAS: EQU      0000H    ;Entrada de datos de las teclas
BANK0:  EQU      00H
BANK1:  EQU      08H
BANK2:  EQU      10H
BANK3:  EQU      18H
CODDIS: EQU      1300H    ;Subrutinas del usuario.
DISPLAYS:        EQU      1100H
DELAY:  EQU      1500H

         ; Programa Principal

        MOV      SP,#STACK        ;Inicializar el Stack en 50h
        MOV      B,#06H           ;Encerar las localidades de
        MOV      A,#00H           ;memoria desde la 7fh(mas sig)
        MOV      R0,#SCREEN5      ;hasta la 7ah (menos sig)
INZERO: MOV      @R0,A
        DEC      R0
        DJNZ     B,INZERO
        MOV      P1,#00H

SHOW:   MOV      R6,#00H
        MOV      R7,#00H
        CLR      44H
        CLR      45H
        CLR      46H
        CLR      47H
```

```
        MOV     PSW,#BANK1
        MOV     R4,#80H
        MOV     R5,#06H
        MOV     R6,#20H
        MOV     R7,#00H
        MOV     PSW,#BANK0
        MOV     DPTR,#2001H

CAR1:   MOV     A,DPL
CAR:    MOV     R0,#SCREEN5
        MOV     @R0,A
        DEC     R0
        DEC     R0
        MOV     A,DPH
        MOV     @R0,A
        LCALL   CODDIS
        JB      45H,ADE
        MOV     SCREEN1,#06H
        SJMP    ADE1
ADE:    MOV     SCREEN1,#00H
ADE1:   JB      46H,ADE2
        MOV     SCREEN0,#80H
        SJMP    SH0
ADE2:   MOV     SCREEN0,#00H
SH0:    MOV     SCREEN3,#40H
        LCALL   DISPLAYS

        LCALL   COMPARE

        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        CJNE    A,#0FFH,IGUAL1
        JMP     IGUAL2
IGUAL1: CJNE    A,#0F7H,IGUAL3

        PUSH    DPH
        PUSH    DPL
PRU:    MOV     PSW,#BANK1
        MOV     SCREEN3,R6
        MOV     SCREEN5,R7
        LCALL   CODDIS
        MOV     SCREEN0,R4
        MOV     SCREEN1,R5
        MOV     SCREEN3,#00H
        LCALL   DISPLAYS
        MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        CJNE    A,#0F7H,IGUAL4
        SJMP    PRU
IGUAL4: CJNE    A,#0F5H,IGUAL5
```

```
        MOV     A,#01H
        ADD     A,R7
        DA      A
        MOV     R7,A
        CJNE    A,#60H,JUMP
        MOV     R7,#00H
JUMP:   SJMP    PRU

IGUAL5: CJNE    A,#0F6H,IGUAL6
        MOV     A,#10H
        ADD     A,R6
        DA      A
        MOV     R6,A
        CJNE    A,#20H,IGUALA
        CJNE    R5,#06H,IGUALA
        CJNE    R4,#80H,IGUAL8
        MOV     R4,#00H
        SJMP    IGUALA
IGUAL8: MOV     R4,#80H

IGUALA: CJNE    A,#30H,IGUAL7
        CJNE    R5,#06H,IGUAL7
        MOV     R5,#00H
        MOV     R6,#10H

IGUAL7: CJNE    A,#00H,IGUAL9
        MOV     R5,#06H
IGUAL9: JMP     PRU

IGUAL6: MOV     PSW,#BANK0
        POP     DPL
        POP     DPH
        JMP     CAR1

IGUAL3: CJNE    A,#0F9H,MINUT
        SETB    47H
        MOV     B,#05H
RETAR1: LCALL   DELAY
        DJNZ    B,RETAR1
        JMP     MINUT1
MINUT:  CJNE    A,#0FAH,IGUAL2
        MOV     B,#05H
RETAR2: LCALL   DELAY
        DJNZ    B,RETAR2
        JMP     HORA
IGUAL2: INC     R7
        CJNE    R7,#7H,SHO5

        MOV     SCREEN3,#00H
        MOV     R7,#00H
SHO1:   LCALL   DISPLAYS
        JNB     44H,SIN
        MOV     P1,#00H
        CLR     44H
```

```
SIN:    INC     R7
        CJNE    R7,#5H,SHO1
        MOV     R7,#00H
        MOV     A,#01H
        CLR     C
        ADDC    A,R6
        DA      A
        MOV     R6,A
        CJNE    R6,#58H,SHO5
        SJMP    MINUT1
SHO5:   JMP     SHO

MINUT1: CLR     C
        MOV     A,#01H
        ADDC    A,DPL
        DA      A
        MOV     DPL,A
        MOV     R6,#00H
        CJNE    A,#59H,CAR8
        MOV     DPL,#00H
        JBC     47H,CAR8
HORA:   MOV     A,#10H
        CLR     C
        ADDC    A,DPH
        DA      A
        MOV     DPH,A
        CJNE    A,#00H,SIGA
        CLR     45H
SIGA:   CJNE    A,#20H,SIGA1
        MOV     A,SCREEN1
        CJNE    A,#06H,SIGA1
        CPL     46H
        MOV     SCREEN0,#00H

SIGA1:  MOV     R0,#SCREEN1
        MOV     A,@R0
        CJNE    A,#06H,CAR9
        MOV     A,DPH
        ANL     A,#0F0H
        SWAP    A
        CJNE    A,#03H,CAR9
        MOV     DPH,#10H
        MOV     SCREEN1,#00H
        SETB    45H
CAR9:   JMP     CAR1
CAR8:   CLR     47H
        JMP     CAR

        SJMP    $

COMPARE: *      PUSH    ACC
        MOV     A,SCREEN0
        MOV     PSW,#BANK1
        CLR     C
        SUBB    A,R4
```

```
        CJNE    A,#00H,NOPIT
        CLR     C
        MOV     A,SCREEN1
        SUBB    A,R5
        CJNE    A,#00H,NOPIT
        CLR     C
        MOV     A,DPH
        SUBB    A,R6
        CJNE    A,#00H,NOPIT
        CLR     C
        MOV     A,DPL
        SUBB    A,R7
        CJNE    A,#00H,NOPIT
        SETB    44H
        LCALL   SOUND

NOPIT:  MOV     PSW,#BANK0
        POP     ACC
        RET
SOUND:  MOV     P1,#2FH
        RET

        END
```

Para realizar otro ejemplo de aplicación, se aprovecha que se ya se tiene el generador de tonos para plantear.

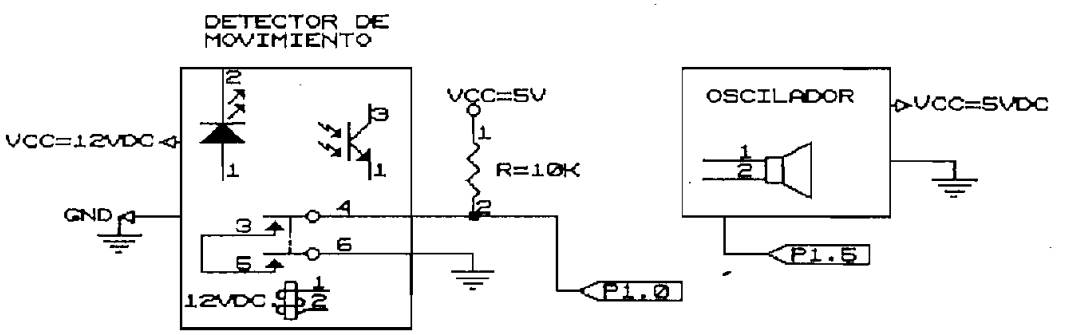
4.3.2 Alarma contra intrusos.

Mediante el uso de una fotocelda, se plantea realizar una alarma contra intrusos, de tal suerte que cuando la fotocelda se active debido a la presencia de alguna persona se tenga una señal audible de alarma.

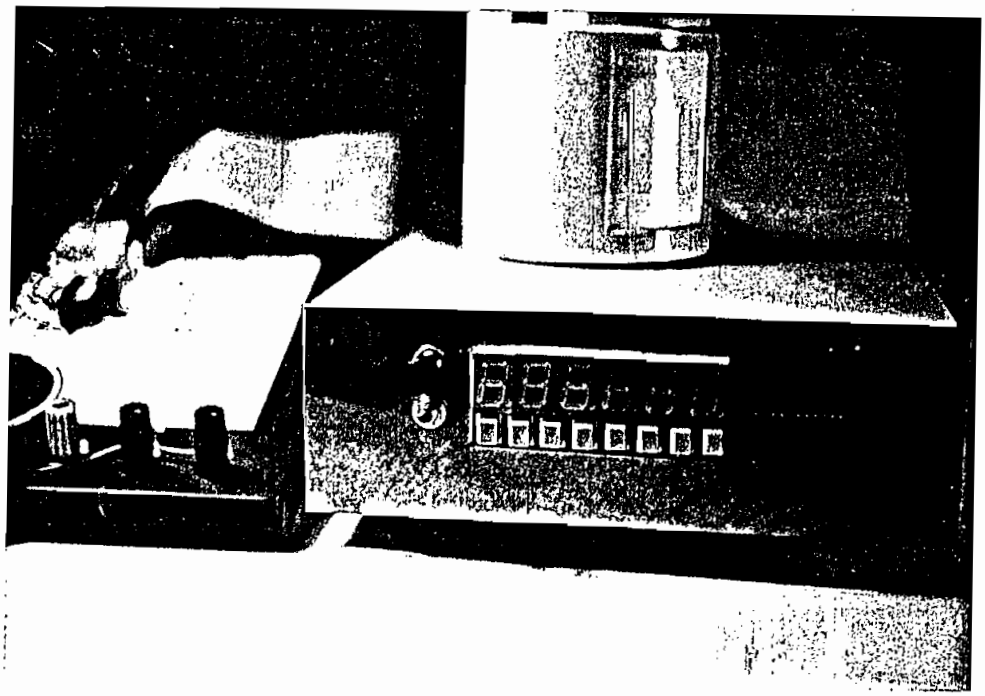
Revisando las características de la fotocelda, se tiene que su fuente de alimentación es de 12VDC, al instante de energizarla esta tiene un tiempo de estabilización de aproximadamente 3 minutos, lo cual se tiene que tener en cuenta en el programa para que al prender la alarma no se

active inmediatamente la señal audible.

En la implementación del hardware utilizamos el mismo oscilador del ejemplo anterior, con el mismo sistema de control, ahora la fotocelda (detector de movimiento), consta básicamente del elemento detector que se polariza con un voltaje DC de 12V, el cual cuando entra en operación abre un pequeño relé, en cambio en estado pasivo el relé se encuentra cerrado. El detector opera momentáneamente, es decir cuando detecta movimiento abre el relé para luego cerrarlo si la causa que accionó el detector ha cesado, lo cual se debe considerar en el programa. El circuito implementado es el siguiente.



Fotografía 9: Módulo de Desarrollo, ejecutando el programa de Alarma.



Nótese que de acuerdo al circuito implementado, cuando el relé se encuentra cerrado en el lado de P1.0 se tiene un 0L, mientras que cuando se abre se tendrá un 1L.

Dentro del programa, primero se realiza un lazo de espera hasta cuando el detector se estabilice, esto se lo hace verificando que P1.0 pase de uno a cero lógico, luego se entra a una pantalla que indica "ALARMA" como señal que el detector esta estabilizado, para activar la alarma se oprime la primera tecla, en la pantalla se presenta el mensaje de "ACTIVADO", ahora el programa esta en un lazo constante de monitoréo de la señal P1.0, si esta cambia de valor, se activa la señal del oscilador dando la alarma en un lazo de repetición que puede ser terminado presionando la tecla número dos.

El programa de la alarma se presenta a continuación:

```
;           MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS
;                MICROPROCESADORES DE LA FAMILIA MCS-51
;TESIS DE GRADO:       ESCUELA POLITECNICA NACIONAL
;                    FACULTAD DE INGENIERIA ELECTRICA
;ESPECIALIZACION:     ELECTRONICA Y TELECOMUNICACIONES
;                    WASHINGTON GIOVANNI SIDEL TORRES
;                              1992-1993
;
;       PROGRAMA DE PRUEBA:  ALARMA CONTRA INTRUSOS
;
$ALLPUBLIC
;
;   Area de inicialización de puntos de entrada de interrupciones.
;
;ETIQUETA:       OPC      OPERANDOS        ;COMENTARIOS


INICIO:           ;Definición de etiquetas
         ;Definir las direcciones de entrada/salida

STACK:  EQU      4FH      ;Dirección del Stack en 50h
SCREEN5:EQU      7FH      ;Pantalla mas significativa #5
SCREEN4:EQU      7EH      ;Pantalla # 4
SCREEN3:EQU      7DH      ;Pantalla # 3
SCREEN2:EQU      7CH      ;Pantalla # 2
SCREEN1:EQU      7BH      ;Pantalla # 1
SCREEN0:EQU      7AH      ;Pantalla # 0
SALHAB: EQU      0000H    ;Habilitación displays de salida
SALDAT: EQU      2000H    ;Salida de datos hacia Displays
TECLAS: EQU      0000H    ;Entrada de datos de las teclas
DISPLAYS:        EQU      1000H
MESSAGE:         EQU      1600H

         ; Programa Principal

        MOV      SP,#STACK        ;Inicializar el Stack en 50h
        MOV      B,#06H           ;Encerar las localidades de
        MOV      A,#00H           ;memoria desde la 7fh(mas sig)
        MOV      R0,#SCREEN5      ;hasta la 7ah (menos sig)
INZERO: MOV      @R0,A
        DEC      R0
        DJNZ     B,INZERO
        CLR      50H
        CLR      55H
        MOV      R7,#00H
        CLR      P1.5
BLOQUEO:         MOV      DPTR,#INIC
        LCALL    MESSAGE
        MOV      A,P1
        ANL      A,#01H
        CJNE     A,#00H,BLOQUEO

ALARM1: MOV      DPTR,#ALARM
        LCALL    MESSAGE
        MOV      DPTR,#TECLAS
```

```
        MOVX    A,@DPTR
        CJNE    A,#0FFH,DECTEC
        SJMP    ALARM1

;   A continuación se realiza la decodificación de la tecla presionada

DECTEC: CJNE    A,#0FEH,GOO
ACTIVO: MOV     DPTR,#ACTIV
        LCALL   MESSAGE
        JB      55H,ACTIV2

        MOV     A,P1
        ANL     A,#01H
        CJNE    A,#00H,SUENE

ACTIV2: JNB     55H,ACTIV3
        INC     R7
        CJNE    R7,#02H,ACTIV3
        MOV     R7,#00H
        CPL     P1.5

