ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

PROGRAMA PARA VISUALIZAR LA COMUNICACION DE UN MICROCONTROLADOR CON SUS PERIFERICOS

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

FRANZ MISAEL ESPARZA ROMERO
PATRICIO XAVIER QUINTANILLA PEÑA

MARZO DE 1999

CERTIFICACIÓN

Certifico que el presente trabajo, ha sido desarrollado en su totalidad por el Sr. Franz Esparza y el Sr. Xavier Quintanilla.

Ing. Jaime Velarde
DIRECTOR DE TESIS

DEDICATORIA

A mi FAMILIA que siempre me ha dado su apoyo.

Franz

DEDICATORIA

A mi querida Familia y amigos por el apoyo incondicional en la consecución de mis aspiraciones.

Xavier

AGRADECIMIENTO

Nuestro sincero agradecimiento al Ing. Jaime Velarde por su ayuda incondicional en el desarrollo de esta tesis.

AGRADECIMIENTO

Nuestro sincero agradecimiento a la Escuela Politécnica Nacional y en especial a la Facultad de Ingeniería Eléctrica por haber fortalecido nuestro espíritu.

INDICE

CAPITULO I :

MICROCONTROLADORES DE LA FAMILIA MCS-51

1.1 INTRODUCCION.....1

1.2 OBJETIVOS.....2

1.3 ANTECEDENTES.....3

1.4 DESCRIPCION DE LA FAMILIA MCS-51.....5

CAPITULO II :

EL MICROCONTROLADOR MCS-8031

2.1 ARQUITECTURA Y ORGANIZACION DE LA MEMORIA.....9
2.1.1 ARQUITECTURA.....9
2.1.2 ORGANIZACION DE LA MEMORIA.....11
2.1.2.1 Memoria de Programa.....11
2.1.2.2 Memoria de Datos.....13

2.2 MODOS DE DIRECCIONAMIENTO Y CONJUNTO DE INSTRUCCIONES.....14
2.2.1 MODOS DE DIRECCIONAMIENTO.....14
2.2.1.1 Direccionamiento por Registros.....15
2.2.1.2 Direccionamiento Directo.....16
2.2.1.3 Direccionamiento Inderecto.....16
2.2.1.4 Direccionamiento Inmediato.....16
2.2.1.5 Direccionamiento Indexado.....16
2.2.2 CONJUNTO DE INSTRUCCIONES.....17

2.3 PERIFERICOS INTERNOS Y EXTERNOS DEL MICROCONTROLADOR.....22
2.3.1 PERIFERICOS INTERNOS.....22
2.3.1.1 Oscilador y Circuito de Reloj.....22
2.3.1.2 Temporización de la CPU.....23

2.3.1.3 Estructura y Operación de los Pórticos Paralelos.....26
2.3.1.4 Escritura en los Pórticos Paralelos.....27
2.3.1.5 Lectura desde los Pines de los Pórticos Paralelos.....28
2.3.1.6 Característica de la Lectura-Modificación-Escritura.....29
2.3.1.7 Temporizadores y Contadores.....29
2.3.1.8 Timer 0 y Timer 1.....30
Modo 0.....31
Modo 1.....32
Modo 2.....33
Modo 3.....33
2.3.1.9 Interrupciones.....33
2.3.1.10 Prioridad de Interrupciones.....37
2.3.1.11 Manejo de Interrupciones.....38
2.3.1.12 Pórtico Serial.....41
2.3.1.13 Registro de Control y Estado SCON.....41
2.3.1.14 Velocidad de Comunicación.....43
2.3.1.15 Modos de Operación.....45
Modo 0.....45
Modo 1.....47
Modo 2 y 3.....50
2.3.2 PERIFERICOS EXTERNOS.....53
2.3.2.1 Acceso a Memoria Externa.....53
PSEN.....54
ALE.....56
2.3.2.2 Dispositivos Externos.....56

CAPITULO III :

BASES PARA LA ELABORACION DEL PROGRAMA

3.1 CONSIDERACIONES Y LIMITACIONES.....58

3.2 DIAGRAMAS DE CONFIGURACION.....62

3.3 INTERACCION DEL MICROCONTROLADOR CON PERIFERICOS EXTERNOS.....64
3.3.1 INTERACCION MICROCONTROLADOR-MEMORIA ROM 64
3.3.2 INTERACCION MICROCNTROLADOR-MEMORIA RAM EXTERNA.....66
3.3.3 INTERACCION DEL PUERTO P1 DEL MICROCONTROLADOR CON ELEMENTOS EXTERNOS.....73

CAPITULO IV :

ELABORACION DEL PROGRAMA

4.1 REQUERIMIENTOS DE HARDWARE Y SOFTWARE PARA LA ELABORACION.....77

4.2 DIAGRAMA DE FLUJO.....78

4.3 ESTRUCTURA Y DESARROLLO DEL PROGRAMA DE VISUALIZACION.....80
4.3.1 DESCRIPCION GENERAL.....80
a) Carga del Programa a Visualizar.....82
b) Ejecuación de las Instrucciones del Programa.....84
c) Descripción del Diagrama Básico en Modo de Diseño.....88
d) Control de P1, Timer e Interrupciones.....90
4.3.2 INTERACCION CON EL PORTICO SERIAL.....94

CAPITULO V :

PRUEBAS DEL PROGRAMA

5.1 PROGRAMAS DE PRUEBA.....97
5.1.1 DEFINICION DE LOS PROGRAMAS DE PRUEBA.....97
5.1.1.1 Programa Ejemplo de Acceso a Memoria Externa.....98
5.1.1.2 Programa Ejemplo de Llamado a Subrutinas y Saltos.....98
5.1.1.3 Programa Ejemplo de Pórticos, Temporizadores e Interrupciones.....99
5.1.1.4 Programa Ejemplo del Pórtico Serial 99

5.2 PRUEBAS Y RESULTADOS.....100
5.2.1 PROGRAMA EJEMPLO DE ACCESO A MEMORIA EXTERNA.....100
a) Código y Resultados Esperados.....100
b) Ejecución y Resultados Obtenidos.....102
5.2.2 PROGRAMA EJEMPLO DE LLAMADO A SUBRUTINAS Y SALTOS.....113
a) Código y Resultados Esperados.....113
b) Ejecución y Resultados Obtenidos.....117
5.2.3 PROGRAMA EJEMPLO DE PORTICOS, TEMPORIZADORES E INTERRUPCIONES.....119
a) Código y Resultados Esperados.....119
b) Ejecución y Resultados Obtenidos.....123

5.2.4 PROGRAMA DEMO DEL PORTICO SERIAL.....128
a) Código y Resultados Esperados.....128
b) Ejecución y Resultados Obtenidos.....131

5.3 CONCLUSIONES Y RECOMENDACIONES.....134

BIBLIOGRAFIA.....139

ANEXOS

A. CODIGO FUENTE.....A1

B. MANUAL DE USUARIO.....B1

C. INFORMACION TECNICA.....C1

CAPITULO I

MICROCONTROLADORES DE LA FAMILIA MCS-51

1.1. INTRODUCCIÓN

Los microcontroladores, desde su primera aparición, fueron en gran parte la solución a muchos problemas en el diseño de equipos de control en áreas tales como la informática, industria, medicina, etc., y todo gracias a sus grandes ventajas en relación a los elementos que en esa época existían. Desde ese entonces han evolucionado de tal manera, que en la actualidad su uso en todos estos campos se a masificado, ya que para cada aplicación en la que se lo requiera existe una solución, y su aplicación depende nada más que del ingenio con el cual se afronte el diseño y desarrollo de un sistema.

Teniendo en cuenta el papel que los microcontroladores desempeñan en la actualidad, se ha visto la necesidad de crear una herramienta que permita entender y visualizar de manera práctica la comunicación del mismo con distintos periféricos, para de este modo, impulsar el ingenio de futuros diseñadores de sistemas que requieran su utilización.

1.2. OBJETIVOS

El desarrollo de la presente tesis se basa en los siguientes objetivos :

- Crear un programa bajo un ambiente de Windows para mostrar en forma visual la comunicación del microcontrolador 8031 con distintos periféricos y memorias como respuesta a la ejecución de una instrucción que los involucre.

- Desarrollar una herramienta que permita suplir la falta de equipo de laboratorio de los cuales en los actuales momentos la universidad adolece.

- Fomentar la creatividad en el desarrollo de programas, puesto que el presente trabajo facilitará la tarea de aprendizaje mostrando en forma más objetiva la función de un microcontrolador.

- Asistir en la solución de problemas durante el desarrollo de ciertos programas antes de ser ensamblados, ya que será parte importante dentro de este programa, el poder consultar el estado de diferentes localidades de memorias y registros especiales.

1.3. ANTECEDENTES

Hasta el año de 1976, los microcontroladores como tal no existían, teniendose hasta ese entonces únicamente los microprocesadores. En ese año se pudo integrar en un solo elemento al microprocesador con elementos independientes como la memoria y los pórticos de entrada /salida de datos. Esta integración trajo como consecuencia el desarrollo de los denominados Microprocesadores de la Tercera Generación o más comunmente conocidos como Microcontroladores.