ACTIV3: MOV     DPTR,#TECLAS
        MOVX    A,@DPTR
        CJNE    A,#0FDH,ACTIVO
        SJMP    GOO

SUENE:  SETB    P1.5
        SETB    55H
        SJMP    ACTIVO

GOO:    CJNE    A,#0FDH,ALARM1
        CLR     P1.5
        CLR     55H
        JMP     ALARM1

INIC:   DB      40H,40H,40H,40H,40H,40H
ALARM:  DB      0DCH,54H,50H,0DCH,06H,0DCH
ACTIV:  DB      B0H,1CH,04H,78H,58H,0DCH

        END
```

Mediante estos ejemplos de aplicación tan diferentes uno de otro se puede dar cuenta que efectivamente el módulo de desarrollo puede adaptarse a diferentes sistemas y acoplarse a ellos de manera optima, con lo cual se demuestra su modularidad y facilidad de ampliación.

4.4 CONCLUSIONES Y RECOMENDACIONES.

Del diseño y construcción del módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51 se ha obtenido las siguientes conclusiones:

- El hardware implementado en el presente trabajo es de propósito general, con las perspectivas de usarse en cualquier sistema dado ó en su defecto acoplarse al mismo mediante una de sus entradas o salidas de datos de tal suerte que cualquier usuario de este sistema encuentre en el mismo un apoyo a sus requerimientos de diseño brindándole un soporte práctico para que la solución buscada sea obtenida de manera rápida y eficiente.
- El programa de control del módulo, es de fácil análisis hecho de manera modular es decir cada función de ayuda al usuario del módulo se encuentra perfectamente definida dentro de una subrutina de tal suerte que si el diseñador quisiera usar una de estas funciones; puede hacerlo sin ningún problema, la filosofía fundamental del presente trabajo es de apoyar de manera incondicional a cualquier usuario del sistema desarrollado.

  Adicionalmente el programa de control del módulo se lo presenta tanto de manera escrita detallando las variables de uso con sus respectivo diagrama de flujo, como en un diskette anexo al presente trabajo separando cada subrutina en un archivo definido, y de manera

global, de tal suerte que el usuario esta en plena capacidad de usar una o varias funciones del módulo o modificarla de acuerdo a su conveniencia.

- Las opciones de ayuda que se presenta dentro del ambiente del módulo de desarrollo, son funciones básicas y asi mismo de propósito general implantadas de acuerdo a la experiencia y sugerencia de diversos tipos de usuarios que en el transcurso del diseño de un programa o un hardware específico se ha visto la necesidad de contar con una opción de ayuda de este tipo.
- Nótese la gran potencialidad que el módulo tiene como herramienta de desarrollo y depuración de un programa o un hardware, ya que con este tipo de ayuda y sus diferentes opciones poner a trabajar un sistema y luego depurarlo no representa mayor problema, ahorrando tiempo y esfuerzo.

  Ejecutar un programa en tiempo real y visualizar cualquier resultado es una de las grandes ventajas que dentro del módulo se puede obtener, dejando a un lado simulaciones poco objetivas que entorpecen en gran medida la depuración de cualquier programa.
- Trabajar en el ambiente del módulo ofrece la objetividad que todo diseñador debe de tener para el desarrollo de sus propios sistemas, y debido a la modularidad y facilidad de ampliación con la que se cuenta, se recomienda en lo posterior realizar nuevos módulos que acoplados aumenten y potencialicen las ventajas que en este trabajo se presentan.

- Como se mencionó anteriormente el programa de control del módulo esta abierto hacia los usuarios, y las opciones que en el se presentan son de propósito general; otra sugerencia podría ser la realización de un programa especializado a una aplicación específica.
- El Módulo de Desarrollo para sistemas basados en los microcontroladores de la familia MCS-51/52, esta realizado con elementos disponibles en el mercado nacinal, que es uno de los factores primordiales para realizar un circuito.
- El trabajo que se presenta en esta Tésis es apenas una pequeña parte de todo lo que se podría proyectar, programar o diseñar; por este motivo se deja la , inquietud y el reto para trabajar con este tipo de elementos y tratar de ampliar un poco mas el gran campo que ofrecen los microcontroladores.

CAPITULO V

APENDICES Y ANEXOS

5.1 - Apéndice I: Manual de la familia MCS-51

5.2 - Apéndice II: Manual del usuario del módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51

5.3 - Anexo I: Tarjeta MCPD51DA.

5.4 - Anexo II: Diskette del sistema y datos, conteniendo programas del módulo y ejemplos de aplicación y pruebas del mismo.

5.1 – APENDICE I: MANUAL DE LA FAMILIA MCS-51/52.

8031/8051/8751
SINGLE COMPONENT 8-BIT MICROCOMPUTER

- 8031 - Control Oriented CPU With RAM and I/O
- 8051 - An 8031 With Factory Mask-Programmable ROM
- 8751 - An 8031 With User Programmable/Erasable EPROM

- 4K x 8 ROM/EPROM
- 128 x 8 RAM
- Four 8-Bit Ports, 32 I/O Lines
- Two 16-Bit Timer/Event Counters
- High-Performance Full-Duplex Serial Channel
- External Memory Expandable to 128K
- Compatible with MCS-80®/MCS-85® Peripherals
- Boolean Processor
- MCS-48® Architecture Enhanced with:
  - Non-Paged Jumps
  - Direct Addressing
  - Four 8-Register Banks
  - Stack Depth Up to 128-Bytes
  - Multiply, Divide, Subtract, Compare
- Most Instructions Execute in 1μs
- 4μs Multiply and Divide

The Intel® 8031/8051/8751 is a stand-alone, high-performance single-chip computer fabricated with Intel's highly-reliable +5 Volt, depletion-load, N-Channel, silicon-gate HMOS technology and packaged in a 40-pin DIP. It provides the hardware features, architectural enhancements and new instructions that are necessary to make it a powerful and cost effective controller for applications requiring up to 64K bytes of program memory and/or up to 64K bytes of data storage.

The 8051/8751 contains a non-volatile 4K x 8 read only program memory; a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits. The 8031 is identical, except that it lacks the program memory. For systems that require extra capability, the 8051 can be expanded using standard TTL compatible memories and the byte oriented MCS-80 and MCS-85 peripherals.

The 8051 microcomputer, like its 8048 predecessor, is efficient both as a controller and as an arithmetic processor. The 8051 has extensive facilities for binary and BCD arithmetic and excels in bit-handling capabilities. Efficient use of program memory results from an instruction set consisting of 44% of one-byte, 41% two-byte, and 15% three-byte instructions. With a 12 MHz crystal, 58% of the instructions execute in 1μs, 40% in 2μs and multiply and divide require only 4μs. Among the many instructions added to the standard 8048 instruction set are multiply, divide, subtract and compare.

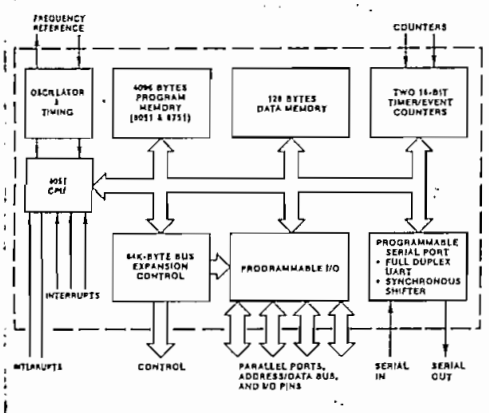


Figure 1.
Block Diagram

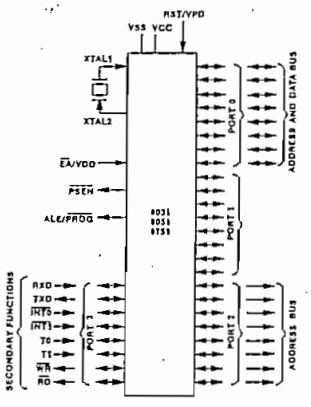


Figure 2.
Logic Symbol

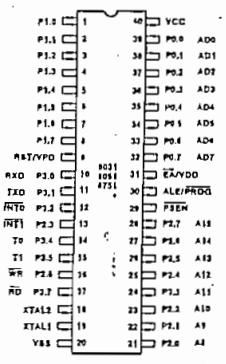


Figure 3. Pin
Configuration

INTRODUCTION

This data sheet provides an introduction to the 8051 family. A detailed description of the hardware required to expand the 8051 with more program memory, data memory, I/O, specialized peripherals and into multiprocessor configurations is described in the 8051 Family User's Manual.

THE 8051 Family

The 8051 is a stand-alone high-performance single-chip computer intended for use in sophisticated real-time applications such as instrumentation, industrial control and intelligent computer peripherals. It provides the hardware features, architectural enhancements and new instructions that make it a powerful and cost effective controller for applications requiring up to 64K-bytes of program memory and/or up to 64K-bytes of data storage. A Block Diagram is shown in Figure 3.

The 8031 is a control-oriented CPU without on-chip program memory. It can address 64K-bytes of external Program Memory in addition to 64K-bytes of External Data Memory. For systems requiring extra capability, each member of the 8051 family can be expanded using standard memories and the byte oriented MCS-80 and MCS-85 peripherals. The 8051 is an 8031 with the lower 4K-bytes of Program Memory filled with on-chip mask programmable ROM while the 8751 has 4K-bytes of UV-light-erasable/electrically-programmable ROM.

The three pin-compatible versions of this component reduce development problems to a minimum and provide maximum flexibility. The 8751 is well suited for development, prototyping, low-volume production and applications requiring field updates; the 8051 for low-cost, high volume production; and the 8031 for applications desiring the flexibility of external Program Memory which can be easily modified and updated in the field.

MACRO-VIEW OF THE 8051 ARCHITECTURE

On a single die the 8051 microcomputer combines CPU; non-volatile 4K x 8 read-only program memory; volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and clock circuits. This section will provide an overview of the 8051 by providing a high-level description of its major elements: the CPU architecture and the on-chip functions peripheral to the CPU. The generic term "8051" is used to refer collectively to the 8031, 8051, and 8751.

8051 CPU Architecture

The 8051 CPU manipulates operands in four memory spaces. These are the 64K-byte Program Memory, 64K-byte External Data Memory, 384-byte Internal Data Memory and 16-bit Program Counter spaces. The Internal Data Memory address space is further divided into the 256-byte Internal Data RAM and 128-byte Special Function Register (SFR) address spaces shown in Figure 4. Four Register Banks (each with eight registers), 128 addressable bits, and the stack reside in the Internal Data RAM. The stack depth is limited only by the available Internal Data RAM and its location is determined by the 8-bit stack pointer. All registers except the Program Counter and the four 8-Register Banks reside in the Special Function Register address space. These memory mapped registers include arithmetic registers, pointers, I/O ports, interrupt system registers, timers and serial port. 128 bit locations in the SFR address space are addressable as bits. The 8051 contains 128 bytes of Internal Data RAM and 20 SFRs.

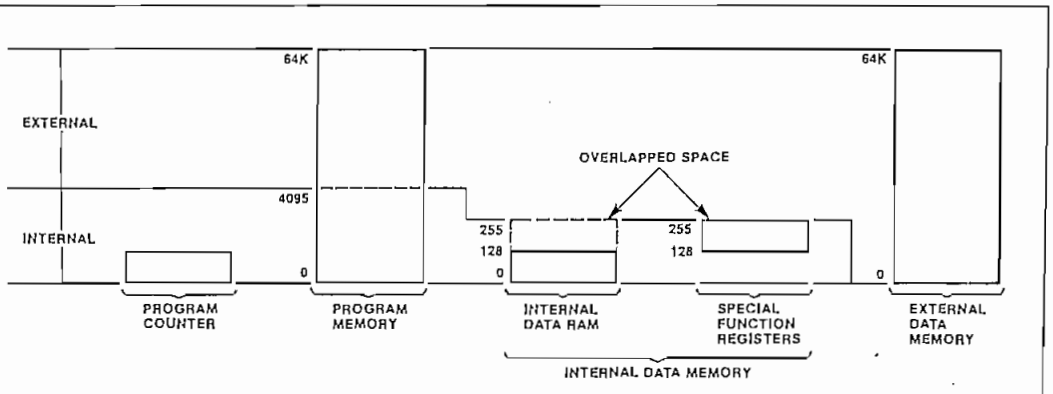


Figure 4. 8051 Family Memory Organization

The 8051 provides a non-paged Program Memory address space to accommodate relocatable code. Conditional branches are performed relative to the Program Counter. The register-indirect jump permits branching relative to a 16-bit base register with an offset provided by an 8-bit index register. Sixteen-bit jumps and calls permit branching to any location in the contiguous 64K Program Memory address space.

The 8051 has five methods for addressing source operands: Register, Direct, Register-Indirect, Immediate and Base-Register- plus Index-Register-Indirect Addressing. The first three methods can be used for addressing destination operands. Most instructions have a "destination, source" field that specifies the *data* type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand.

Registers in the four 8-Register Banks can be accessed through Register, Direct, or Register-Indirect Addressing; the 128 bytes of Internal Data RAM through Direct or Register-Indirect Addressing; and the *Special Function* Registers through Direct Addressing. External Data Memory is accessed through Register-Indirect Addressing. Look-Up-Tables resident in Program Memory can be accessed through Base-Register- plus Index-Register- Indirect Addressing.

The 8051 is classified as an 8-bit machine since the internal ROM, RAM, Special Function Registers, Arithmetic/Logic Unit and external data bus are each 8-bits wide. The 8051 performs operations on bit, nibble, byte and double-byte data types.

The 8051 has extensive facilities for byte transfer, logic, and integer arithmetic operations. It excels at bit handling since data transfer, logic and conditional branch operations can be performed directly on Boolean variables.

The 8051's instruction set is an enhancement of the instruction set familiar to MCS-48 users. It is enhanced to allow expansion of on-chip CPU peripherals and to optimize byte efficiency and execution speed. Op codes were reassigned to add new high-power operations and to permit new addressing modes which make the old operations more orthogonal. Efficient use of program memory results from an instruction set consisting of 49 single-byte, 45 two-byte and 17 three-byte instructions. When using a 12 MHz oscillator, 64 instructions execute in 1μs and 45 instructions execute in 2μs. The remaining instructions (multiply and divide) execute in only 4μs. The number of bytes in each instruction and the number of oscillator periods required for execution are listed in the appended 8051 Instruction Set Summary.

On-Chip Peripheral Functions

Thus far only the CPU and memory spaces of the 8051 have been described. In addition to the CPU and memories, an interrupt system, extensive I/O facilities, and several peripheral functions are integrated on-chip to relieve the CPU of repetitious, complicated or time-critical tasks and to permit stringent real-time control of external system interfaces. The extensive I/O facilities include the I/O pins, parallel I/O ports, bidirectional address/data bus and the serial port for I/O expansion. The CPU peripheral functions integrated on-chip are the two 16-bit counters and the serial port. All of these work together to greatly boost system performance.

INTERRUPT SYSTEM

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, a sophisticated multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency ranges from 3μs to 7μs when using a 12 MHz crystal.

The 8051 acknowledges interrupt requests from five sources: Two from external sources via the $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ pins, one from each of the two internal counters and one from the serial I/O port. Each interrupt vectors to a separate location in Program Memory for its service program. Each of the five sources can be assigned to either of two priority levels and can be independently enabled and disabled. Additionally all enabled sources can be globally disabled or enabled. Each external interrupt is programmable as either level- or transition-activated and is active-low to allow the "wire or-ing" of several interrupt sources to the input pin. The interrupt system is shown diagrammatically in Figure 5.

I/O FACILITIES

The 8051 has instructions that treat its 32 I/O lines as 32 individually addressable bits and as four parallel 8-bit ports addressable as Ports 0, 1, 2 and 3. Ports 0, 2 and 3 can also assume other functions. Port 0 provides the multiplexed low-order address and data bus used for expanding the 8051 with standard memories and peripherals. Port 2 provides the high-order address bus when expanding the 8051 with external Program Memory or more than 256 bytes of External Data Memory. The pins of Port 3 can be configured individually to provide external interrupt request inputs, counter inputs, the serial port's receiver input and transmitter output, and to generate the control signals used for reading and writing External Data Memory. The generation or use of an alternate function on a Port 3 pin is done automatically by the 8051 as long as the pin

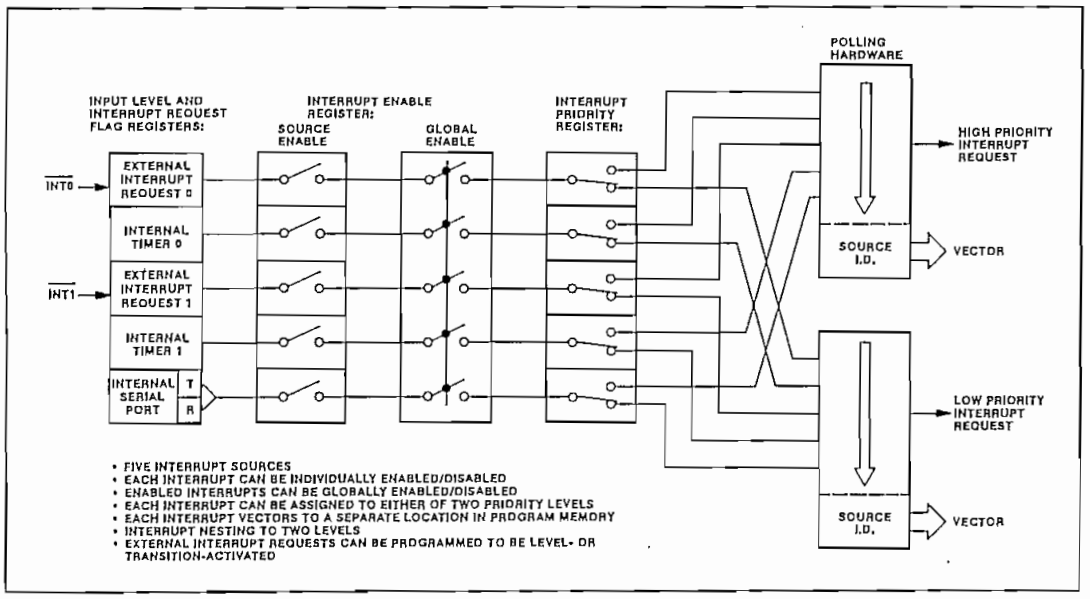


Figure 5. 8051 Interrupt System

is configured as an input. The configuration of the ports is shown on the 8051 Family Logic Symbol of Figure 2.

Open Drain I/O Pins

Each pin of Port 0 can be configured as an open drain output or as a high impedance input. Resetting the microcomputer programs each pin as an input by writing a one (1) to the pin. If a zero (0) is later written to the pin it becomes configured as an output and will continuously sink current. Re-writing the pin to a one (1) will place its output driver in a high-impedance state and configure the pin as an input. Each I/O pin of Port 0 can sink two TTL loads.

Quasi-Bidirectional I/O Pins

Ports 1, 2 and 3 are quasi-bidirectional buffers. Resetting the microcomputer programs each pin as an input by writing a one (1) to the pin. If a zero (0) is later written to the pin it becomes configured as an output and will continuously sink current. Any pin that is configured as an output will be reconfigured as an input when a one (1) is written to the pin. Simultaneous to this reconfiguration the output driver of the quasi-bidirectional port will source current for two oscillator periods. Since current is sourced only when a bit previously written to a zero (0) is updated to a one (1), a pin programmed as an input will not source current into the TTL gate that is driving it if the pin is later written with another one (1). Since the quasi-bidirectional output driver sources current for only two oscillator periods, an internal pullup resistor of approximately 20K- to 40K-ohms is provided to hold the external driver's loading at a TTL high level. Ports 1, 2 and 3 can sink/source one TTL load.

Microprocessor Bus

A microprocessor bus is provided to permit the 8051 to solve a wide range of problems and to allow the upward growth of user products. This multiplexed address and data bus provides an interface compatible with standard memories, MCS-80 peripherals and the MCS-85 memories that include on-chip programmable I/O ports and timing functions. These are summarized in the 8051 Microcomputer Expansion Components chart of Figure 6.

When accessing external memory the high-order address is emitted on Port 2 and the low-order address on Port 0. The ALE signal is provided for strobing the address into an external latch. The program store enable ($\overline{PSEN}$) signal is provided for enabling an external memory device to Port 0 during a read from the Program Memory address space. When the MOVX instruction is executed Port 3 automatically generates the read ($\overline{RD}$) signal for enabling an External Data Memory device to Port 0 or generates the write ($\overline{WR}$) signal for strobing the external memory device with the data emitted by Port 0. Port 0 emits the address and data to the external memory through a push/pull driver that can sink/source two TTL loads. At the end of the read/write bus cycle Port 0 is automatically reprogrammed to its high

Compatible MCS-80/85 Components

Category	I.D.	Description	Comments	Program Or Data Memory	Crystal Frequency MHz (Max)
I/O Expander		8 Line I/O Expander (Shift Register)	Low Cost I/O Expander		12
Standard EPROMs	2758	1K x 8 450 ns Light Erasable	User programmable and erasable.	P	9
	2716-1	2K x 8 350 ns Light Erasable		P	11
	2732	4K x 8 450 ns Light Erasable		P	9
	2732A	4K x 8 250 ns Light Erasable		P	12
Standard RAMs	2114A	1K x 4 100 ns RAM	Data memory can be easily expanded using standard NMOS RAMs.	D	12
	2148	1K x 4 70 ns RAM		D	12
	2142-2	1K x 4 200 ns RAM		D	12
Multiplexed Address/ Data RAMs	8185A	1K x 8 300 ns RAM		D	12
Standard I/O	8212	8-Bit I/O Port	Serves as Address Latch or I/O port.	D	12
	8282	8-Bit I/O Port		D	12
	8283	8-Bit I/O Port		D	12
	8255A	Programmable Peripheral Interface	Three 8-bit programmable I/O ports.	D	12
	8251A	Programmable Communications Interface	Serial Communications Receiver/Transmitter.	D	12
Standard Peripherals	8205	1 of 8 Binary Decoder	MCS-80 and MCS-85 peripheral devices are compatible with the 8051 allowing easy addition of specialized interfaces. Future MCS-80/85 devices will also be compatible.	D	12
	8286	Bi-directional Bus Driver		D	12
	8287	Bi-directional Bus Driver (Inverting)		D	12
	8253A	Programmable Interval Timer		D	12
	8279	Programmable Keyboard/Display Interface (128 Keys)		D	12
	8291	GPIB Talker/Listener		D	12
	8292	GPIB Controller		D	11.7
Universal Peripheral Interfaces	8041A	ROM Program Memory	User programmable to perform custom I/O and control functions.	D/P	12/11.7