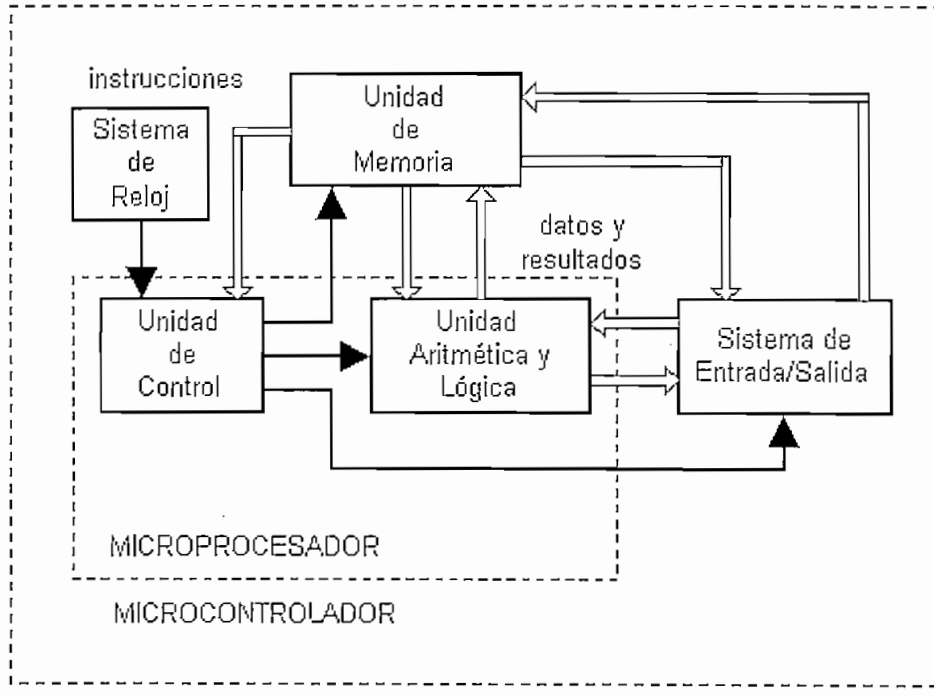


Fig.1.1 Diagrama de Bloques del Microcontrolador

La Fig.1.1 muestra mediante diagramas de bloques el límite que existe entre lo que es un microprocesador y lo que es un microcontrolador.

De lo anteriormente expuesto se puede concluir que un microcontrolador es un circuito que integra los siguientes subsistemas, aunque algunos no necesariamente incluyen todos :

- CPU
- RAM
- ROM
- Entrada/Salidas
- Contadores y Temporizadores
- Conversores Analógico/Digital y Digital/Analógico
- Gestión de Interrupciones.

Entre los microcontroladores más destacados que fueron desarrollados en esa época podemos mencionar los siguientes :

Las Familias de INTEL : MCS-48 , MCS-51 y MCS-52

La Familia de ZILOG : Z8

Los Microntroladores de MOTOROLA : 6801 , 6803 Y 6805.

1.4 DESCRIPCION DE LA FAMILIA MCS-51

Dentro de la familias de los microcontroladores desarrollados por INTEL, se encuentra la MCS-52/51 la misma que esta conformada por los microcontroladores tipo 8XX2 y 8XX1, los cuales presentan las siguientes diferencias :

8032/31 : Memoria de programas externa

8052/51 : Memoria de programas interna (ROM)

8752/51 : Memoria de programas interna (EPROM)

Versión con ROM	Versión sin ROM	Versión con EPROM	ROM Bytes	RAM Bytes	Temporizadores de 16 Bits	Tecnología
8051	8031	8751	4K	128	2	HMOS
8051AH	8031AH	8751H	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Tabla 1.1 Microcontroladores INTEL, Tipo 8XXX

Como se puede apreciar en la Tabla 1.1 la diferencia básica dentro de la familia MCS-51 entre el 8XX2 y 8XX1 es la mayor capacidad en RAM y ROM, siendo la capacidad de los 8XX2 el doble que para los 8XX1; además de poseer un temporizador adicional.

Las características principales que reúnen los microcontroladores que pertenecen a esta familia son :

- CPU de 8 bits
- Procesador booleano (operación sobre bits)
- 4 puertos de 8 bits
- Para el 8052, 256 bytes de memoria interna RAM útil para el usuario y 128 bytes correspondientes al área de los registros especiales (SFR). Para el 8051, 128 bytes útiles para el usuario y 128 bytes de los SFR.
- 8 Kbytes de ROM (8XX2) , 4 Kbytes para el 8XX1.
- Espacio de memoria de 64 Kbytes para datos externos
- El 8XX2 contiene 3 contadores - temporizador (timers). El 8XX1 solamente 2 .
- Comunicación asíncrona full-duplex
- 6 fuentes de interrupciones con 2 niveles de prioridad (5 para el 8XX1)

- 2 interrupciones externas

- 3 interrupciones de los temporizadores (2 para el 8XX1)

- 1 interrupción del pórtico serial

- Oscilador interno

Las variantes que presenta esta Familia de microcontroladores, depende también del fabricante, como se puede apreciar en la Tabla 1.2.

VARIANTE	# PINS	FABRICANTE	RAM INTERNA	MEMORIA DE PROGRAMA	XRAM
MCS251	40	Intel	1K	16K	0
80C509L	100qf	Siemens	256	64Kx	3K
80C517A	84	Siemens	256	64Kx	2K
80C537A	84	Siemens	256	32K	2K
80537	84	Siemens	256	64Kx	0
80517	84	Siemens	256	8K	0
73D2910	100q fp	SSI	256	128Kx	0
80C535A	068	Siemens	256	64Kx	1K
80CE558	80q fp	Philips	256	64Kx	768
80C515A	68	Siemens	256	32K	1K
80535	68	Siemens	256	64Kx	0
80515	68	Siemens	256	8K	0
80C535	68	Siemens	256	64Kx	0
80C51GB	68	Intel	256	64Kx	0
87C51GB	68	Intel	256	8K	0
80C592	68	Philips	256	64Kx	256
87C592	68	Philips	256	16K	256
87C598	80	Philips	256	32K	256
80C552	68	Philips	256	64Kx	0
87C552	68	Philips	256	8K	0
80C562	68	Philips	256	64Kx	0
SABC505C	44	Siemens	256	64Kx	256
SABC504	44	Siemens	256	64Kx	256
87C451	68	Philips	128	4K	0
80C451	68	Philips	128	64Kx	0
87453	68	Philips	256	8K	0
83CL580	56,64	Philips	256	6K	0
80C320	40	Dallas	256	64Kx	0
80C310	40	Dallas	256	64Kx	0
87C520	40	Dallas	256	16K	1K
80C51FX	40	Intel	256	64Kx	0
87C51FA	40	Intel	256	8K	0
87C51FB	40	Intel	256	16K	0
87C51FC	40	Intel	256	32K	0
8XC51FB	40	Philips	256	16K	0
87C51FXL	40	Intel	256	32K	0
80C152JD	68	Intel	256	64Kx	0
80C152	48	Intel	256	64Kx	0
8044	40	Intel	192	64Kx	0
80C575	40	Philips	256	64Kx	0
87C575	40	Philips	256	8K	0
80C576	40	Philips	256	8K	0
87C576	40	Philips	256	8K	0
SABC501	40	Siemens	256	64Kx	0
SABC502	40	Siemens	256	64Kx	256
80C528	40	Philips	256	64Kx	256
87C528	40	Philips	256	32K	256
89CE528	44	Philips	256	32KF	256
87C524	40	Philips	256	16K	256
80C550	40	Philips	128	4K	0

VARIANTE			INTERNA	DE PROGRAMA	
80CL781	40	Philips	256	64Kx	0
83CL781	40	Philips	256	16K	0
80CL782	40	Philips	256	64Kx	0
89S8252	40,44	Atmel	256	10KF	0
89C55	40,44	Atmel	256	20KF	0
89C52	40,44	Atmel	256	8KF	0
87C54	40	Intel	256	16KF	0
87C58	40	Intel	256	32K	0
87C52	40	Intel	256	8K	0
80C154	40	Matra	256	64Kx	0
83C154D	40	Matra	256	32K	0
83C154	40	OKI	256	16K	0
80C654	40	Philips	256	64Kx	0
87C652	40	Philips	256	8K	0
87C654	40	Philips	256	16K	0
83CE654	44q fp	Philips	256	16K	0
DS5000	40	Dallas	128	32K	32K
DS2250	40sim	Dallas	128	32K	32K
DS5001	80q fp	Dallas	128	64Kx	64K
80C851	40	Philips	128	64Kx	0
83C852	6	Philips	256	6K	0
8052	40	All	256	64Kx	0
8752	40	Intel	256	8K	0
80C52	40	Siemens	256	64Kx	0
88SC54C	8	Atmel	256	64Kx	512
80CL410	40	Philips	128	64Kx	0
80CL31	40	Philips	128	64Kx	0
80CL610	40	Philips	256	64Kx	0
83CL411	40	Philips	256	64Kx	0
89C51	40,44	Atmel	128	4KF	0
8751	40	All	128	4K	0
87C51	40	All	128	4K	0
8031	40	[illegible]	[illegible]	64Kx	0
8051	40	All	128	4K	0
80C31L	40	Matra	128	64Kx	0
87C752	28	Philips	64	2K	0
87C749	28	Philips	64	2K	0
87C751	24	Philips	64	2K	0
87C748	24	Philips	64	2K	0
87C750	24	Philips	64	1K	0
89C2051	20	Atmel	128	2KF	0
89C1051	20	Atmel	64	1KF	0

En la columna de Memoria de Programa las letras **x, F** significan memoria externa y flash respectivamente.

Tabla1.2 Microcontroladores MCS-51 de diferentes Casas Fabricantes

CAPITULO II

EL MICROCONTROLADOR MCS-8031

2.1 ARQUITECTURA Y ORGANIZACIÓN DE LA MEMORIA

2.1.1 ARQUITECTURA

El microcontrolador 8031 posee una tecnología de construcción HMOS, no tiene memoria interna de programa, la RAM interna útil para el usuario es de 128 bytes, 4 pórticos paralelos de 8 bits, 2 temporizadores/contadores de 16 bits, y un pórtico serial .

En la Fig.2.1 se muestra el diagrama de bloques del Microcontrolador 8031.