	8741A	EPROM Program Memory		D/P	12/11.7
Memories with on-chip I/O and Peripheral Functions.	8155-2	256 x 8 330 ns RAM		D	12
	8355-2	2K x 8 330 ns ROM		P	11.6
	8755-2	2K x 8 330 ns EPROM		P	11.6

Figure 6. 8051 Microcomputer Expansion Components

impedance state and Port 2 is returned to the state it had prior to the bus cycle. The 8051 generates the address, data and control signals needed by memory and I/O devices in a manner that minimizes the requirements placed on external program and data memories. At 12 MHz, the Program Memory cycle time is 500ns and the access times required from stable address and $\overline{PSEN}$ are approximately 320ns and 150ns respectively. The External Data Memory cycle time is 1μs and the access times required from stable address and from read ($\overline{RD}$) or write ($\overline{WR}$) command are approximately 600ns and 250ns respectively.

TIMER/EVENT COUNTERS

The 8051 contains two 16-bit counters for measuring time intervals, measuring pulse widths, counting events and generating precise, periodic interrupt requests. Each can be programmed independently to operate similar to an 8048 8-bit timer with divide by 32 prescaler or as an 8-bit counter with divide by 32 prescaler (Mode 0), as a 16-bit time-interval or event counter (Mode 1), or as an 8-bit time-interval or event counter with automatic reload upon overflow (Mode 2).

Additionally, counter 0 can be programmed to a mode that divides it into one 8-bit time-interval or

event counter and one 8-bit time-interval counter (Mode 3). When counter 0 is in Mode 3, counter 1 can be programmed to any of the three aforementioned modes, although it cannot set an interrupt request flag or generate an interrupt. This mode is useful because counter 1's overflow can be used to pulse the serial port's transmission-rate generator. Along with their multiple operating modes and 16-bit precision, the counters can also handle very high input frequencies. These range from 0.1 MHz to 1.0 MHz (for 1.2 MHz to 12 MHz crystal) when programmed for an input that is a division by 12 of the oscillator frequency and from 0 Hz to an upper limit of 50 KHz to 0.5 MHz (for 1.2 MHz to 12 MHz crystal) when programmed for external inputs. Both internal and external inputs can be gated to the counter by a second external source for directly measuring pulse widths.

The counters are started and stopped under software control. Each counter sets its interrupt request flag when it overflows from all ones to all zeros (or auto-reload value). The operating modes and input sources are summarized in Figures 7 and 8. The effects of the *configuration flags* and the status flags are shown in Figures 9 and 10.

Serial Communications

The 8051 has a serial I/O port that is useful for serially linking peripheral devices as well as multiple 8051s through standard asynchronous protocols with full-duplex operation. The serial port also has a synchronous mode for expansion of I/O lines using CMOS and TTL shift registers. This hardware serial communications interface saves ROM code and permits a much higher transmission rate than could be achieved through software. In response to a serial port interrupt request the CPU has only to read/write the serial port's buffer to service the serial link. A block diagram of the serial port is shown in Figures 11 and 12. Methods for linking UART (universal asynchronous receiver/transmitter) devices are

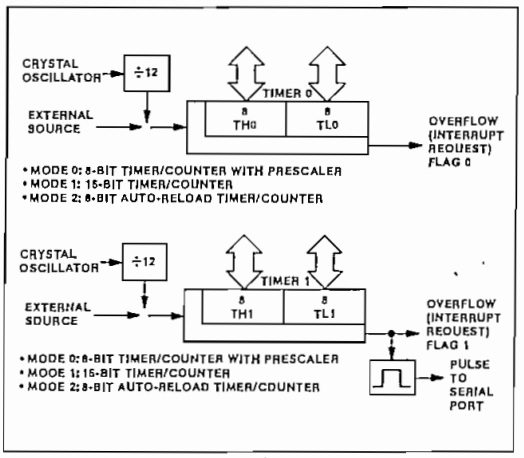


Figure 7. Timer/Event Counter Modes 0, 1 and 2

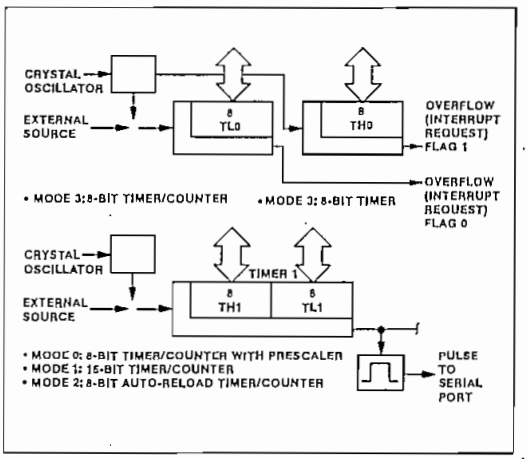


Figure 8. Timer/Event Counter 0 in Mode 3

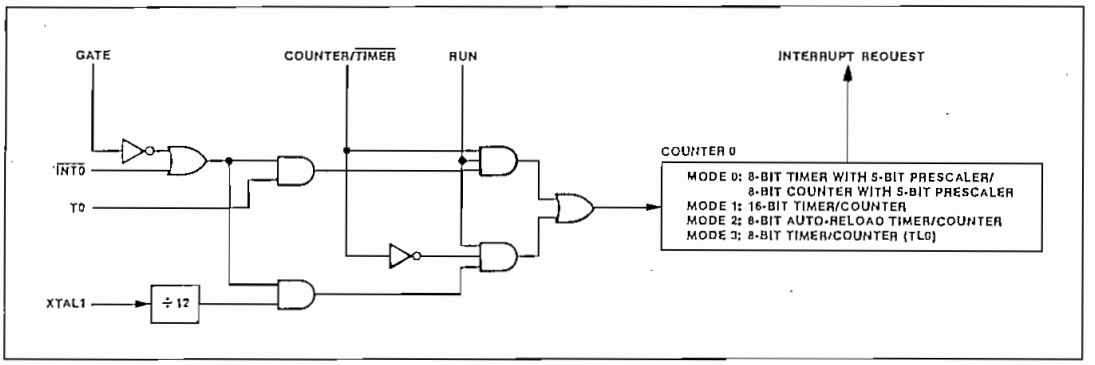


Figure 9. Timer/Counter 0 Control and Status Flag Circuitry

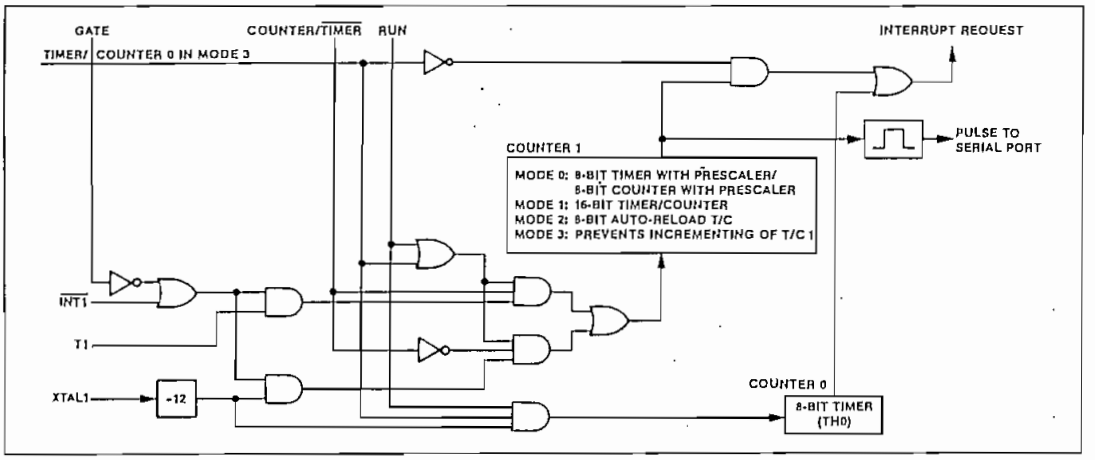


Figure 10. Timer/Counter 1 Control and Status Flag Circuitry

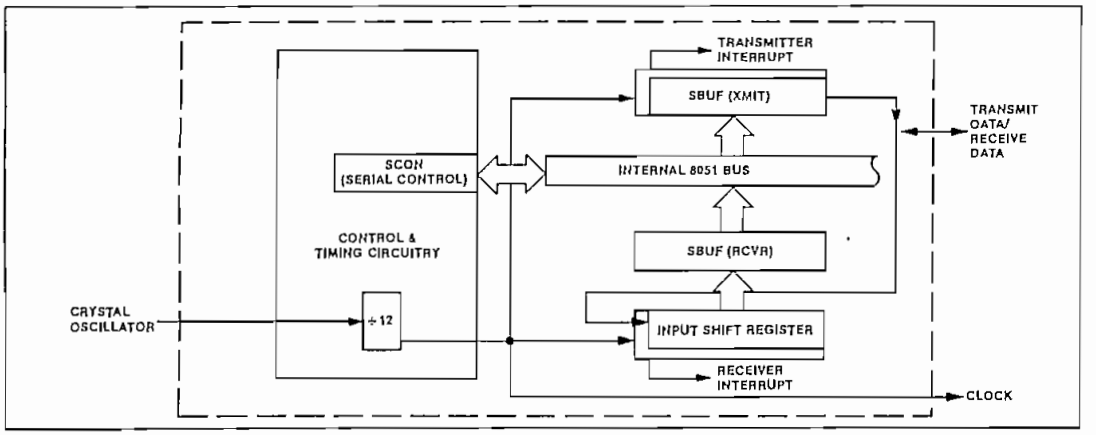


Figure 11. Serial Port—Synchronous Mode 0

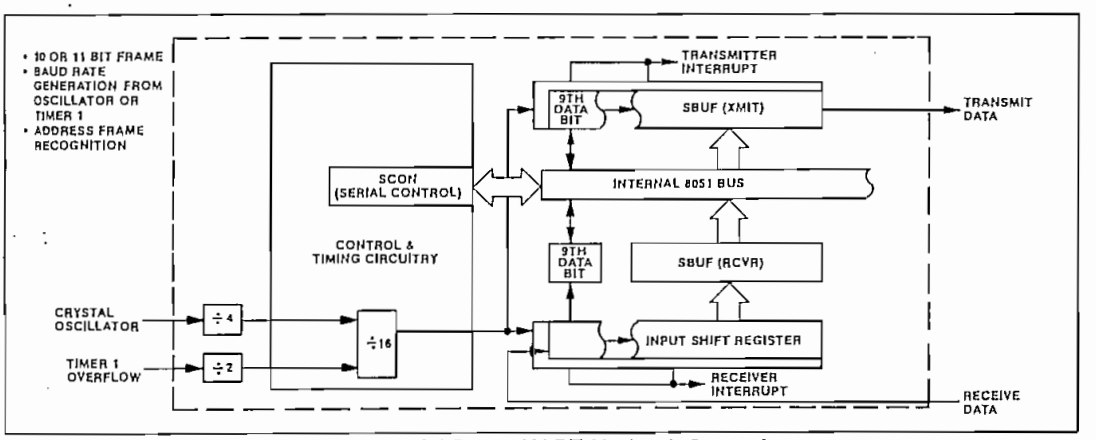


Figure 12. Serial Port—UART Modes 1, 2, and 3

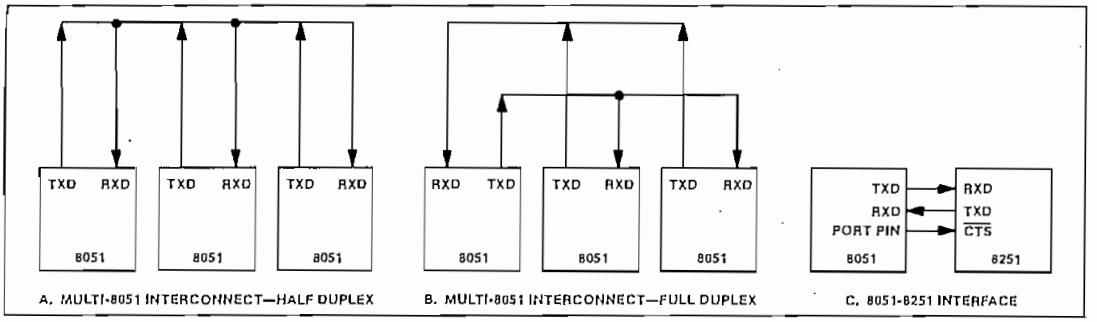


Figure 13. UART Interfacing Schemes

shown in Figure 13 and a method for I/O expansion is shown in Figure 14.

The full-duplex serial I/O port provides asynchronous modes to facilitate communications with standard UART devices, such as printers and CRT terminals, or communications with other 8051s in multi-processor systems. The receiver is double buffered to eliminate the overrun that would occur if the CPU failed to respond to the receiver's interrupt before the beginning of the next frame. Double buffering of the transmitter is not needed since the 8051 can generally maintain the serial link at its maximum rate without it. A minor degradation in transmission rate can occur in rare events such as when the servicing of the transmitter has to wait for a lengthy interrupt service program to complete. In asynchronous modes, false start-bit rejection is provided on received frames. For noise rejection a best two-out-of-three vote is taken on three samples near the center of each received bit.

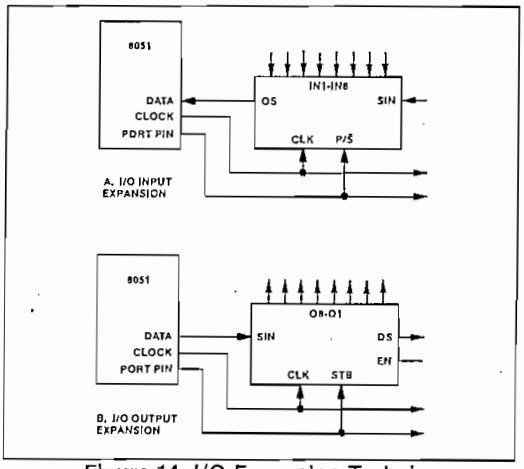


Figure 14. I/O Expansion Technique

When interfacing with standard UART devices the serial channel can be programmed to a mode (Mode 1) that transmits/receives a ten-bit frame or programmed to a mode (Mode 2 or 3) that transmits/receives an eleven-bit frame as shown in Figure 15. The frame consists of a start bit, eight or nine data bits and a stop bit. In Modes 1 and 3, the transmission-rate timing circuitry receives a pulse from counter 1 each time the counter overflows. The input to counter 1 can be an external source or a division by 12 of the oscillator frequency. The auto-reload mode of the counter provides communication rates of 122 to 31,250 bits per second (including start and stop bits) for a 12 MHz crystal. In Mode 2 the communication rate is a division by 64 of the oscillator frequency yielding a transmission rate of 187,500 bits per second (including start and stop bits) for a 12 MHz crystal.

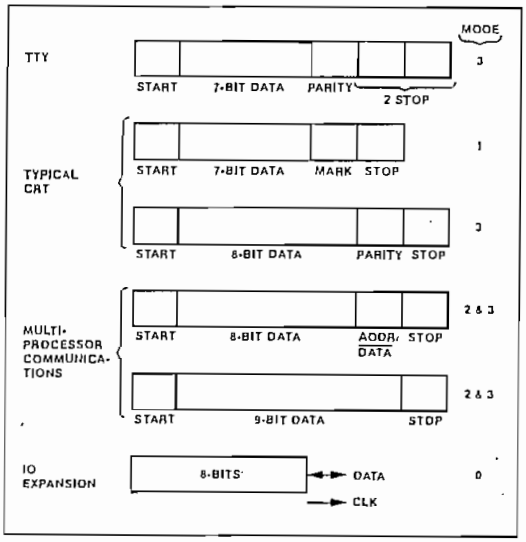


Figure 15. Typical Frame Formats

Distributed processing offers a faster, more powerful system than can be provided by a single CPU processor. This results from a hierarchy of interconnected processors, each with its own memories and

1. Slaves—Configure serial port to interrupt CPU if the received ninth data bit is a one (1).
2. Master—Transmit frame containing address in first 8 data bits and set ninth data bit (i.e. ninth data bit designates address frame).
3. Slaves—Serial port interrupts CPU when address frame is received. Interrupt service program compares received address to its address. The slave which has been addressed reconfigures its serial port to interrupt the CPU on all subsequent transmissions.
4. Master—Transmit control frames and data frames (these will be accepted only by the previously addressed slave).

Figure 16. Protocol for Multi-Processor Communications

/O. In multiprocessing, a host 8051 microcomputer controls a multiplicity of 8051s configured to operate imultaneously on separate portions of the program, ach controlling a portion of the overall process. The interconnected 8051s reduce the load on the ost processor and result in a low-cost system of ata transmission. This form of distributed processng is especially effective in systems where controls n a complex process are required at physically eparated locations.

n Modes 2 and 3 the automatic wake-up of slave rocessors through interrupt driven address-frame ecognition is provided to facilitate interprocessor ommunications. The protocol for interprocessor ommunications is shown in Figure 16. synchronous mode (Mode 0) the high speed serial ort provides an efficient, low-cost method of expandg I/O lines using standard TTL and CMOS shift egisters. The serial channel provides a clock output r synchronizing the shifting of bits to/from an xternal register. The data rate is a division by 12 the oscillator frequency and is 1M bits per second 12 MHz.

051 Family Pin Description

s
rcuit ground potential.

c
V power supply during operation, programming d verification.

rt 0
rt 0 is an 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus when using external memory. It is used for data input and output during programming and verification. Port 0 can sink/source two TTL loads.

PORT 1

Port 1 is an 8-bit quasi-bidirectional I/O port. It is used for the low-order address byte during programming and verification. Port 1 can sink/source one TTL load.

PORT 2

Port 2 is an 8-bit quasi-bidirectional I/O port. It also emits the high-order 8 bits address when accessing external memory. It is used for the high-order address and the control signals during programming and verification. Port 2 can sink/source one TTL load.

PORT 3

Port 3 is an 8-bit quasi-bidirectional I/O port. It also contains the interrupt, timer, serial port and RD and $\overline{WR}$ pins that are used by various options. The output latch corresponding to a secondary function must be programmed to a one (1) for that function to operate. Port 3 can sink/source one TTL load. The secondary functions are assigned to the pins of Port 3, as follows:

- RXD/data (P3.0). Serial port's receiver data input (asynchronous) or data input/output (synchronous).
- TXD/clock (P3.1). Serial port's transmitter data output (asynchronous) or clock output (synchronous).
- $\overline{INT0}$ (P3.2). Interrupt 0 input or gate control input for counter 0.
- $\overline{INT1}$ (P3.3). Interrupt 1 input or gate control input for counter 1.
- T0 (P3.4). Input to counter 0.
- T1 (P3.5). Input to counter 1.
- $\overline{WR}$ (P3.6). The write control signal latches the data byte from Port 0 into the External Data Memory.
- $\overline{RD}$ (P3.7). The read control signal enables External Data Memory to Port 0.

RST/V_{PD}