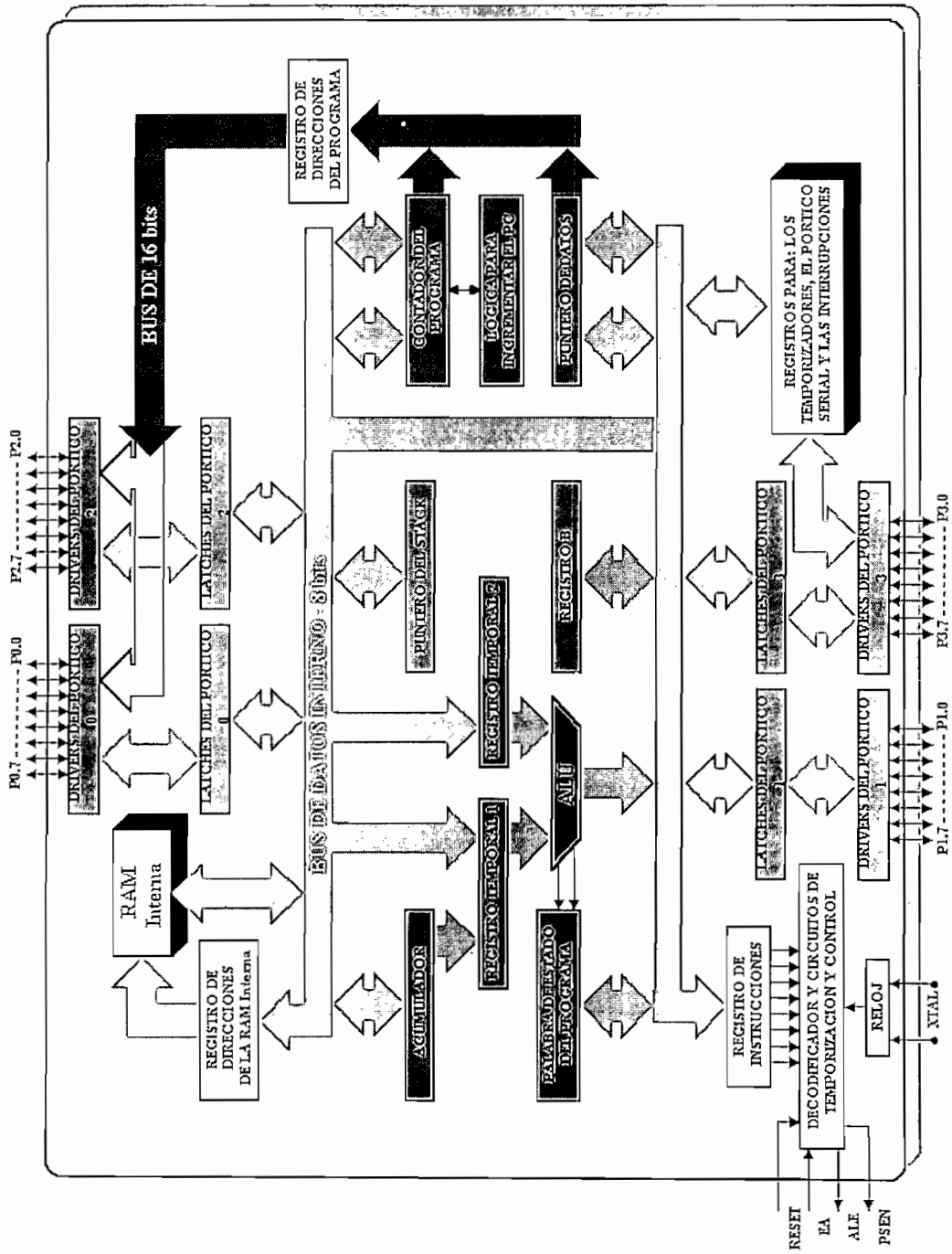


Fig.2.1 Diagrama de Bloques del Microcontrolador 8031

2.1.2 ORGANIZACIÓN DE LA MEMORIA

La arquitectura del microcontrolador MCS-8031 permite manejar memoria de datos incorporada en el chip como también direccionarla externamente, en cambio la memoria de programa (ROM) solo la puede manejar de manera externa.

El 8031 tiene tres bloques de memoria para direccionar :

- Máximo 64 Kbyte, para memoria de programa externa
- Máximo 64 Kbyte, para memoria de datos externa
- 256 byte, para memoria de datos interna

2.1.2.1 Memoria de Programa

El espacio de 64 Kbyte para memoria de programa consiste de una parte de la memoria externa al microcontrolador 8031, para accesarla, el pin $\overline{EA}$ del micro debe estar en 0_L, con ello el direccionamiento será desde la dirección 0000H hasta FFFFH como se muestra en la Fig. 2.2. Las primeras localidades (0000H hasta 0023H) de la memoria del programa , están reservadas como direcciones iniciales de las rutinas de servicio a las interrupciones , como se indica en la Tabla 2.1

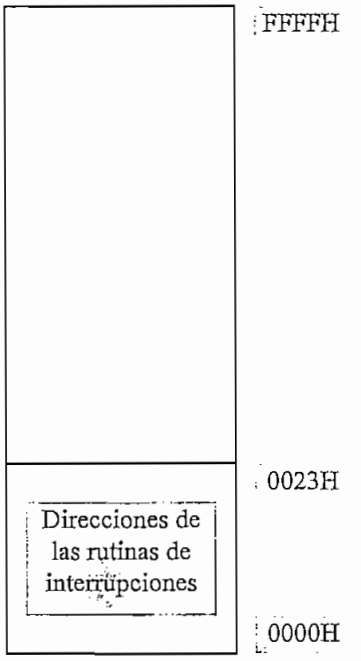


Fig.2.2 Organización de Memoria de Programa del 8031

INTERRUPCION	DIRECCION
Reset	0000H
Interrupción Externa 0	0003H
Desbordamiento de temporizador 0	000BH
Interrupción Externa 1	0013H
Desbordamiento de temporizador 1	001BH
Pórtico Serial	0023H

Tabla 2.1 Primeras localidades de Memoria Externa

2.1.2.2 Memoria de Datos

La memoria de datos, consiste de una parte interna y otra externa. La memoria de datos externa es accesada cuando se ejecuta una instrucción MOVX.

La memoria de datos interna está dividida físicamente en dos bloques para el 8031. Los 128 bytes bajos de memoria (00H hasta 7FH) constituyen el área de datos, y los 128 bytes altos (80H hasta FFH) forman los registros de funciones especiales SFR, como se muestra en la Fig.2.3, el área de datos de la memoria RAM interna se divide de la siguiente manera :

- Las 32 primeras localidades (00H hasta 1FH), son ocupadas por 4 Bancos de 8 registros cada uno (R0 .....R7).

- Las siguientes 16 localidades (20H hasta 2FH), contienen 128 bits que pueden ser direccionados independientemente con direcciones comprendidas entre 00H y 7FH.

- Las restantes 80 localidades (30H hasta 7FH), constituyen el área para datos.

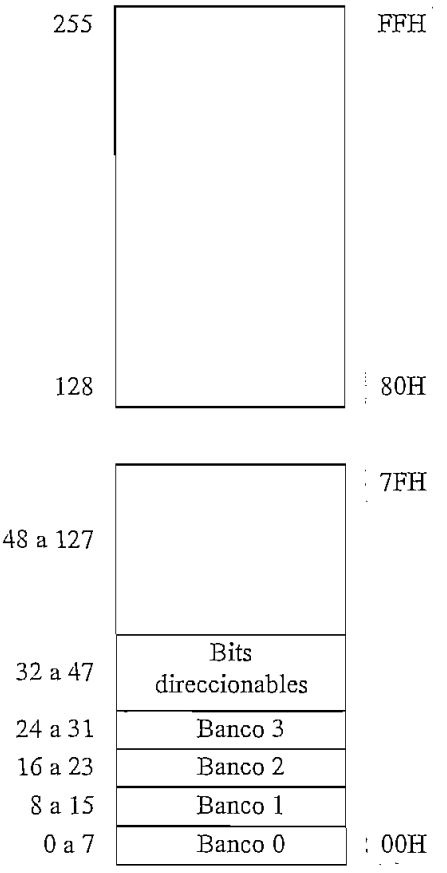


Fig.2.3 Organización de Memoria Interna de Datos para el 8031

2.2 MODOS DE DIRECCIONAMIENTO Y CONJUNTO DE INSTRUCCIONES

2.2.1 MODOS DE DIRECCIONAMIENTO

El 8031 usa 5 modos de direccionamiento :

- Direccionamiento por Registros

- Direccionamiento Directo
- Direccionamiento Indirecto
- Direccionamiento Inmediato
- Direccionamiento Indexado

En la Tabla 2.2 se muestra los modos de direccionamiento con los espacios de memoria que puede manejar cada uno.

TIPO DE DIRECCIONAMIENTO	AREA DE MEMORIA
Direccionamiento por registros	R0 ..... R7 A, B, C (bit del carry), DPTR
Modo directo	128 byte bajos de RAM interna Registros de funciones especiales
Modo indirecto	RAM interna (@R0, @R1, SP) RAM externa (@R0, @R1, @DPTR)
Modo Inmediato	Memoria de programa
Modo indexado	Memoria de programa (@ A + DPTR, @ A + PC)

Tabla 2.2. Modo de direccionamiento asociado al área de memoria

2.2.1.1 Direccionamiento por Registros

El direccionamiento por registros utiliza los 8 registros que van desde el R0 hasta R7 del banco de registros seleccionado a través de los bits RS1 y RS0 de la palabra de estado del programa PSW. También puede usar los registros ACC, B, DPTR y C (bit del carry).

2.2.1.2 Direccionamiento Directo

Este modo de direccionamiento permite accesar a los Registros de Funciones Especiales y a los 128 bytes bajos de la Memoria de Datos Interna.

2.2.1.3 Direccionamiento Indirecto

Se utiliza este modo de direccionamiento cuando se accesa a las 128 localidades bajas de datos de la RAM Interna, para ello se especifica la dirección del operando mediante el contenido de los registros R0 y R1 o SP. También se utiliza para direccionar localidades de RAM Externa con el uso de los registros R0, R1 y DPTR. A estos registros que contienen la dirección de memoria deseada se les denomina punteros.

2.2.1.4 Direccionamiento Inmediato

Se da cuando el operando se encuentra en las mismas localidades de la Memoria de Programa.

2.2.1.5 Direccionamiento Indexado

Sirve para operar con los contenidos de cualquier localidad de la de la Memoria del Programa, utilizando los contenidos de los registros DPTR o PC, al que se le suma el contenido del Acumulador.

2.2.2. CONJUNTO DE INSTRUCCIONES.

Al igual que la mayoría de los microprocesadores, el 8031 maneja un set de instrucciones, el cual ha sido dividido según su especialidad en :

* Instrucciones Aritméticas
* Instrucciones Lógicas
* Instrucciones de Transferencia de Datos
* Instrucciones de Saltos
* Instrucciones Booleanas

En conjunto, forman un set de 111 instrucciones .