A low to high transition on this pin (at approximately 3V) resets the 8051. If V_{PD} is held within its spec (approximately +5V), while V_{CC} drops below spec, V_{PD} will provide standby power to the RAM. When V_{PD} is low, the RAM's current is drawn from V_{CC}. A small internal resistor permits power-on reset using only a capacitor connected to V_{CC}.

ALE/$\overline{PROG}$

Provides Address Latch Enable output used for latching the address into external memory during normal operation. Receives the program pulse

input during EPROM programming.

$\overline{\text{PSEN}}$

The Program Store Enable output is a control signal that enables the external Program Memory to the bus during normal fetch operations.

$\overline{\text{EA}}$/VDD

When held at a TTL high level, the 8051 executes instructions from the internal ROM/EPROM when the PC is less than 4096. When held at a TTL low level, the 8051 fetches all instuctions from external Program Memory. The pin also receives the 21V EPROM programming supply voltage.

XTAL1

Input to the oscillator's high gain amplifier. A crystal or external source can be used.

XTAL2

Output from the oscillator's amplifier. Required when a crystal is used.

8051 FAMILY DEVELOPMENT SYSTEM AND SOFTWARE SUPPORT

The 8051 is supported by a total range of Intel development tools. This broad range of support shortens the product development cycle and thus brings the product to market sooner.

- ASM51 Absolute macro assembler for the 8051.
- CONV51 8048 assembly language source code to 8051 assembly source code conversion program.
- EM-51 8051/8751 emulator board that uses a modified 8051 and an EPROM.
- ICE-51™ Real-time in-circuit emulator.
- SDK-51 System Design Kit for Developing User Prototype around the 8051.
- UPP-551 8751 personality card for UPP-103 Universal PROM Programmer.
- 8051 Workshop.

8051 Software Development Package (ASM51 and CONV51)

The 8051 software development package provides development system support for the powerful 8051 family of single chip microcomputers. The package contains a symbolic macro assembler and a 8048 to 8051 source code converter. This diskette-based software package runs under ISIS-II on any Intellec® Microcomputer Development System with 64K bytes of memory.

8051 Macro Assembler (ASM51)

The 8051 macro assembler translates symbolic 8051 assembly language instructions into machine executable object code. These assembly language mnemonics are easier to program and are more readable than binary or hexidecimal machine instructions. Also, by allowing the programmer to give symbolic names to memory locations rather than absolute addresses, software design and debug are performed more quickly and reliably.

ASM51 provides symbolic access for the many useful addressing methods in the 8051 architecture which reference bit, nibble and byte locations.

The assembler supports macro definitions and calls. This provides a convenient means of programming a frequently used code sequence only once. The assembler also provides conditional assembly capabilities. Cross referencing is provided in the symbol table listing, which shows the user the lines in which each symbol was defined and referenced.

If an 8051 program contains errors, the assembler provides a comprehensive set of error diagnostics, which are included in the assembly listing.

The object code generated may be used to program the 8751 EPROM version of the chip or sent to Intel for fabricating the 8051 ROM version. The assembler output can also be debugged using the ICE-51 in-circuit emulator.

8048 to 8051 Assembly Language Converter Utility Program (CONV51)

The 8048 to 8051 assembly language converter is a utility to help users of the MCS-48 family of microcomputers upgrade their designs to the high performance 8051 architecture. By converting 8048 source code to 8051 source code, the investment in software developed for the 8048 is maintained when the system is upgraded.

8051 Emulation Board (EM-51)

The EM-51 8051 emulation board is a small (2.85" x 5.25") board which emulates an 8031/8051/8751 microcomputer using standard PROMs or EPROMs in place of the 8051's on-chip program memory. The board includes a modified 8051 microcomputer, supporting circuits, and two sockets for program memory. The user may select two 2716 EPROMs, a 2732 EPROM, or two 3636 bipolar PROMs depending on crystal frequency and power requirements.

8051 In-Circuit Emulator (ICE-51™)

The 8051 In-Circuit Emulator resides in the Intellec development system. The development system interfaces with the user's 8051 system through an in-cable buffer box with the cable terminating in an 8051 pin-compatible plug. Together these replace the 8051 device in the system. With the emulator plug in place, the designer can exercise the system in real-time while collecting up to 255 instruction

AFN-01462B-10

ycles of real-time data. In addition, he can single ep the system program.

atic RAM memory is available in the ICE-51 buffer ox to emulate the 8051's internal and external pro- am memories and external data memory. The signer can display and alter the contents of the placement memory in the ICE-51 buffer box, lernal 8051 registers, internal data RAM, and ecial Function Registers. Symbolic reference pability allows the designer to use meaningful mbols provided by ASM51 rather than absolute lues when examining and modifying these mem- y, register, flag, and I/O locations in his system.

niversal PROM Programmer ersonality Card (UPP-551)

ne UPP-551 is a personality card for the UPP-103 niversal PROM Programmer. The Universal PROM Programmer is an Intellec system peripheral capable of programming and verifying the 8751 when the UPP-551 is inserted. Programming and verfication operations are initiated from the Intellec development system console and are controlled by the Universal PROM Mapper (UPM) program.

8051 Workshop

The workshop provides the design engineer or system designer hands-on experience with the 8051 microcomputers. The course includes explanation of the Intel 8051 architecture, system timing and input/output design. Lab sessions will allow the attendee to gain detailed familiarity with the 8051 family and support tools.

INSITE™ Library

The INSITE Library contains 8051 utilities and applications programs.

BSOLUTE MAXIMUM RATINGS*

nbient Temperature Under Bias ..... 0° C to 70° C
orage Temperature ..... -65° C to +150° C
ltage on Any Pin With
spect to Ground (V_{SS}) ..... -0.5V to +7V
wer Dissipation ..... 2 Watts

NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

.C. CHARACTERISTICS (T_A = 0° C to 70° C; V_{CC} = 5V ± 5%; V_{SS} = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
IL	Input Low Voltage (All except XTAL1)	-0.5		0.8	V	
IL1	Input Low Voltage (XTAL1)	-0.5		TBD	V	
IH	Input High Voltage (All Except XTAL1, RST/V_{PD})	2.0		V_{CC}+0.5	V	
IH1	Input High Voltage (XTAL1)	TBD		V_{CC}+0.5	V	
IH2	Input High Voltage (RST)	3.0		V_{CC} + 0.5	V	
IH3	Input High Voltage (V_{PD})	4.5		5.5	V	Power Down Only (V_{CC} = 0)
OL	Output Low Voltage (All Outputs Except Port 0)			0.45	V	I_{OL}=2 mA
OL1	Output Low Voltage (Port 0)			0.45	V	I_{OL}=4 mA
OH	Output High Voltage (All Outputs Except Port 0, ALE and $\overline{PSEN}$)	2.4			V	I_{OH}=-100 μA
OH1	Output High Voltage (ALE and $\overline{PSEN}$, Port 0 in External Bus Mode)	2.4			V	I_{OH}=-400 μA
O	Pullup Resistor Current (P1, P2, P3)			500	μA	.45V ≤ V_{IN} ≤ V_{CC}
O1	Output Leakage Current (P0)			±10	μA	.45V ≤ V_{IN} ≤ V_{CC}
C	Power Supply Current (All Outputs Disconnected)			150	mA	T_A=25° C
D	Power Down Supply Current			20	mA	T_A=25° C, V_{PD}=5V, V_{CC}=0V
IO	Capacitance Of I/O Buffer			10	pF	fc=1MHz

AFN-01462B-11

A.C. CHARACTERISTICS (TA = 0°C to 70°C; V_{CC} = 5V ± 5%; CL for Port 0, ALE and $\overline{PSEN}$ Outputs = 150 PF; CL for All Other Outputs = 80 PF)

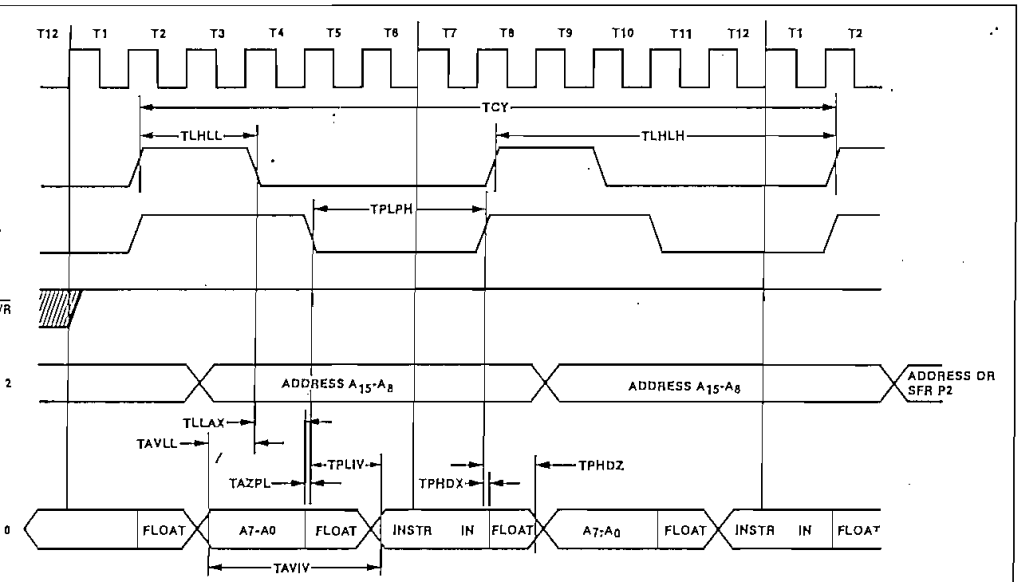
Program Memory Characteristics

Symbol	Parameter	12MHz Clock			Variable Clock 1/TCLCL=1.2 MHz to 12 MHz		
		Min.	Max.	Units	Min.	Max.	Units
TCLCL	Oscillator Period	83		ns			ns
TCY	Min Instruction Cycle Time	1.0		μs	12TCLCL	12TCLCL	ns
TLHLL	ALE Pulse Width	140		ns	2TCLCL-30		ns
TAVLL	Address Set Up To ALE	60		ns	TCLCL-25		ns
TLLAX	Address Hold After ALE	50		ns	TCLCL-35		ns
TPLPH	$\overline{PSEN}$ Width	230		ns	3TCLCL-20		ns
TLHLH	$\overline{PSEN}$, ALE Cycle Time	500		ns	6TCLCL		ns
TPLIV	$\overline{PSEN}$ To Valid Instr In		150	ns		3TCLCL-100	ns
TPHDX	Input Data Hold After $\overline{PSEN}$	0		ns	0		ns
TPHDZ	Input Data Float After $\overline{PSEN}$		75	ns		TCLCL-10	ns
TAVIV	Address To Valid Instr In		320	ns		5TCLCL-100	ns
TAZPL	Address Float To $\overline{PSEN}$	0		ns	0		ns

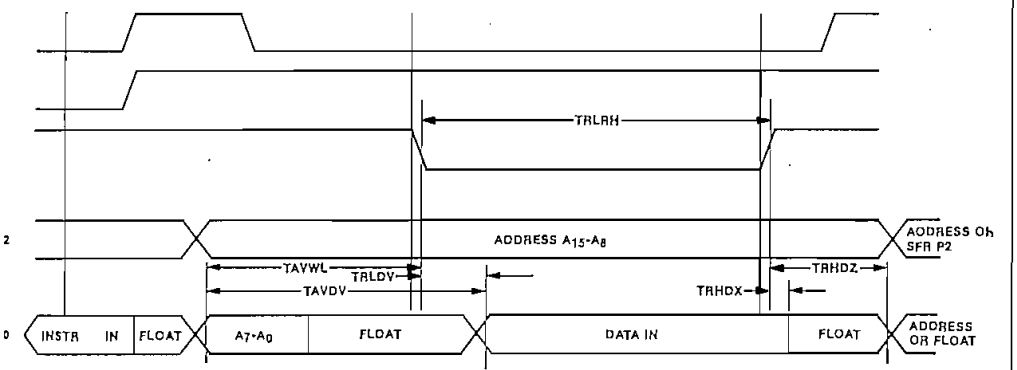
External Data Memory Characteristics

Symbol	Parameter	12MHz Clock			Variable Clock		
		Min.	Max.	Units	Min.	Max.	Units
TRLRH	$\overline{RD}$ Pulse Width	400		ns	6TCLCL-100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		ns	6TCLCL-100		ns
TRLDV	$\overline{RD}$ To Valid Data In		250	ns		5TCLCL-170	ns
TRHDX	Data Hold After $\overline{RD}$	0		ns	0		ns
TRHDZ	Data Float After $\overline{RD}$		100	ns		2TCLCL-70	ns
TAVDV	Address To Valid Data In		600	ns		9TCLCL-150	ns
TAVWL	Address To $\overline{WR}$ or $\overline{RD}$	200		ns	4TCLCL-130		ns
TQVWH	Data Setup Before $\overline{WR}$	400		ns	7TCLCL-180		ns
TWHQX	Data Held After $\overline{WR}$	80		ns	2TCLCL-90		ns

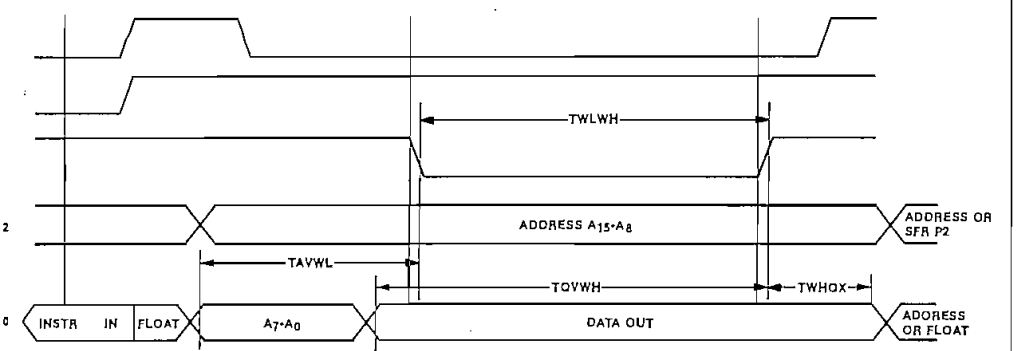
NOTE:
There are 2 to 8 ALE cycles per instruction. Clocks and state timing are shown on the timing diagram for reference purposes only. They are not accessible outside the package. TCY is the minimum instruction cycle time which consists of 12 oscillator clocks or two ALE cycles. Address setup and hold time from ALE are the same for data and program memory.



Program Memory Read Cycle



Data Memory Read Cycle



Data Memory Write Cycle

Table 1. 8051 Instruction Set Summary

Notes on instruction set and addressing modes:

- **Rn** —Register R7-R0 of the currently selected Register Bank.
- **data** —8-bit internal data location's address. This could be an Internal Data Ram location (0-127) or a SFR (i.e. I/O port, control register, status register, etc. (128-255).
- **@Ri** —8-bit internal Data RAM location (0-255) addressed indirectly through register R1 or R0.
- **#data** —8-bit constant included in instruction.
- **#data16** —16-bit constant included in instruction.
- **addr16** —16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.
- **addr11** —11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.
- **rel** —Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
- **bit** —Direct Addressed bit in Internal Data RAM or Special Function Register.
- * —New operation not provided by 8048/8049.

Interrupt Response Time: To finish execution of current instruction, respond to the interrupt request, push the PC and to vector to the first instruction of the interrupt service program requires 38 to 81 oscillator periods (3 to 7 μs @12MHz).

INSTRUCTIONS THAT AFFECT FLAG SETTINGS[1]

INSTRUCTION	FLAG C	OV	AC	INSTRUCTION	FLAG C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C, bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C, bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C, bit	X		
RLC	X			CJNE	X		
SETB C	1						

[1]Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e. the PSW or bits in the PSW) will also affect flag settings.

Data Transfer

Mnemonic	Description	Bytes	Oscillator Periods
MOV A,Rn	Move register to A	1	12
*MOV A,data	Move direct byte to A	2	12
MOV A,@Ri	Move indirect RAM to A	1	12
MOV A,#data	Move immediate data to A	2	12
MOV Rn,A	Move A to register	1	12
*MOV Rn,data	Move direct byte to register	2	24
MOV Rn,#data	Move immediate data to register	2	12
*MOV data,A	Move A to direct byte	2	12
*MOV data,Rn	Move register to direct byte	2	24
*MOV data,data	Move direct byte to direct byte	3	24
*MOV data,@Ri	Move indirect RAM to direct byte	2	24
*MOV data,#data	Move immediate data to direct byte	3	24
MOV @Ri,A	Move A to indirect RAM	1	12
*MOV @Ri,data	Move direct byte to indirect RAM	2	24
MOV @Ri,#data	Move immediate data to indirect RAM	2	12
*MOV DPTR, #data16	Move 16-bit constant to Data Pointer	3	24
*MOV C,bit	Move direct bit to carry	2	12
*MOV bit,C	Move carry to direct bit	2	24
*MOVC A,@A+ DPTR	Move Program Memory byte addressed by A+DPTR to A	1	24
*MOVC A,@A+PC	Move Program Memory byte addressed by A+PC to A	1	24
MOVX A,@Ri	Move External Data (8-bit address) to A	1	24
*MOVX A,@DPTR	Move External Data (16-bit address) to A	1	24
MOVX @Ri,A	Move A to External Data (8-bit address)	1	24
*MOVX @DPTR,A	Move A to External Data (16-bit address)	1	24
*PUSH data	Move direct byte to stack and inc. SP	2	24
*POP data	Move direct byte from stack and dec. SP	2	24
XCH A,Rn	Exchange register with A	1	12
*XCH A,data	Exchange direct byte with A	2	12
XCH A,@Ri	Exchange indirect RAM with A	1	12
XCHD A,@Ri	Exchange indirect RAM's least sig nibble with A's LSN	1	12

Logic

Mnemonic	Description	Bytes	Oscillator Periods
ANL A,Rn	AND register to A	1	12
*ANL A,data	AND direct byte to A	2	12
ANL A,@Ri	AND indirect RAM to A	1	12
ANL A,#data	AND immediate data to A	2	12
*ANL data,A	AND A to direct byte	2	12
*ANL data,#data	AND immediate data to direct byte	3	24
*ANL C,bit	AND direct bit to carry	2	24
*ANL C,/bit	AND complement of direct bit to carry	2	24
ORL A,Rn	OR register to A	1	12
*ORL A,data	OR direct byte to A	2	12
ORL A,@Ri	OR indirect RAM to A	1	12
ORL A,#data	OR immediate data to A	2	12
*ORL data,A	OR A to direct byte	2	12
*ORL data,#data	OR immediate data to direct byte	3	24
*ORL C,bit	OR direct bit to carry	2	24
*ORL C,/bit	OR complement of direct bit to carry	2	24
XRL A,Rn	Exclusive-OR register to A	1	12
*XRL A,data	Exclusive-OR direct byte to A	2	12
XRL A,@Ri	Exclusive-OR indirect RAM to A	1	12
XRL A,#data	Exclusive-OR immediate data to A	2	12
*XRL data,A	Exclusive-OR A to direct byte	2	12
*XRL data,#data	Exclusive-OR immediate data to direct byte	3	24
*SETB C	Set carry	1	12
*SETB bit	Set direct bit	2	12
CLR A	Clear A	1	12
CLR C	Clear carry	1	12
*CLR bit	Clear direct bit	2	12
CPL A	Complement A	1	12
CPL C	Complement carry	1	12
*CPL bit	Complement direct bit	2	12
RL A	Rotate A Left	1	12
RLC A	Rotate A Left through carry	1	12
RR A	Rotate A Right	1	12
RRC A	Rotate A Right through carry	1	12
SWAP A	Rotate A left four (exchange nibbles within A)	1	12

Arithmetic

Mnemonic	Description	Bytes	Oscillator Periods
ADD A,Rn	Add register to A	1	12
*ADD A,data	Add direct byte to A	2	12
ADD A,@Ri	Add indirect RAM to A	1	12
ADD A,#data	Add immediate data to A	2	12
ADDC A,Rn	Add register and carry flag to A	1	12
*ADDC A,data	Add direct byte and carry flag to A	2	12
ADDC A,@Ri	Add indirect RAM and carry flag to A	1	12
ADDC A,#data	Add immediate data and carry flag to A	2	12
*SUBB A,Rn	Subtract register and carry flag from A	1	12
*SUBB A,data	Subtract direct byte and carry flag from A	2	12
*SUBB A,@Ri	Subtract indirect RAM and carry flag from A	1	12
*SUBB A,#data	Subtract immediate data and carry flag from A	2	12
INC A	Increment A	1	12
INC Rn	Increment register	1	12
*INC data	Increment direct byte	2	12
INC @Ri	Increment indirect RAM	1	12
DEC A	Decrement A	1	12
DEC Rn	Decrement register	1	12
*DEC data	Decrement direct byte	2	12
*DEC @Ri	Decrement indirect RAM	1	12
*INC DPTR	Increment Data Pointer	1	24
*MUL AB	*Multiply A* times B	1	48
*DIV AB	Divide A by B	1	48
DA A	Decimal add Adjust of A	1	12

Control Transfer (Branch)

Mnemonic	Description	Bytes	Oscillator Periods
AJMP addr 11	Absolute Jump	2	24
*LJMP addr16	Long Jump	3	24
*SJMP rel	Short Jump	2	24
*JMP @A+DPTR	Jump indirect relative to the DPTR	1	24
JZ rel	Jump if A is zero	2	24
JNZ rel	Jump if A is not zero	2	24
JC rel	Jump if carry is set	2	24
JNC rel	Jump if carry is not set	2	24
*JB bit,rel	Jump relative if direct bit is set	3	24
*JNB bit,rel	Jump relative if direct bit is not set	3	24
*JBC bit,rel	Jump relative if direct bit is set, then clear bit	3	24
*CJNE A,data,rel	Compare direct byte to A & Jump if not Eq. See Note a.	3	24
*CJNE A,#data,rel	Compare immed. to A & Jump if not Eq. See Note a.	3	24
*CJNE Rn,#data, rel	Compare immed. to reg & Jump if not Eq. See Note a.	3	24
*CNJE @Ri, #data,rel	Compare immed. to indirect RAM & Jump if not Eq. See Note a.	3	24
DJNZ Rn,rel	Decrement register & Jump if not zero	2	24
*DJNZ data,rel	Decrement direct byte & Jump if not zero	3	24

Note a) Set C if the first operand is less than the second operand; else clear

Other

Mnemonic	Description	Bytes	Oscillator Periods
NOP	No Operation	1	12

Control Transfer (Subroutine)

Mnemonic	Description	Bytes	Oscillator Periods
ACALL addr11	Absolute Subroutine Call	2	24
LCALL addr16	Long Subroutine Call	3	24
RET	Return from Subroutine Call	1	24
RETI	Return from Interrupt Call	1	24

5.2 APENDICE II: MANUAL DEL USUARIO DEL MODULO DE DESARROLLO PARA SISTEMAS BASADOS EN LOS MICROCONTROLADORES DE LA FAMILIA MCS-51

El módulo de desarrollo para sistemas basados en los microcontroladores de la familia MCS-51/52 consta de:

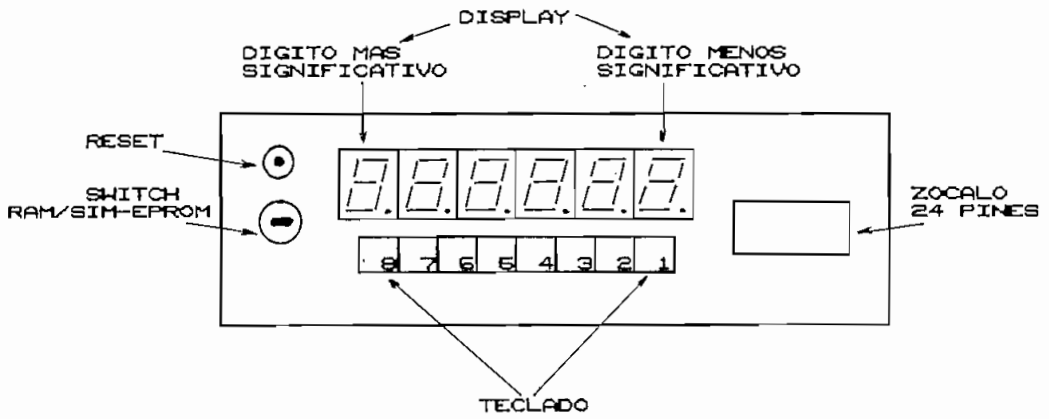
- Módulo de Desarrollo.
- Cable para comunicación serial, terminado en conector DB-9 hembra en un extremo y DB-25 hembra en el otro.
- Cable plano terminado en conectores de 40 pines.
- Baterías de 1.5 V x 4 .

5.2.1 Descripción General.

En la parte frontal del módulo se tiene:

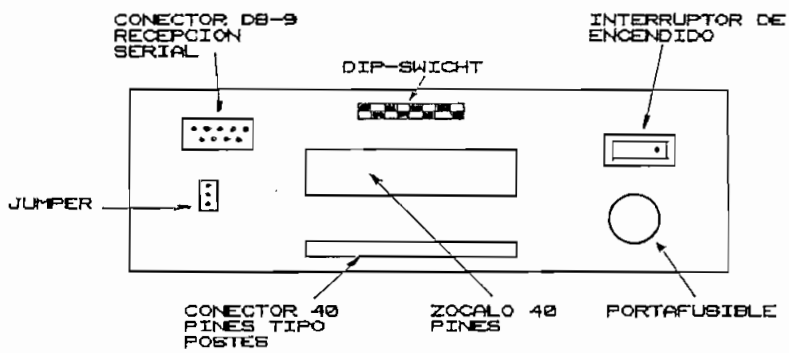
- Pulsador de Reset.
- Switch de palanca, dos posiciones. Mediante este switch se realiza el cambio del modo normal de operación al modo de simulación de EPROM.
- Display de seis dígitos, en donde el dígito derecho corresponde al menos significativo del display, y el del lado izquierdo al mas significativo respectivamente.
- Ocho teclas, en donde la tecla número uno se encuentra en el lado derecho, y la número ocho al lado izquierdo respectivamente.

- Zócalo de 24 pines, donde se coloca la RAM (SIM-EPROM) o la EPROM de acuerdo a la necesidad del usuario.



En la parte posterior del módulo se encuentra:

- Switch de encendido.
- Portafusible, con fusible de 0.5 A 250 V.
- Zócalo de 40 pines, con todas las líneas presentes en el microcontrolador.
- Conector de 40 pines tipo poste.
- Conector DB-9 macho para comunicación serial.
- Jumper dos posiciones, con el cual se selecciona el tipo de memoria a colocarse en el zócalo de 24 pines de la parte frontal.
- Cable de alimentación a la red 115 V.AC.



5.2.2 Operación.

- Modo de operación normal.

Este modo es seleccionable mediante el Switch de palanca de dos posiciones localizado en la parte frontal del módulo.

Al encender el módulo de desarrollo se presenta en el display el mensaje "SIDEL." como señal de que este se encuentra en espera de una tecla de función.

- Tecla 1: Prueba del display, teclado y RAM´s.

Al presionar esta tecla se observa en el display el mensaje "test.", como señal de que se ha ingresado a la función de prueba del módulo; luego consecutivamente se muestra en el display los dígitos hexadecimales desde el "cero" hasta la "efe", para luego mostrar el mensaje "------". Una vez que se tiene en el display este mensaje se debe ir presionando cada una de las teclas una sola vez en cualquier orden. Si el código de ingreso de la tecla es el correcto se presenta el mensaje de "OK.", caso contrario se observa el mensaje de "Error".

Concluida esta parte de la prueba, se observa en el display el mensaje de "RAM", como señal de que se esta probando el buen funcionamiento de la RAM.

Si esta memoria se encuentra en buen estado se muestra "OK." en el display, caso contrario se presenta "Error".

A continuación se prueba la SIM-EPROM, para luego retornar al mensaje inicial ("SIDEL.").

- Tecla 2: Recepción serial.

Al presionar esta tecla se presenta en el display el mensaje "recep.", como señal que se ha ingresado a la función de recepción serial. A continuación y mientras el microcontrolador esta recibiendo datos, el display se encuentra en blanco hasta que este reciba los códigos de finalización de comunicación, emitidos por el programa SIDES-i51 en su opción de programación de EPROM´s.