Para poder interpretar las diferentes formas de representar los operandos en la Tabla 2.3a, primero describiremos que significan cada uno de ellos :

Rn : Registros R0-R7 del banco activo

direct : Dirección de 8 bits de la RAM interna .

Ri : Se refiere a los contenidos de los registros R0 y R1 del banco activo, como punteros que permiten el direccionamiento indirecto para acceder a la RAM interna (00-7F) o la RAM externa.

data : Operando constante de 8 bits.

data16 : Operando constante de 16 bits.

addr16 : Dirección de destino de 16 bits. Utilizada por las instrucciones LCALL Y LJMP para permitir el salto dentro del espacio de 64 Kbytes de la memoria de programa.

addr11 : Dirección de destino de 11 bits. Utilizada por las instrucciones ACALL y AJMP para permitir el salto dentro de la página de 2 Kbytes de la memoria de programa, donde se encuentra el primer byte de la siguiente instrucción.

rel : Salto relativo. Utilizado para la instrucción SJMP y todos los saltos condicionales. El rango del salto está comprendido entre -128 a +127 bytes a partir del primer byte de la siguiente instrucción.

bit : Direción de 8 bits para un bit direccionable de la RAM de datos o de los SFR .

Además, en la Tabla 2.3.b, se señalan las instrucciones que afectan a las banderas del registro PSW.

NEMONICO	DESCRIPCION	BYTES	PERIODO DE OSCILADOR
	INSTRUCCIONES LOGICAS		
ANL A,Rn	AND registro a acumulador	1	12
ANL A,direct	AND contenido de dirección a acumulador	2	12
ANL A,@Ri	AND RAM indirecta a acumulador	1	12
ANL A,#data	AND dato inmediato a acumulador	2	12
ANL direct,A	AND acumulador a contenido de dirección	2	12
ANL direct,#data	AND dato inmediato a contenido de dirección	3	24
ORL A,Rn	OR registro a acumulador	1	12
ORL A,direct	OR contenido de dirección a acumulador	2	12
ORL A,@Ri	OR RAM indirecta a acumulador	1	12
ORL A,#data	OR dato inmediato a acumulador	2	12
ORL direct,A	OR acumulador a contenido de dirección	2	12
ORL direct,#data	OR dato inmediato a contenido de dirección	3	24
XRL A,Rn	OR Exclusivo registro a acumulador	1	12
XRL A,direct	OR Exclusivo contenido de dirección a acumulador	2	12
XRL A,@Ri	OR Exclusivo RAM indirecta a acumulador	1	12
XRL A,#data	OR Exclusivo dato inmediato a acumulador	2	12
XRL direct,A	OR Exclusivo acumulador a contenido de dirección	2	12
XRL direct,#data	OR Exclusivo dato inmediato a contenido de dirección	3	24
CLR A	Borrar el acumulador	1	12
CPL A	Complementar el acumulador	1	12
RL A	Rotar acumulador a la izquierda	1	12
RLC A	Rotar acumulador a la izquierda a través del C	1	12
RR A	Rotar acumulador a la derecha	1	12
RRC A	Rotar acumulador a la derecha a través del C	1	12
SWAP A	Intercambiar los 4 bits bajos con los altos de Acumulador	1	12
	INSTRUCCIONES ARITMETICAS		
ADD A,Rn	Sumar registro a acumulador	1	12
ADD A,direct	Sumar contenido de dirección a acumulador	2	12
ADD A,@Ri	Sumar RAM indirecta a acumulador	1	12
ADD A,#data	Sumar dato inmediato a acumulador	2	12
ADDC A,Rn	Sumar registro a acumulador con llevo	1	12
ADDC A,direct	Sumar contenido de dirección a acumulador con llevo	2	12
ADDC A,@Ri	Sumar RAM indirecta a acumulador con llevo	1	12
ADDC A,#data	Sumar dato inmediato a acumulador con llevo	2	12
SUBB A,Rn	Sustraer registro de acumulador con debo	1	12
SUBB A,direct	Sustraer contenido de dirección de acumulador con debo	2	12
SUBB A,@Ri	Sustraer RAM indirecta de acumulador con debo	1	12
SUBB A,#data	Sustraer dato inmediato de acumulador con debo	2	12
INC A	Incrementar acumulador	1	12
INC Rn	Incrementar registro	1	12
INC direct	Incrementar contenido de dirección	2	12
INC @Ri	Incrementar RAM indirecta	1	12
DEC A	Decrementar acumulador	1	12
DEC Rn	Decrementar registro	1	12
DEC direct	Decrementar contenido dirección	2	12
DEC @Ri	Decrementar RAM indirecta	1	12
INC DPTR	Incrementar DPTR	1	24
MUL AB	Multiplicar A & B	1	48
DIV AB	Dividir A para B	1	48
DA A	Ajuste decimal del acumulador	1	12
	INSTRUCCIONES DE TRANSFERENCIA DE DATOS		
MOV A,Rn	Mover registro a acumulador	1	12
MOV A,direct	Mover contenido de dirección a acumulador	2	12

NEMONICO	DESCRIPCION	BYTES	PERIODO DE OSCILADOR
MOV A,@Ri	Mover RAM indirecta a acumulador	1	12
MOV A,#data	Mover dato inmediato a acumulador	2	12
MOV Rn,A	Mover acumulador a registro	1	12
MOV Rn,direct	Mover contenido de dirección a registro	2	24
MOV Rn,#data	Mover dato inmediato a registro	2	12
MOV direct,A	Mover acumulador a contenido de dirección	2	12
MOV direct,Rn	Mover registro a contenido de dirección	2	24
MOV direct,direct	Mover contenido de dirección a contenido de dirección	3	24
MOV direct,@Ri	Mover RAM indirecta a contenido de dirección	2	24
MOV direct,#data	Mover dato inmediato a contenido de dirección	3	24
MOV@Ri,A	Mover acumulador a RAM indirecta	1	12
MOV @Ri,direct	Mover contenido de dirección a RAM indirecta	2	24
MOV @Ri,#data	Mover dato inmediato a RAM indirecta	2	12
MOV DPTR,#data16	Cargar DPTR con dato inmediato de 16 bits	3	24
MOVC A,@A+DPTR	Mover código de byte relativo a DPTR a acumulador	1	24
MOVC A,@A+PC	Mover código de byte relativo a PC a acumulador	1	24
MOVX A,@Ri	Mover de RAM externa (8 bits) a acumulador	1	24
MOVX A,@DPTR	Mover de RAM externa (16 bits) a acumulador	1	24
MOVX @Ri,A	Mover acumulador a RAM externa (8 bits)	1	24
MOVX @DPTR,A	Mover acumulador a RAM externa (16 bits)	1	24
PUSH direct	Almacenar contenido de dirección dentro de Stack	2	24
POP direct	Tomar de Stack a contenido de dirección	2	24
XCH A,Rn	Intercambiar registro con acumulador	1	12
XCH A,direct	Intercambiar contenido de dirección con acumulador	2	12
XCH A,@Ri	Intercambiar RAM indirecta con acumulador	1	12
XCHD A,@Ri	Intercambiar 4 bits menos significativos de RAM indirecta con acumulador	1	12
	INSTRUCCIONES DE SALTOS		
ACALL addr11	Llamada a subrutina (11 bits)	2	24
LCALL addr16	Llamada a subrutina (16 bits)	3	24
RET	Retorno de subrutina	1	24
RETI	Retorno de interrupción	1	24
AJMP addr11	Salto Medio (11 bits)	2	24
LJMP addr16	Salto Largo (16 bits)	3	24
SJMP rel	Salto Corto	2	24
JMP @A+DPTR	Salto indirecto ralativo a DPTR	1	24
JZ rel	Salta si acumulador es cero	2	24
JNZ rel	Salta si acumulador no es cero	2	24
CJNE A,direct,rel	Compare contenido de dirección a acumulador y salte si no son iguales	3	24
CJNE A,#data,rel	Compare dato inmediato a acumulador y salte si no son iguales	3	24
CJNE Rn,#data,rel	Compare dato inmediato a registro y salte si no son iguales	3	24
CJNE @Ri,#data,rel	Compare dato inmediato a RAM indirecta y salte si no son iguales	3	24
DJNZ Rn,rel	Decremente registro y salte si no es cero	3	24
DJNZ direct,rel	Decremente contenido de dirección y salte si no es cero	3	24
NOP	Ninguna operación	1	12
	INSTRUCCIONES BOOLEANAS		
CLR C	Borrar el C	1	12
CLR bit	Borra contenido de bit	2	12
SETB C	Poner a 1 el C	1	12
SETB bit	Poner a 1 contenido de bit	2	12

NEMONICO	DESCRIPCION	BYTES	PERIODOS OSCILADOR
CPL C	Complementar el C	1	12
CPL bit	Complementar contenido de bit	2	12
ANL C,bit	AND contenido de bit a C	2	24
ANL C,/bit	AND contenido de bit complementado a C	2	24
ORL C,bit	OR contenido de bit a C	2	24
ORL C,/bit	OR contenido de bit complementado a C	2	24
MOV C,bit	Mover contenido de bit a C	2	12
MOV bit,C	Mover C a contenido de bit	2	24
JC rel	Salte si C es 1	2	24
JNC rel	Salte si C es cero	2	24
JB bit,rel	Salte si contenido de bit es 1	3	24
JNB bit,rel	Salte si contenido de bit es cero	3	24
JBC bit,rel	Salte si contenido de bit es 1, y borre el bit	3	24

Tabla 2.3.a Conjunto de Instrucciones del Microcontrolador MCS-8031

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

Tabla 2.3.b Instrucciones que afectan banderas

2.3 PERIFERICOS INTERNOS Y EXTERNOS DEL MICROCONTROLADOR

2.3.1 PERIFERICOS INTERNOS

2.3.1.1 Oscilador y Circuito de Reloj

XTAL1 y XTAL2 son la entrada y salida de un circuito inversor, el cual puede ser configurado como un dispositivo oscilador. Para el oscilador se puede utilizar indistintamente un cristal de cuarzo (Fig. 2.4) o un resonador cerámico.