Todos los datos recibidos en primera instancia son almacenados en la RAM de propósito general, para luego ser procesados por el microcontrolador y almacenados en la SIM-EPROM.

Si todo el proceso se cumple a cabalidad se muestra el mensaje de "OK." en el display, caso contrario el de "Error"; para luego retornar al mensaje inicial.

Para poder ejecutar esta función se necesita tener una RAM en el zócalo frontal (6116 o equivalente), y que jumper posterior se encuentre en la posición adecuada.

La longitud del programa no debe exceder de los 2 Kbytes.

- Tecla 3: Visualización.

Al ingresar a esta función se muestra en el display el mensaje "Ver.", hasta que se seleccione una de las subopciones presentes en esta función.

- Tecla 1: ingreso a la RAM de propósito general en la dirección 8000H.
- Tecla 2: ingreso a la SIM-EPROM en la dirección A000H.
- Tecla 3: ingreso a la RAM interna del

microcontrolador en la dirección 00H.

- Tecla 4: ingreso a los registros de funciones especiales del microcontrolador (SFR´s).
- Tecla 5, tecla 6, tecla 7, tecla 8: retorno al mensaje inicial ("SIDEL.").

Una vez que se escoge la memoria o registros se presenta en el display la dirección seleccionada y el dato contenido en esta localidad, permaneciendo así hasta que se escoja una de las ayudas presentes:

- Tecla 1: Incremento automático de localidad de memoria o registro.
- Tecla 2: Decremento automático de localidad de memoria o registro.
- Tecla 4: Inserción de una localidad de memoria.
- Tecla 5: Borrar una localidad de memoria.
- Tecla 7: Incremento de localidad de memoria o registro, paso a paso.
- Tecla 8: Decremento de una localidad de memoria o registro, paso a paso.
- Tecla 6: Cambio de dirección o dato. Conforme se presiona esta tecla aparece el punto decimal en cada dígito del display, como señal de que se realizará un cambio en el dígito seleccionado.
- Tecla 3: Una vez seleccionado la dirección o dato a cambiarse (tecla 6), mediante esta tecla se incrementa en uno el valor que se encuentre en el dígito seleccionado. De esta manera se puede colocar una dirección o un dato específico en una localidad de memoria o registro.

Para retornar al mensaje inicial se debe presionar dos teclas a la vez.

- **Tecla 4:** Llenar un bloque con un valor prefijado.

Al presionar esta tecla se presenta en el display el mensaje "Fill.", como señal que se ha ingresado a la

función de llenar un bloque. A continuación se presenta el mensaje "P-8000", el cual indica que el principio del bloque a llenarse esta prefijado en la localidad 8000H; mediante la tecla 2, tecla 3, tecla 4, y tecla 5 se puede escoger la localidad inicial del bloque. Al presionar la tecla 1 (ENTER), se acepta la dirección de inicio de bloque y se muestra en el display el mensaje "F-8000", o la dirección que se haya escogido como inicio; así mismo se debe fijar el final de bloque y presionar la tecla 1 para que se presente el mensaje "VAL-FF", en donde mediante la tecla 2, y la tecla 3 se puede escoger el valor con el cual se llenará el bloque prefijado. Al presionar nuevamente la tecla 1 se llena el bloque con el valor escogido y se regresa al mensaje inicial.

Las teclas 6, 7, 8 son teclas de escape que retornan al mensaje inicial, sin realizar proceso alguno.

- **Tecla 5:** Puntos de parada.

Al presionar esta tecla se presenta el mensaje "Parada" como señal que se ha ingresado a la función de colocar puntos de parada en el programa.

Luego se muestra el mensaje "1-A000", como señal que el primer punto de parada esta prefijado en la localidad A000H, al igual que en la función anterior mediante la tecla 2, tecla 3, tecla 4 y tecla 5 se puede escoger la localidad en la cual se desea poner el primer punto de parada; mediante la tecla 8 se acepta este valor y se presenta en el display el mensaje "2-A002", o la dirección que se haya escogido en el primer punto de parada

incrementado en dos, ya que se necesitan dos códigos de máquina para colocar un punto de parada en el programa.
Si el usuario desea colocar un nuevo punto de parada, se puede escoger la dirección y presionar la tecla 8 (ENTER).
Una vez que se ha colocado el o los puntos de parada, para salir de esta función se debe presionar la tecla 1 (ESCAPE).
Para remover del programa el o los puntos de parada colocados se debe ingresar a esta función y presionar la tecla 7, se presenta el mensaje "Remov." y luego regresa al mensaje inicial. Si no existen puntos de parada colocados en el programa esta tecla no realizará ninguna función.
Se pueden colocar un número infinito de puntos de parada, pero únicamente los dos últimos puntos colocados se podrán remover automáticamente.

- Modo de Simulación de Eprom.

Este modo es seleccionable mediante el switch de palanca de dos posiciones en la parte frontal del módulo.
El modo de Simulación de EPROM consta de dos opciones a seleccionar desde el jumper en la parte posterior del módulo.

- Ejecución de un programa desde EPROM 2732.

Si el jumper se encuentra en la posición uno, se ejecuta

el programa desde EPROM 2732 previamente grabada y colocada en el zócalo de 24 pines de la parte frontal del módulo.

- Ejecución de un programa desde RAM 6116 (SIM-EPROM).

Cuando el jumper se encuentra en la posición dos se ejecuta el programa desde la RAM 6116 (o equivalente) colocada en el zócalo de 24 pines de la parte frontal del módulo.

Para esto primero se debe pasar al modo de operación normal del módulo de desarrollo, ingresar a la función de recepción serial (tecla 2), y mediante la opción de grabación de EPROM´s del programa SIDES-i51 trasmitir los códigos de máquina del programa a ejecutarse, una vez que en el display se presente el mensaje "OK." se puede pasar al modo de simulación de EPROM para ejecutar el programa previamente trasmitido.

Al realizar el paso de un modo al otro se recomienda presionar la tecla de Reset antes de cambiar la posición del switch de selección del modo de operación del módulo, para asegurar que el programa comience su ejecución desde la localidad 0000H.

5.2.3 Subrutinas del Usuario.

El usuario del módulo de desarrollo tiene acceso a varias subrutinas de propósito general contempladas en el programa del módulo.

Estas pueden ejecutarse mediante la instrucción LCALL a la dirección de la subrutina que se necesite.

- Subrutina Displays: tiempo de barrido 0.8 seg. localidad 1000H.
- Subrutina Displays1: tiempo de barrido 0.5 seg. localidad 1100H.
- Subrutina Displays2: tiempo de barrido 0.1 seg. localidad 1200H.
- Subrutina Coddis: localidad 1300H.
- Subrutina Decrem: localidad 1400H.
- Subrutina Delay: localidad 1500H.
- Subrutina Message: localidad 1600H.
- Subrutina Writ: localidad 1700H.
- Subrutina Err: localidad 1750H.

El usuario puede llamar a estas subrutinas desde cualquier banco de registros de la RAM interna sin peligro de perder datos o valores que se encuentren en los registros.

5.2.4 Restricciones.

El programa del módulo de desarrollo utiliza las localidades de RAM interna desde la 7AH hasta la 7FH, para salida de datos hacia displays, por lo tanto se recomienda

no utilizar estas localidades de memoria de la RAM interna.

Para banderas se salto y ejecución de funciones se utilizan los bits direccionables directamente; los bits ocupados por el programa están a partir del 70H hasta el 7FH. Por lo tanto al igual que en el caso anterior se recomienda no utilizar estos bits.

El programa del módulo de desarrollo hace uso extensivo de la pila de almacenamiento de datos, por lo cual se recomienda inicializar el puntero del STACK en la dirección 4FH.

5.3 ANEXO I: TARJETA MCPD51DA.

CARACTERISTICAS TECNICAS
Y GUIA DE USO

TARJETA MCPD51DA

PARA DESARROLLO DE PROYECTOS CON
MICROCONTROLADORES INTEL MCS-51

Preparado por:

Ing. Bolívar Ledema G.

Quito, Septiembre de 1992

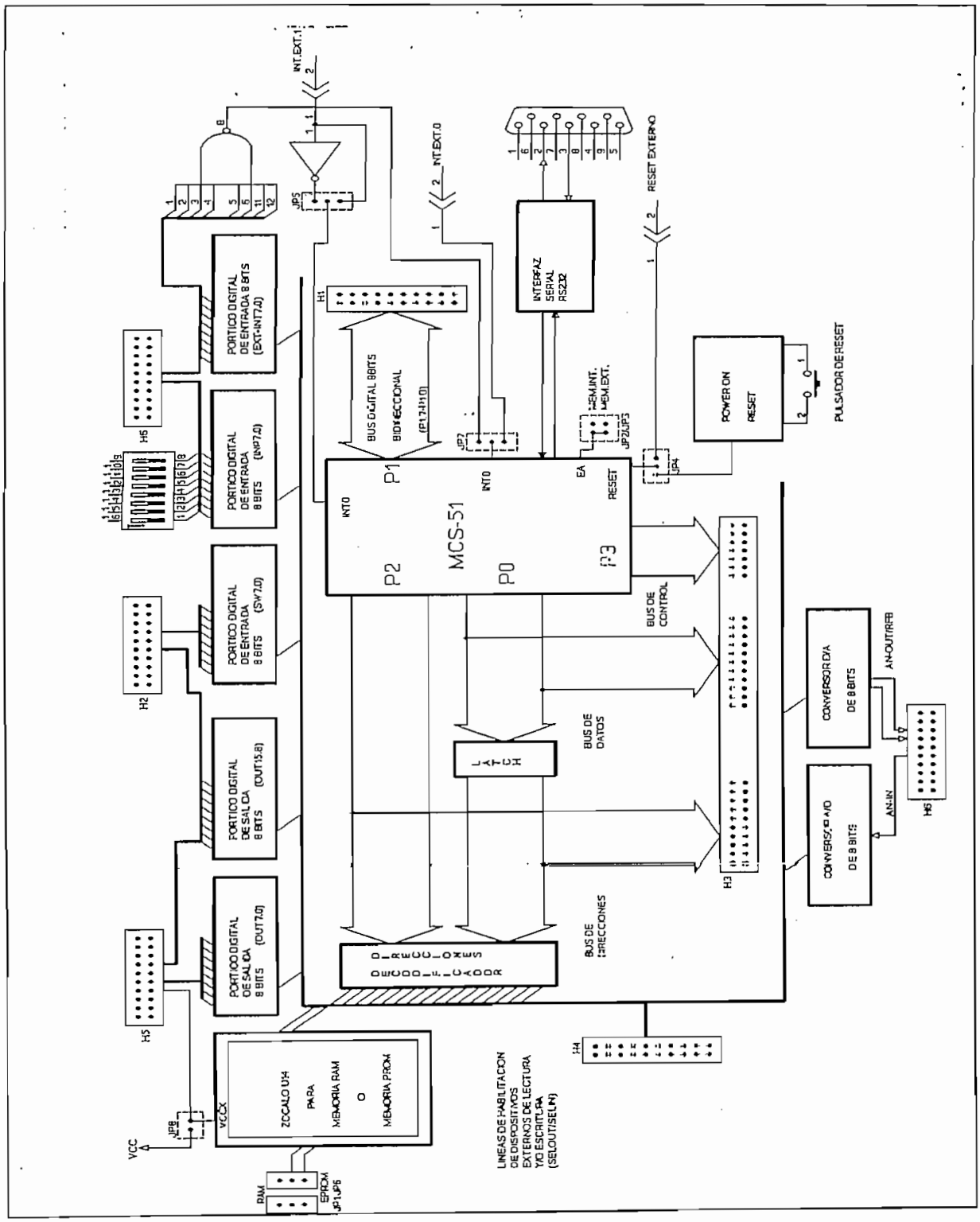


Fig. 1 DIAGRAMA DE BLOQUES TARJETA MCPD51DA

La figura 1 muestra el diagrama de bloques simplificado de la tarjeta MCPD51DA, la misma que ha sido diseñada para permitir el desarrollo de proyectos y aplicaciones de caracter general, en base a los microcontroladores INTEL de la familia MCS-51.

La tarjeta pone a disposición del usuario los siguientes recursos:

- 1 pórtico digital bidireccional de 8 bits (P1.7 a P1.0).
- 2 pórticos digitales de salida de 8 bits (OUT0 a OUT15).
- 1 pórtico digitales de entrada de 8 bits (SW0 a SW7).
- 1 pórtico digital de entrada de 8 bits con opción de colocar un dipswitch-8 en la tarjeta (INP0 a INP7).
- 1 pórtico digital de entrada de 8 bits con opción de generar interrupción externa (EXT-INT0 a EXT-INT7).
- *1 entrada* analógica (0 a 5 V) y conversor A/D de 8 bits.
- 1 salida analógica (0 a 5 V) proveniente de un conversor D/A de 8 bits.
- 1 pórtico de comunicación serial RS232 (conector DB9).
- 1 Bus de datos del microcontrolador (D7 a D0).
- 1 Bus de direcciones del microcontrolador (A15 a A0).
- 1 Bus de señales de control del microcontrolador (RD, WR, T0, T1, ALE, PSEN, RESET, INT0, INT1).
- 1 Bus de señales de control para habilitación de dispositivos externos: 3 de entrada: SELIN5,6,7 y 4 de salida: SELOUT3,5,6,7.
- Memoria RAM de 2 Kbytes.
- Circuito de reset interno con pulsador.

CONFIGURACIONES:

La *tarjeta* MCPD51DA incluye toda la circuitería básica asociada a un microcontrolador MCS-51, permitiendo al usuario configurar la tarjeta para sus aplicaciones específicas. Para el efecto la tarjeta tiene 8 "jumpers" de configuración (JP1 a JP8) que permiten seleccionar las alternativas de la tabla No. 1.