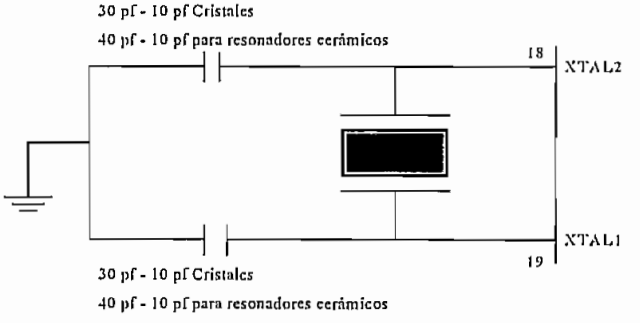


Fig. 2.4 Circuito oscilador con cristal de cuarzo

2.3.1.2. Temporización de la CPU

Un Ciclo de Máquina consiste de 6 estados (S1....S6). Cada estado esta formado por dos períodos de oscilación que se conocen como fases (P1 y P2). Así un Ciclo de Máquina consiste de 12 periodos de oscilación (12 ciclos de reloj), numerados como S1P1 (Estado1, Fase1) hasta S6P2 (Estado6, Fase2), entonces cada fase está formada por un período de oscilación y cada estado por dos períodos de oscilación. Típicamente cuando se realizan operaciones aritméticas o lógicas estas tienen lugar en la Fase 1 y durante la Fase 2 se da la transferencia interna (dentro del microcontrolador) de datos de registro a registro.

La Fig.2.5 muestra los periodos de traída y ejecución de instrucciones referidas a los Estados y Fases ya indicadas. La señal de ALE (Address Latch Enable) es activada dos veces durante cada ciclo de máquina; un vez durante S1P2 y S2P1, y otra durante S4P2 y S5P1.

La ejecución de una instrucción de un ciclo de máquina empieza en S1P2, cuando su opcode es tomado por el Registro de Instrucciones. Si esta instrucción es de dos bytes, el segundo byte es leído durante S4 del mismo ciclo de máquina. Si la instrucción es de un solo byte, el ciclo de lectura que tiene lugar durante S4 ignora el byte leído (el cual puede ser el próximo al opcode) y el PC (Program Counter, Contador del Programa) no se incrementa. En algunos casos la ejecución de la instrucción se completa al final de S6P2.

La mayor cantidad de instrucciones para el 8031 se dan durante un ciclo de máquina. Solamente las instrucciones MUL y DIV toman más de un ciclo de máquina en completarse, estas se demoran 4 ciclos de máquina.

Normalmente dos bytes de código son traídos de la Memoria de Programa durante todos los ciclos de máquina. La única excepción es la instrucción MOVX que consta de 1 byte y 2 ciclos de máquina y que accesa a la memoria de datos externa, por ello dos búsquedas o traídas son ignoradas mientras se direcciona y se toma los datos de la memoria externa, como se indica en la Fig. 2.5 D.

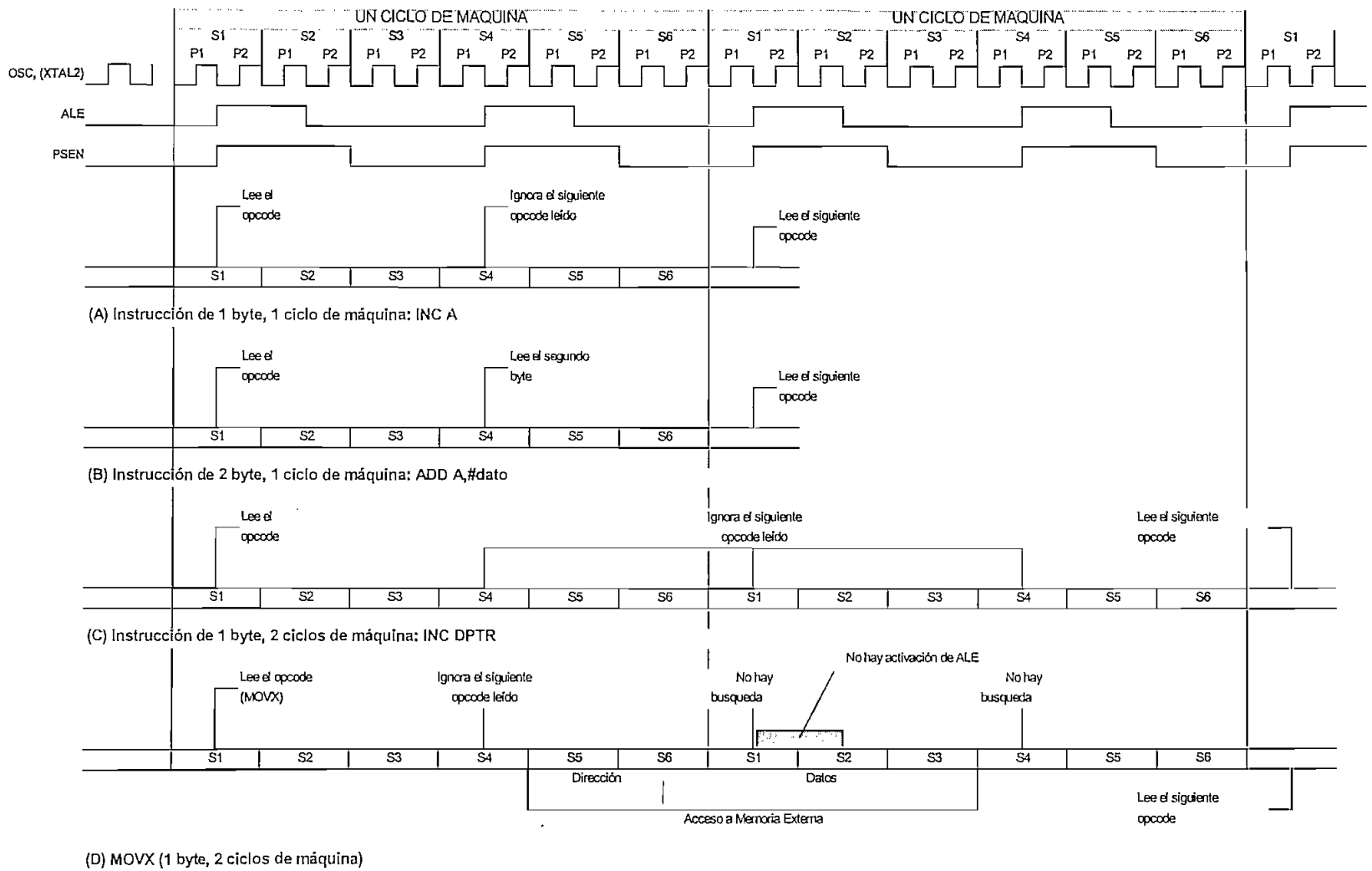


Fig.2.5 Secuencias de traída y ejecución de instrucciones

2.3.1.3 Estructura y Operación de los Pórticos Paralelos

Los cuatro pórticos paralelos de P0 a P3 son bidireccionables, consisten de un latch (SFR), un driver de salida y un buffer de entrada. Los drives de salida de los pórticos 0 y 2, y el buffer de entrada del pórtico 0 se utilizan para accesar a la memoria externa. En esta aplicación el driver de salida del pórtico 0 constituye los bits menos significativos de las direcciones de memoria externa, y los bits de direcciones más significativos son las salidas del pórtico 2, formando así un total de 16 bits para direccionar memoria externa. En este caso de direccionamiento de memoria externa se utiliza la función alterna de los pórticos 0 y 2. El pórtico 2 para el 8031 únicamente se utilizará para direccionar memoria externa, conteniendo su latch el byte más significativo del PC cuando se accesa a la Memoria de Programa, o el DPH que es el byte más significativo del DPTR cuando se accesa a la Memoria de Datos externa (dependiendo si la instrucción es MOVX @DPTR).

El pórtico 1 no posee función alterna. En cambio cada pin del pórtico 3 tiene una función alterna que se detalla en la Tabla 2.4.

PIN DEL PORTICO P3	FUNCION ALTERNA
P3.0	RXD, Entrada del pórtico serial
P3.1	TXD, Salida del pórtico serial
P3.2	$\overline{\text{INT0}}$, Interrupción externa 0
P.3.3	$\overline{\text{INT1}}$, Interrupción externa 1
P3.4	T0, Entrada externa del temporizador/Contador 0
P3.5	T1, Entrada externa del temporizador/Contador 1
P3.6	$\overline{\text{WR}}$, Escritura en memoria externa de datos
P3.7	$\overline{\text{RD}}$, Lectura de memoria externa de datos

Tabla 2.4 Función alterna del Pórtico 3

La función alterna trabaja solamente cuando el bit del latch en el SFR del pórtico esta en 1, de otra manera el pin del pórtico estará en 0.

En la Fig. 2.6 se muestra el esquema de un bit del pórtico paralelo, en el cual hacia la parte interna del microcontrolador se observa la línea del bus interno por donde circula información, para escribir en el latch si se activa la señal de escritura, para leer del terminal si se activa la señal de lectura del pin y para leer del latch si se activa la respectiva señal.

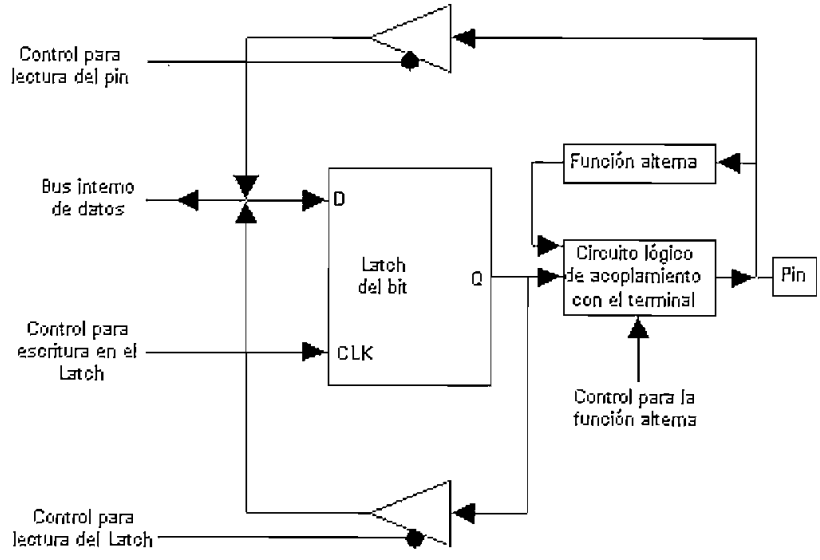


Fig. 2.6 Esquema del bit de un pórtico paralelo

2.3.1.4 Escritura en los Pórticos Paralelos

En la ejecución de una instrucción que cambia el contenido del latch de un pórtico, el nuevo valor llega al latch durante S6P2 del ciclo final de la instrucción. Sin embargo los latchs de los pórticos son muestreados por su buffers de salida durante

la Fase 1 de cualquier periodo de reloj, durante la Fase 2 los buffers de salida almacenan el valor visto durante la Fase 1. Consecuentemente, el nuevo valor almacenado en el latch del pórtico no aparecerá hasta que llegue la próxima Fase1, la que ocurre en S1P1 del próximo ciclo de máquina. Como la señal de control para la función alterna está desactivada (ver Fig 2.6), el valor de la salida del latch pasa por el circuito lógico de acoplamiento hacia el terminal que en este caso estará funcionando como salida de datos.

2.3.1.5 Lectura desde los Pines de los Pórticos Paralelos

Cuando se ejecutan instrucciones como MOV que tienen en el operando de origen una dirección de pórtico paralelo, al momento de ejecutarse la instrucción la señal de control que se activa es la de lectura desde el pin, debido a esto el valor que se encuentra en el pin pasará por el buffer de acoplamiento hacia el bus interno de datos. En este caso, como el circuito lógico de acoplamiento con el terminal deja pasar el valor de la salida del latch, para poder leer el verdadero valor de entrada en el pin es necesario que se escriba en el latch 1 lógico antes de leer desde los pines del pórtico. Ya que si el latch tiene 0 lógico, este sería el valor que se lee, independientemente del valor que se encuentra en el terminal. debido a que el 0 lógico prevalece sobre el 1 lógico.

2.3.1.6 Característica de Lectura-Modificación-Escritura

Aquellas instrucciones que leen un dato del latch del pórtico, posiblemente lo cambien, y luego lo vuelven a escribir en el latch se conocen como instrucciones de Lectura-Modificación-Escritura ; las cuales serán especificadas en el capítulo siguiente.

Al momento de leer el dato del pórtico, la señal de control que se activa es la de lectura desde el latch, debido a esto el valor que se encuentra en la salida del latch pasará por el buffer de acoplamiento hacia el bus interno de datos para modificarlo y volverlo a escribir en el latch.

2.3.1.7 Temporizadores y Contadores

El 8031 tiene dos registros de 16 bits que funcionan como temporizadores o contadores ascendentes, denominados: Timer 0 y Timer 1.

Si el registro es utilizado como temporizador, este se incrementa cada ciclo de máquina, por ello la razón de conteo es 1/12 de la frecuencia del oscilador.

En la función de contador, el registro es incrementado en respuesta a una transición de 1 a 0 en su pin de entrada externa correspondiente. Esta señal externa es

muestreada durante S5P2 de cada ciclo de máquina. El contador se incrementa cuando la muestra señala un nivel alto de la señal de entrada en un ciclo y un nivel bajo en el siguiente ciclo. El nuevo valor del contador aparece en el registro durante S3P1 del ciclo de máquina siguiente al ciclo en el cual se detectó la transición. Como esto toma dos ciclos de máquina, para reconocer la transición de 1 a 0 , la máxima tasa de conteo es 1/24 de la frecuencia del oscilador.

2.3.1.8 Timer 0 y Timer 1

Los dos Timers están presentes en el 8031. Para seleccionar su función de temporizador o contador se lo hace por medio del bit $C/\overline{T}$ del registro TMOD (Fig.2.7). Estos dos Timers tienen 4 modos de operación seleccionados a través de los bits M0 y M1 en TMOD . Los modos 0, 1 y 2 son lo mismo para ambos Timers, pero el modo 3 es diferente.

GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
TIMER 1				TIMER 0			

GATE: Bit que habilita la sincronización del Timer 0 y 1 con las señales externas que llegan a los terminales $\overline{INT0}$ y $\overline{INT1}$

$C/\overline{T}$: Bit para seleccionar entre temporizador y contador

$C/\overline{T}$ = 0, trabaja como temporizador, la entrada de reloj es la señal del oscilador del microcontrolador

$C/\overline{T}$ = 1, trabaja como contador, la entrada de reloj es la señal del terminal de entrada T0 y T1, respectivamente

M0 y M1: Bits para seleccionar el modo de operación de los Timers

M0 = 0 y M1 = 0 , Modo 0
M0 = 1 y M1 = 0 , Modo 1
M0 = 0 y M1 = 1 , Modo 2
M0 = 1 y M1 = 1 , Modo 3

Fig. 2.7 Estructura del Registro TMOD

En la Fig.2.8 se indica el esquema de funcionamiento de los Timers 0 y 1

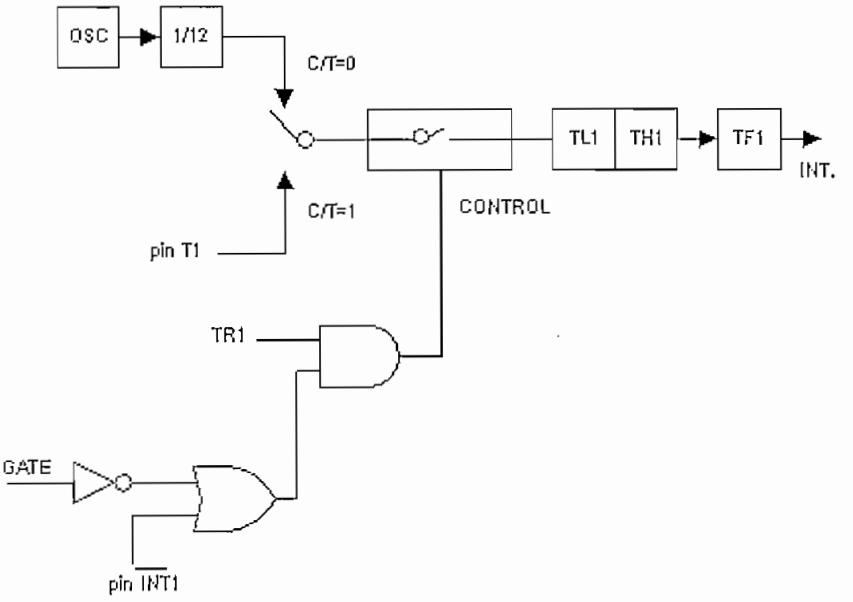


Fig.2.8 Esquema de funcionamiento de los Timers 0 y 1

<u>Modo 0</u>

Colocando los bits M0=M1=0 de cada Timer, estos trabajarán en el Modo 0. En este modo el registro contador es configurado con 13 bits de los 16 disponibles, repartiéndose 8 bits para TH0 o TH1 y los 5 bits de menor peso para TL0 o TL1. Cuando el registro contador pasa de todos sus bit 1 a todos 0 se activa la bandera de interrupción del Timer respectivo TF0 o TF1. La activación del Timer se da con el bit TR0 o TR1 puesto en 1 y GATE = 0 o $\overline{INT0}$ = 1, $\overline{INT1}$ = 0 (si colocamos GATE = 1, permitimos que el Timer sea controlado por con la entrada de interrupción externa

$\overline{INT0}$ o $\overline{INT1}$). TF0, TF1, TR0, TR1 son bits de control del registro especial TCON que se indica en la Fig. 2.9.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

ETIQUETA	DIRECCION COMO BIT	SIGNIFICADO
IT0	88H	Selección del tipo de señal de la $\overline{INT0}$ y Si igual a 0, se activa por nivel bajo Si igual a 1, se activa por flanco de bajada
IE0	89H	Bandera de la $\overline{INT0}$ Se pone a 1 cuando se detecta interrupción externa Se repone automáticamente al atender la rutina de la interrupción
IT1	8AH	Selección del tipo de señal de la $\overline{INT1}$
IE1	8BH	Bandera de la $\overline{INT1}$
TR0	8CH	Arranque/parada del Timer 0 Si igual a 1, activa el Timer 0 Si igual a 0, deshabilita Timer 0
TF0	8DH	Bandera de desbordamiento del Timer 0 Se repone automáticamente al atender la rutina de la interrupción
TR1	8EH	Arranque/parada del Timer 1
TF1	8FH	Bandera de desbordamiento del Timer 1

Fig. 2.9 Estructura del Registro TCON

Modo 1

El Modo 1 trabaja igual que el modo 0 excepto que el registro contador utiliza los 16 bits, es decir, 8 bits de THn como los más significativos y los 8 de TLn como los menos significativos.

Modo 2

En este modo de operación, ambos Timers trabajan como un contador de 8 bits con recarga automática. TL0 o TL1 hacen el papel de contadores de 8 bits, mientras que TH0 o TH1 contienen el valor de recarga automática para TL0 o TL1. La recarga automática se produce el momento en que los contadores se desbordan.

Modo 3

Para este modo el Timer 1 permanece deshabilitado como si TR1 fuese igual a 0. El Timer 0 divide sus dos registros TL0 y TH0 en dos contadores independientes de 8 bits cada uno. Los bits de control y bandera para el contador TL0 serán los del Timer 0, y para el TH0 los del Timer 1.

En este modo de operación el registro TH0 trabajará únicamente con la señal de frecuencia del oscilador dividida para 12, en cambio el registro TL0 si puede trabajar con la señal de frecuencia del oscilador dividida para 12 y con la señal que ingrese por el pin externo T0 del Pórtico 3.

2.3.1.9 Interrupciones

La comunicación asíncrona de los periféricos con la CPU, en ambos sentidos se da de dos maneras: por consultas (poleo) y por interrupciones.

En el modo de Consultas, se comprueba cíclicamente el estado de los registros de los dispositivos de E/S, usando instrucciones de programa. Suponiendo que existe un Periférico 1 conectado al Pórtico 0, mediante programa se consulta si el bit 0 del pórtico está activo, si es así significa que el periférico solicita la atención del sistema, atendiendo de esta manera el microcontrolador al periférico. Cuando termina, el programa activa el bit 1 del Pórtico 0, para indicarle al periférico que el proceso a culminado. Las desventajas de este modo radican en la necesidad de interrogar los bits de consulta en cada ciclo del programa, y que al periférico se lo atienda luego de realizar la consulta y no el momento en que solicita la intervención del microcontrolador.