La nomenclatura utilizada para los jumpers de dos puntos hace referencia a la colocación física del "jumper" cuando se tiene "ON" y su ausencia física cuando se tiene "OFF". Los jumpers de 3 puntos funcionan como switches de dos posiciones, donde el punto central es el común. Por ejemplo, JP6 es un jumper de 3 puntos donde los extremos se denominan "RAM" y "EPROM" respectivamente. Cuando la tabla No. 1 dice que JP6 = "RAM" ésto significa que se debe colocar un jumper entre el punto central de JP6 y el extremo "RAM".

JUMPERS	POSICION	FUNCION
JP1 y JP6	RAM	El microcontrolador trata a U14 como RAM (memoria de datos).
JP1 y JP6	EPROM	El microcontrolador trata a U14 como ROM (memoria de programa).
JP2 y JP3	JP2 = "ON" JP3 = "OFF"	El microcontrolador ejecuta el programa que reside en memoria externa.
JP2 y JP3	JP2 = "OFF" JP3 = "ON"	El microcontrolador ejecuta el programa que reside en su memoria ROM interna.
JP4	INT-RESET	El RESET del microcontrolador queda conectado al circuito "power on reset" y al pulsador internos.
JP4	EXT-RESET	El RESET del microcontrolador queda conectado a una entrada exterior del conector H3 (H3.3 EXRST).
JP5	0L-EX1	Interrupción externa 1 del microcontrolador activa con 0L, accesible desde conector H3.38 y H1.13.
JP5	1L-EX1	Interrupción externa 1 del microcontrolador activa con 1L, accesible desde conector H3.38 y H1.13.
JP7	0..7-EX0	Interrupción externa 0 del microcontrolador activa con 0L y accesible desde cualquier línea del pórtico EXT-INT0..7, conector H6.
JP7	EXT-EX0	Interrupción externa 0 del microcontrolador activa con 0L y accesible desde conector H3.39.
JP8	"ON"	U14 se polariza con la fuente VCC de la tarjeta.
JP8	"OFF"	U14 recibe polarización externa VCCX desde el conector H5.20.

Tabla No. 1

Alrededor de la arquitectura de la tarjeta MCPD51DA se puede desarrollar proyectos de aplicación específica, utilizando componentes de la familia MCS-51 con memoria interna de programa, como el 8051 ó el 8751, en cuyo caso el zócalo U14 queda disponible para un chip de RAM de 2 Kbytes. Si se utiliza el microcontrolador 8031 que carece de memoria interna de programa, el zócalo U14 necesariamente deberá alojar una memoria de programa como la 2716 o 2732.

La tabla No. 2 muestra la manera de configurar la tarjeta en función del tipo de microcontrolador y de la disposición de la memoria de programa.

INTEL MCS-	JUMPERS	U14	DESCRIPCION
8051 ó 8751	JP3 = "ON" JP2 = "OFF" JP1 = "RAM" JP6 = "RAM"	RAM 2 Kb (6116)	El micro ejecuta el programa residente en su ROM interna. U14 puede alojar una memoria RAM o no ser utilizado.
8051 ó 8751	JP3 = "OFF" JP2 = "ON" JP1 = "EPROM" JP6 = "EPROM"	EPROM 4 Kb (2732)	El micro ejecuta el programa residente en la EPROM externa colocada en U14.
8051 ó 8751	JP3 = "OFF" JP2 = "ON" JP1 = "EPROM" JP6 conectado a "INT" de JP3.	RAM 2Kb (6116)	El micro ejecuta el programa residente en la RAM externa colocada en U14.
8031	JP3 = "OFF" JP2 = "ON" JP1 = "EPROM" JP6 = "EPROM"	EPROM 4 Kb (2732)	El micro ejecuta el programa residente en la EPROM externa colocada en U14.

Tabla No. 2

Por otro lado, la tarjeta puede utilizarse como un sistema de desarrollo y depuración de programas de aplicación que se pueden descargar a través del puerto serial, desde un computador personal hacia la memoria RAM de la tarjeta. Luego, se reconfigura la tarjeta para que ejecute el programa residente en RAM. De esta manera se agilita la realización de pruebas de operación sin necesidad de borrar y reprogramar EPROMS.

MAPA DE MEMORIA:

Los microcontroladores Intel de la familia MCS-51 tienen posibilidad de direccionar 64 K localidades externas a través del bus de direcciones de 16 bits. En la tarjeta MCPD51DA se ha incluído toda la circuitería que se requiere para direccionar independientemente a 8 dispositivos de entrada y 8 dispositivos de salida, dividiendo los 64K en páginas de 8K. Las líneas decodificadas para habilitación de dispositivos de entrada (lectura) se denominan SELIN0 a SELIN7. Las líneas para habilitación de dispositivos de salida (escritura), se denominan SELOUT0 a SELOUT7. Para la decodificación del bus de direcciones y la correspondiente división en páginas de 8K, se han utilizado los tres bits más

significativos A15, A14 y A13 los cuales, en combinación con las señales READ y WRITE del micro, determinan la activación de la correspondiente señal de habilitación.

Por ejemplo, si A15, A14 y A13 tienen el valor 0L, cuando el micro ejecuta una instrucción de escritura en memoria externa (MOVX @DPTR, A), la línea que se habilitará será SELOUT0. Si el micro ejecuta una instrucción de lectura de memoria externa (MOVX A,@DPTR), la línea que se habilitará será SELIN0.

De las 8 líneas de habilitación de entrada y 8 de salida existentes, varias están utilizadas para los pórticos de entrada, salida, conversor A/D y otros recursos disponibles en la tarjeta. Las restantes están accesibles para el usuario a través del conector H4.

La tabla No. 3 muestra la distribución de memoria y las líneas disponibles para el usuario.

DIRECCIONES	HABILITACION	DISPOSITIVO DE ENTRADA (LECTURA)	DISPOSITIVO DE SALIDA (ESCRITURA)
0000H-1FFFH	SELIN0/SELOUT0	Pórt.SW0-SW7	Pórt.OUT0-OUT7
2000H-3FFFH	SELIN1/SELOUT1	Pórt.EXT-INT0 a EXT-INT7	Pórt.OUT8-OUT15
4000H-5FFFH	SELIN2/SELOUT2	Pórt.INP0-INP7	Conv. D/A.
6000H-7FFFH	SELIN3/SELOUT3	Conv. A/D.	DISPONIBLE
8000H-8FFFH	SELIN4/SELOUT4	MEMORIA RAM	MEMORIA RAM
A000H-BFFFH	SELIN5/SELOUT5	DISPONIBLE	DISPONIBLE
C000H-DFFFH	SELIN6/SELOUT6	DISPONIBLE	DISPONIBLE
E000H-FFFFH	SELIN7/SELOUT7	DISPONIBLE	DISPONIBLE

Tabla No. 3

DISTRIBUCION DE COMPONENTES DE LA TARJETA:

En la figura 2 se puede apreciar la distribución de los componentes en la tarjeta.

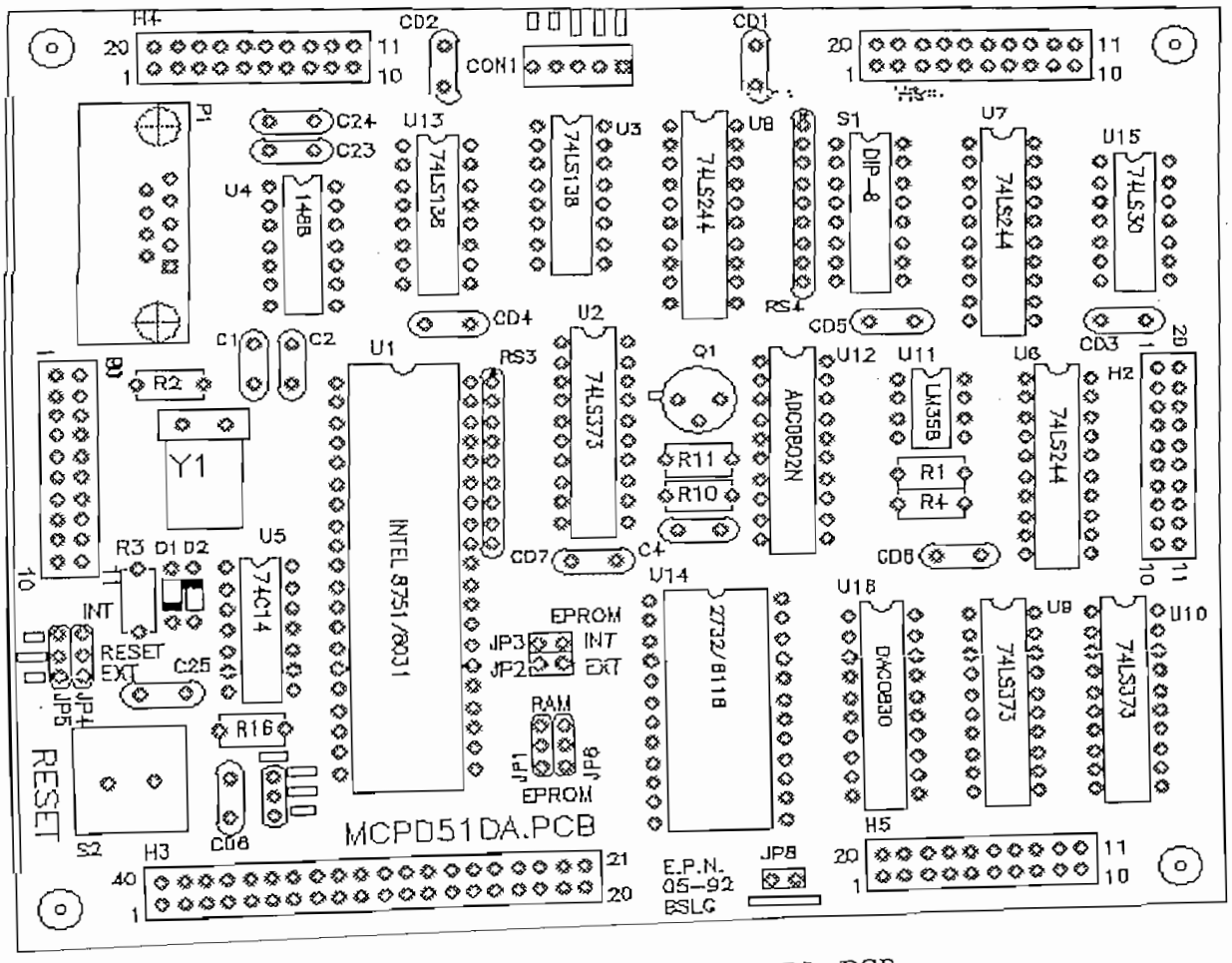


Fig. 2 TARJETA MCPD51DA.PCB

DISTRIBUCION DE SEÑALES EN LOS CONECTORES:

La tarjeta tiene seis conectores (H1 a H6) a través de los cuales entrega y recibe señales y datos. Prácticamente en todos los conectores se tiene acceso a las fuentes principales de polarización del circuito lógico a través de los pines VCC y VSS.

Existe un conector especial denominado "CON1" que sirve para conectar la tarjeta con la fuente de polarización principal, donde se tiene que:

VCC = + 5 V;
VSS = 0 V;
V+ = + 12 V;
V- = - 12 V.

CONECTOR H1:

P17..P10:	Pórtico P1 del microcontrolador (8 bits bidireccional).
EXINT0:	Acceso a la interrupción externa 0 del micro.
EXINT1:	Acceso a la interrupción externa 1 del micro.
T0:	Entrada al timer/counter 0 del micro.
T1:	Entrada al timer/counter 1 del micro.

CONECTOR H2:

SW7..SW0:	Pórtico digital de entrada de 8 bits.
OUT15..OUT8:	Pórtico digital de salida de 8 bits.

CONECTOR H3:

El conector H3 permite el acceso directo hacia el microcontrolador: el bus de datos, el bus de direcciones y todas las señales de control: ALE, READ, WRITE, PSEN.

D7..D0:	Bus de datos del micro.
A15..A0:	Bus de direcciones del micro.
T0:	Entrada al timer/counter 0 del micro.
T1:	Entrada al timer/counter 1 del micro.
EXRST:	Entrada para ingreso de señal de reset externo.
EXINT0:	Acceso a la interrupción externa 0 del micro.
EXINT1:	Acceso a la interrupción externa 1 del micro.

CONECTOR H4:

SELIN5,6,7:	Señales de habilitación para dispositivos externos de entrada (lectura).
SELOUT3,5,6,7:	Señales de habilitación para dispositivos externos de salida (escritura).
RXD:	Línea de recepción para comunicación serial del microcontrolador (niveles TTL).
TXD:	Línea de transmisión para comunicación serial del microcontrolador (niveles TTL).
RXIN:	Línea de recepción serial RS232 (conector DB9) de la tarjeta (niveles de voltaje ±12V).
TXOUT:	Línea de transmisión serial RS232 (conector DB9) de la tarjeta (niveles de voltaje ±12V).
T0PWM:	Señal de salida correspondiente al complemento lógico del pin T0 del microcontrolador y en niveles lógicos ±12 V.
T1PWM:	Señal de salida correspondiente al complemento lógico del pin T1 del microcontrolador y en niveles lógicos ±12 V.

CONECTOR H5:

OUT7..OUT0:	Pórtico digital de salida de 8 bits.
OUT15..OUT8:	Pórtico digital de salida de 8 bits.
VCCX:	Entrada externa para polarización de U14.

CONECTOR H6:

EXT-INT7..EXT-INT0:	Pórtico digital de entrada de 8 bits con opción a generar interrupción externa 0 en el microcontrolador.
INP7..INP0:	Pórtico digital de entrada de 8 bits con opción de colocar un dip-switch 8 en la tarjeta.
AN-OUT:	Salida analógica (0 a +5V) proveniente del conversor D/A (DAC0830).
RFB:	Salida analógica (0 a -5V) correspondiente a la señal invertida que viene del conversor D/A.
AN-IN:	Entrada analógica (0 a +5V) hacia el conversor A/D (ADC0804).

ESQUEMA DE CONECTORES:

H1

VCC	1	20	VCC
VCC	2	19	VCC
P10	3	18	P11
P12	4	17	P13
P14	5	16	P15
P16	6	15	P17
T0	7	14	T1
EXINT0	8	13	EXINT1
VSS	9	12	VSS
VSS10		11	VSS

H2

SW1	1	20	SW0
SW7	2	19	SW2
SW6	3	18	SW3
SW5	4	17	SW4
VCC	5	16	VCC
VSS	6	15	VSS
OUT8	7	14	OUT15
OUT14	8	13	OUT13
OUT12	9	12	OUT10
OUT11	10	11	OUT9

H4

TXOUT	1	20	RXIN
TXD	2	19	RXD
V+	3	18	V-
TOPWM	4	17	T1PWM
VSS	5	16	VSS
SELIN7	6	15	SELIN6
SELIN5	7	14	VCC
SELOUT7	8	13	SELOUT6
SELOUT5	9	12	SELOUT3
VCC	0	11	VCC

H3

VCC	1	40	VCC
VCC	2	39	EXINT0
EXRST	3	38	EXINT1
T0	4	37	T1
WRITE	5	36	READ
D0	6	35	D1
D2	7	34	D3
D4	8	33	D5
D6	9	32	D7
ALE	10	31	PSEN
A15	11	30	A14
A13	12	29	A12
A11	13	28	A10
A9	14	27	A8
A6	15	26	A7
A4	16	25	A5
A2	17	24	A3
A0	18	23	A1
VSS	19	22	VSS
VSS	20	21	VSS

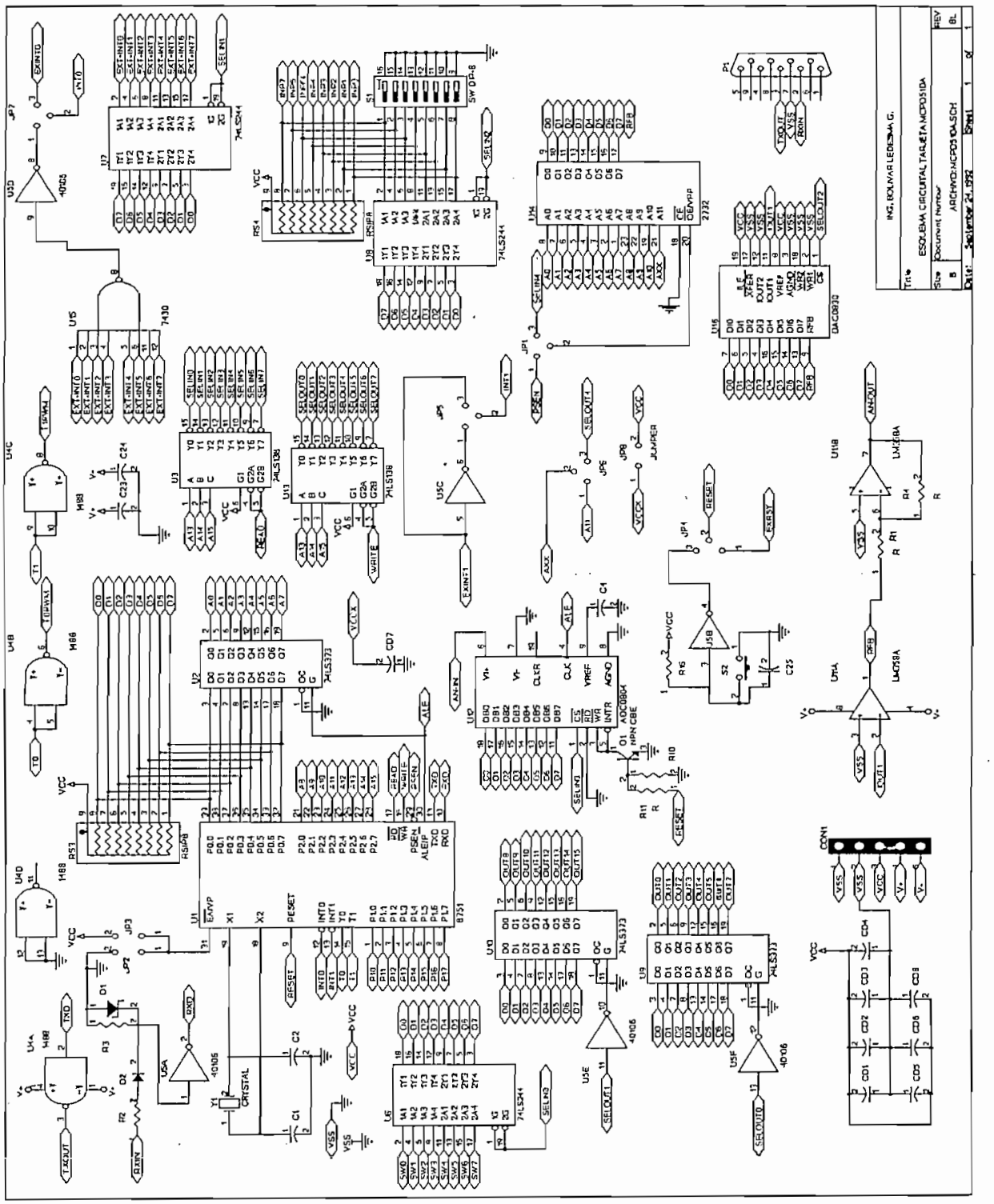
H5

VCCX	1	20	VCC
OUT2	2	19	OUT3
OUT0	3	18	OUT1
OUT6	4	17	OUT7
OUT4	5	16	OUT5
VSS	6	15	VSS
OUT10	7	14	OUT11
OUT8	8	13	OUT9
OUT14	9	12	OUT15
OUT12	10	11	OUT13

H6

INP6	1	20	INP7
INP4	2	19	INP5
INP2	3	18	INP3
INP0	4	17	INP1
RFB	5	16	AN-IN
VSS	6	15	AN-OUT
EXT-INT0	7	14	EXT-INT1
EXT-INT2	8	13	EXT-INT3
EXT-INT4	9	12	EXT-INT5
EXT-INT6	10	11	EXT-INT7

ESQUEMA CIRCUITAL TARJETA MCPD51DA:



5.4 ANEXO II: DISKETTE DEL SISTEMA Y DATOS.

BIBLIOGRAFIA.

[1] GONZALEZ, J., INTRODUCCION A LOS MICROCONTROLADORES HARDWARE, SOFTWARE Y APLICACIONES. McGRAW-HILL 1992.

[2] INTEL, COMPONENT DATA CATALOG 1991.

[3] INTEL, MCS-51 ARCHITECTURAL OVERVIEW 1991.

[4] INTEL, MCS-51 ARCHITECTURE. 1990.

[5] AVOCET SYSTEMS INC, AVMAC 8051 USER´S MANUAL 1986.

[6] TEXAS INSTRUMENTS, TTL DATA BOOK 1985.