Cuando se usa interrupciones, la interacción entre periférico y microcontrolador es directa. Este servicio se da en tiempo real, pueden eliminarse total o parcialmente los ciclos de consulta y permitir inhibir la interrupción cuando se considera que es inoportuna. Cuando termina la ejecución de una rutina de interrupción, el microcontrolador regresa al programa principal, continuando su tarea cíclica justo donde la dejo. El microcontrolador no atenderá una rutina de interrupción sin antes terminar de ejecutar la instrucción en la cual se detectó la interrupción.

El microcontrolador 8031 posee 5 fuentes de interrupciones, las mismas que se indican en la Fig. 2.10 y Tabla 2.5., en donde además se anticipa la prioridad de cada una de ellas.

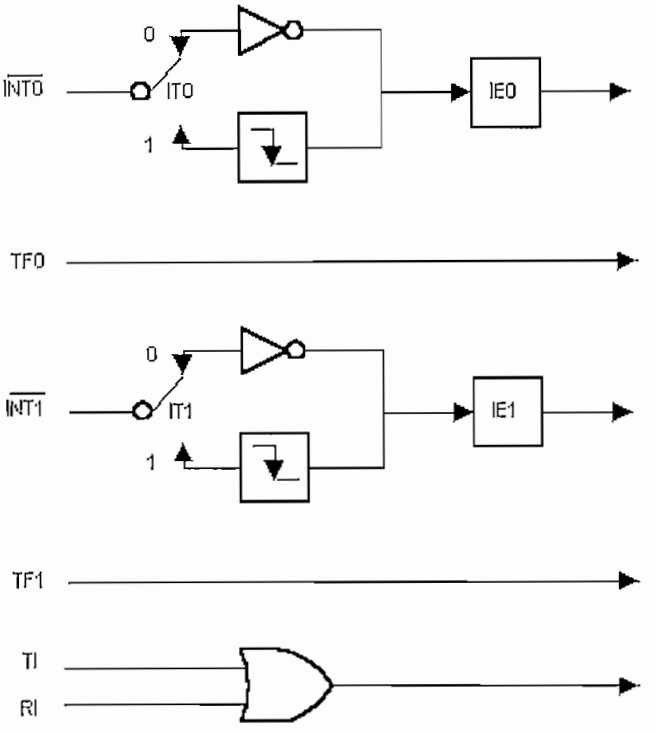


Fig. 2.10 Elementos generadores de interrupciones

FUENTE	EVENTO	BANDERA	DIRECCION Y MNEMONICO
Interrupción Externa 0, $\overline{INT0}$	Nivel de o lógico o transición negativa de la señal en el pin	IE0	0003H EXTI0
Interrupción del Timer0	Desbordamiento del Timer 0	TF0	000BH TIMER0
Interrupción Externa 1, $\overline{INT1}$	Nivel de o lógico o transición negativa de la señal en el pin	IE1	0013H EXTI1
Interrupción del Timer 1	Desbordamiento del Timer 1	TF1	001BH TIMER1
Interrupción del Pórtico Serial	Finalización de la transmisión o de la recepción de un dato realizado por parte del Pórtico Serial	TI o RI	0023H SINT

Tabla 2.5 Características de los elementos generadores de interrupciones

Como sabemos las interrupciones $\overline{INT0}$ y $\overline{INT1}$ pueden ser activadas por nivel o transición, dependiendo del estado de los bits IT0 e IT1 respectivamente. Las banderas que generan estas interrupciones son borradas por hardware el momento que se establece la rutina de servicio a la interrupción, solamente si ésta fue activada por transición. Si la interrupción se activa por nivel, la señal externa activa la bandera, y se deshabilita por software poniendo el bit de la bandera del registro correspondiente en cero. En la Fig.2.11 se indica el Registro IE que permite habilitar las interrupciones, para que se establezcan cuando se activa su bandera.

En el caso de interrupciones generadas por desbordamiento de temporizadores, las banderas son borradas por hardware el momento de establecerse la rutina de servicio a la interrupción.

La interrupción generada por el Pórtico Serial, sea por R1 o T1, no es borrada por hardware. Para borrar esta bandera, la rutina de servicio determinará si fue R1 o T1 la que generó la interrupción y luego de ello el bit de la bandera debe ser borrado por software.

Es necesario mencionar que todos los bits de las banderas que generan interrupciones pueden ser habilitadas individualmente por software, para ello únicamente debemos manejar sus bits en el Registro de Funciones Especiales (SFR).

secuencia de poleo interno determina cual de las dos es atendida. La secuencia de poleo interno se realiza en base a la prioridad implícita de las interrupciones, definida en la Fig.2.12.

-	-	*	PS	PT1	PX1	PT0	PX0

ETIQUETA	DIRECCION COMO BIT	SIGNIFICADO
PX0	B8H	Determina prioridad de la interrupción Externa 0
PT0	B9H	Determina prioridad de la interrupción del Timer 0
PX1	BAH	Determina prioridad de la interrupción Externa 1
PT1	BBH	Determina prioridad de la interrupción del Timer 1
PS	BCH	Determina prioridad de la interrupción del Pórtico Serial
*	BDH	*
-	BEH	Reservado
-	BFH	Reservado

Mayor

Menor

Prioridad Implícita

Para todos los Bits: Si es igual a 1, define alta prioridad para la interrupción

Fig.2.12. Estructura del Registro IP

Debe entenderse que este poleo interno se utiliza únicamente cuando ocurren peticiones simultáneas de atención a interrupciones del mismo nivel de prioridad.

2.3.1.11 Manejo de Interrupciones

Las banderas de interrupciones son muestreadas durante S5P2 de cada ciclo de máquina. Estas muestras se estructuran durante el siguiente ciclo de máquina. El sistema de interrupciones generará un LCALL a la dirección apropiada de servicio a

la interrupción, siempre y cuando no sea bloqueada por una de las siguientes condiciones:

- Una interrupción de igual o mayor nivel de prioridad esta es progreso
- No se a finalizado la instrucción que en ese momento está en proceso. Luego de finalizada se hará el direccionamiento para el servicio a la interrupción
- Si la instrucción en proceso es RETI o un acceso a los registros IE o IP. Esta condición asegura que al menos una instrucción más se ejecutará antes de establecer el servicio a la interrupción

El escrutinio se repite en cada ciclo de máquina. los valores escrutados corresponden a la activación de la interrupción presente en S5P2 del ciclo de máquina anterior.

Si una rutina de servicio a interrupción no puede ser atendida porque se a producido una condición de bloqueo, puede suceder:

- Que la bandera está todavía activada luego de desaparecer la situación de bloqueo. Si es así la interrupción será atendida.
- Si se desactiva la bandera de interrupción durante la situación de bloqueo, la interrupción no será atendida, salvo el caso de que sea solicitada nuevamente.

Todo el ciclo de manejo de las interrupciones se muestra en la Fig. 2.13, considerando que el ciclo de máquina C2 coincide con el último ciclo de una

instrucción en ejecución y no sea una instrucción RETI o un acceso a los registros IE o IP, es decir, existen condiciones idóneas.

Puede suceder que una interrupción de prioridad más elevada se active antes del estado S5P2 del ciclo de máquina etiquetado C3; por lo dicho anteriormente, esta interrupción será direccionada durante los ciclo C5 y C6, sin que ninguna instrucción de la rutina de prioridad más baja haya sido ejecutada.

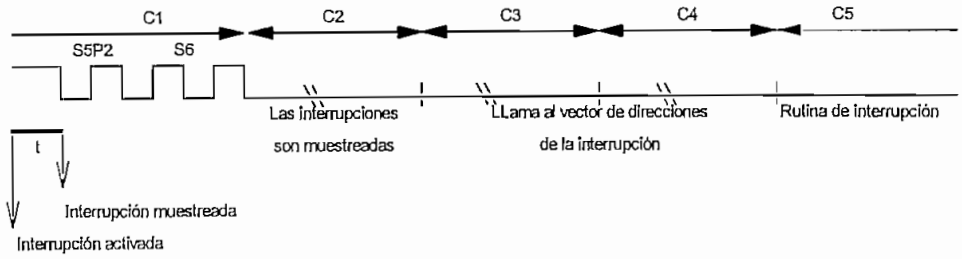


Fig. 2.13. Diagrama de tiempo de respuesta a interrupciones

Una vez validada la interrupción y producido el salto a la rutina de servicio, el microcontrolador guarda en el Stack el contenido del PC, y no resguarda el registro PSW.

La rutina de servicio a la interrupción debe finalizar con la instrucción RETI, para indicarle al microcontrolador que tal rutina a finalizado. Luego de ello se recupera del Stack los dos bytes de dirección del programa principal que permiten que se siga ejecutando la instrucción siguiente a la que en proceso recibió la interrupción y que concluyó antes de pasar el proceso de tratamiento.

En una interrupción simple, el tiempo de respuesta a la misma siempre será más de 3 ciclo de máquina y menos de 8.

2.3.1.12 Pórtico Serial

El Pórtico Serial establece una comunicación en Full Duplex, lo que significa que puede transmitir (a través de TXD) y recibir (a través de RXD) datos simultáneamente. Para recibir datos dispone de un buffer que le permite iniciar la recepción de un segundo byte antes de haber leído el byte anterior del registro de recepción, pero si el primer byte recibido no es leído durante la recepción en el buffer del segundo byte uno de los dos bytes se pierde. El registro SBUF se utiliza para acceder a los registros de transmisión y recepción, la dirección de este registro en el SFR es 99H. Al escribir el SBUF carga el byte a transmitir y leyendo del SBUF se accede al byte recibido.

2.3.1.13 Registro de Control y Estado SCON

El registro de control y estado del Pórtico Serial es nombrado como SCON (Serial Port Control Register) mostrado en la Fig. 2.14. Este registro contiene:

- Dos bits para seleccionar el modo de operación (Modo 0 ,1, 2, 3 que se resumirán posteriormente)
- El estado del 9no a ser transmitido o recibido
- Las banderas de interrupción para transmisión y recepción.

SM0	SM1	SM2	REM	TB8	RB8	TI	RI

ETIQUETA		DIRECCION COMO BIT		SIGNIFICADO	
RI		98H		Bandera de interrupción para recepción Se activa por hardware al finalizar la recepción el 8vo bit en el Modo 0 o hacia la mitad del intervalo de tiempo del bit de parada en los otros modos (excepto ver SM2) Debe ser desactivado por software	
TI		99H		Bandera de interrupción para transmisión Se activa por hardware al finalizar la transmisión del 8vo bit en el Modo 0 o al comienzo del bit de parada en los otros modos Debe ser desactivado por software	
RB8		9AH		En los Modos 2 y 3 es el 9no bit que se recibe En el Modo 1, si SM2 = 0 RB8 es el bit de parada En Modo 0 no se utiliza	
TB8		9BH		9no bit de datos en los Modos 2 y 3 Es programado por el usuario. Habitualmente es el bit de paridad	
REM		9CH		Si igual a 1 (por software) permite la recepción Si igual a cero no la permite	
SM2		9DH		En los Modos 2 y 3, si SM2 = 1 entonces RI no se activará si el 9no bit de datos RB8 es igual a 0 En el Modo 1, si SM2 = 1 entonces RI no se activará si el bit de parada no fue recibido En Modo 0, SM2 debe ser 0	
SM1	SM0	9EH	9FH	Especifican el modo	
SM1	SM0	MODO		DESCRIPCION	VELOCIDAD
0	0	0		Desplaza 8 bits	Reloj/12
1	0	1		8 bits - UART*	Variable
0	1	2		9 bits - UART	Reloj/64 o Reloj/32
1	1	3		9 bits - UART	Variable

* UART = Universal Asynchronous Receiver and Transmiter.
Reloj = frecuencia del oscilador del microcontrolador.

Fig. 2.14. Estructura del registro SCON

2.3.1.14 Velocidad de Comunicación

La velocidad de comunicación de cada modo se indicó de manera general en última parte de la Fig. 2.14. Ahora se explica más detalladamente como se logra establecer la velocidad de comunicación para los modos que lo ameriten.

Los Buadios en Modo 2 dependen del valor del bit SMOD del registro PCON, según la siguiente ecuación:

$$\text{Baudios en Modo 2} = \frac{2^{SMOD}}{64} * (frecuencia.oscilador) \quad (1)$$

El registro de control PCON (Power Control Register), posicionado en la dirección de memoria 87H, que tiene como misión configurar los modos de trabajo de bajo consumo, y de asignar parcialmente la velocidad de comunicación del Pórtico Serial en los modos 1, 2 y 3 (PCON.7).

Cuando el Timer 1 es utilizado como generador de Baudios para los Modos 1 y 3, por el valor de carga y desbordamiento del registro contador del Timer y el valor de SMOD, se lo hace según la ecuación siguiente:

$$\text{Baudios en Modo 2} = \frac{2^{SMOD}}{64} * (\text{relacion de desbordamiento Timer 1}) \quad (2)$$

La interrupción del Timer 1 debería ser deshabilitada para esta aplicación. El Timer puede ser configurado como temporizador o contador en cualquiera de sus tres

modos de operación. La mayoría de aplicaciones configuran el Timer como temporizador para que opere en el modo de autorecarga, si este fuera nuestro caso la ecuación (2) queda así:

$$\text{Baudios en Modo 1,3} = \frac{2^{SMOD}}{32} * \frac{frecuencia.oscilador}{12*\left[256-(TH1)\right]} \qquad (3)$$

Si permitimos que la interrupción del Timer este habilitada y lo configuramos para que trabaje con los 16 bits , podemos lograr velocidades de comunicación muy bajas. La interrupción la utilizamos para efectuar la recarga mediante software del registro de conteo.

Las velocidades de comunicación comúnmente utilizadas con el Timer 1, se listan en la Tabla 2.6.

BAUDIOS	FRECUENCIA OSCILADOR MHz	SMOD	TIMER 1		
			$C/\overline{T}$	MODO	VALOR DE RECARGA
Modo 0 Max: 1 Mhz	12	X	X	X	X
Modo 2 Max: 375 K	12	1	X	X	X
Modos 1,3: 62.5 K	12	1	0	2	FFH
119.2 K	11.059	1	0	2	FDH
9.6 K	11.059	0	0	2	FDH
4.8 K	11.059	0	0	2	FAH
2.4 K	11.059	0	0	2	F4H
1.2 K	11.059	0	0	2	E8H
137.5 K	11.986	0	0	2	1DH
110 K	6	0	0	2	72H
110 K	12	0	0	1	FEEBH

Tabla 2.6. Velocidades más utilizadas

2.3.1.15 Modos de Operación

Modo 0

Como ya se había indicado, los datos manejados por el Pórtico Serial entran y salen a través del pin RXD, TXD es el pin de salida para los pulsos de desplazamiento (shift clock). Ocho bits de datos son transmitidos o recibidos, empezando por el memos significativo (LSB).

La transmisión es iniciada por alguna instrucción que utilice como destino el registro SBUF. La señal de escritura sobre SBUF en S6P2, también carga un 1 en la posición del 9no bit del registro de desplazamiento transmitido e indica al bloque de control TX que comience la transmisión. Un ciclo de máquina completo transcurre entre la señal de escritura en SBUF y la activación de la señal interna del microcontrolador SEND.

La señal interna SEND habilita la salida del dato contenido en el registro de desplazamiento por RXD, y también habilita la salida de los pulsos de desplazamiento por TXD, Los pulsos de desplazamiento permanecen a bajo durante S3, S4 y S5 de cada ciclo de máquina ; y arriba durante S6 , S1 y S2.

En la Fase S6P2 de cada ciclo de máquina en la cual la señal interna SEND es activada, los bits contenidos en el registro de desplazamiento transmisor son desplazados a la derecha una posición. Estos bits que van quedando vacíos a la izquierda son llenados con ceros. Cuando el bit más significativo (MSB) del byte que se está transmitiendo se encuentra en la posición de salida del registro de desplazamiento, entonces el 1 que fue cargado en la posición novena está justo a la izquierda del MSB y todas las posiciones a la izquierda de este contienen ceros. Esta condición determina que el bloque de control TX haga un último desplazamiento, desactive SEND y active la bandera TI. Estas dos acciones ocurren en la Fase S1P1 del décimo ciclo de máquina luego de la escritura en SBUF. (Ver Fig.2.15)

La recepción de datos empieza por la condición REM = 1 y RI = 1. En S6P2 del siguiente ciclo de máquina, el bloque de control RX escribe los bits 11111110 en el registro de desplazamiento receptor, y en la siguiente fase de reloj activa la señal interna RECEIVE.

La señal RECEIVE habilita los pulso de desplazamiento simétricos. Estos pulsos efectúan la transmisión en S3P1 y S6P1 de cada ciclo de máquina. En S6P2 de cada ciclo de máquina en el cual RECEIVE es activado, los bits contenidos en el registro de desplazamiento receptor son desplazados una posición a la izquierda. El valor que se coloca en la parte derecha es aquel muestreado por el pin RXD en S5P2 del mismo ciclo de máquina.

Como los bits de datos ingresan por la parte derecha, los bits 11111110 colocados previamente se desplazan hacia la izquierda. Cuando el cero ubicado al extremo derecho llega al extremo izquierdo en el registro de desplazamiento, en esta condición el bloque de control RX realiza un ultimo desplazamiento y carga SBUF. En S1P1 del décimo ciclo de máquina después de la escritura en SCON que borra RI, RECEIVE es desactivado y activa RI. (Ver Fig.2.15)

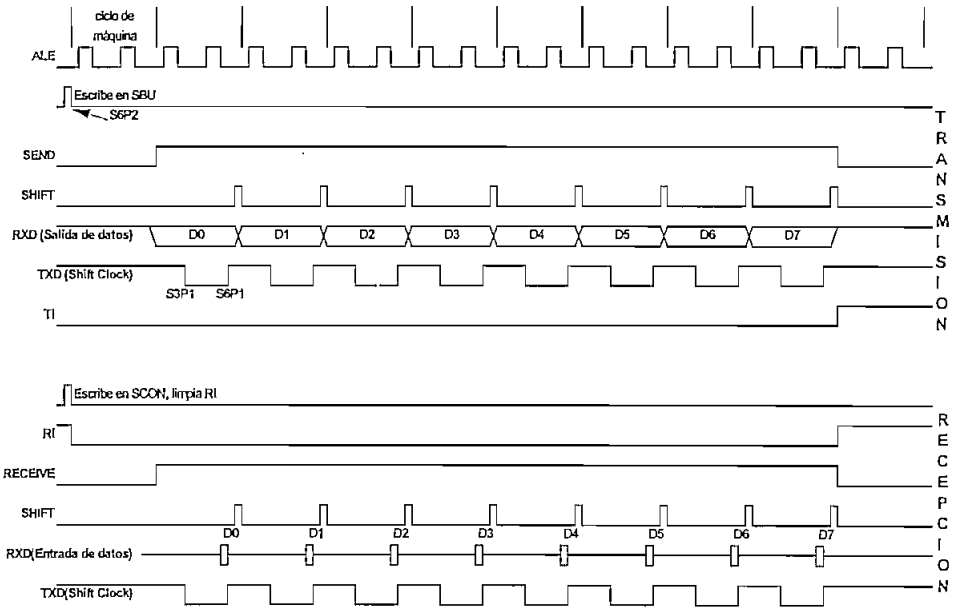


Fig. 2.15 Temporización para el Modo 0, Transmisión y Recepción

Modo 1

En este modo se manejan 10 bits, la transmisión se realiza a través de TXD y la recepción por RXD. Estos 10 bits están divididos en : un bit de inicio (0,start), ocho bits de datos y un bit de parada(1, stop). Para la recepción el bit de parada va dentro de RB8.

La transmisión se inicia con alguna instrucción que utilice el registro SBUF como destino. La señal de escritura en SBUF carga un 1 en la novena posición del registro de desplazamiento transmisor y se activa el bloque de control TX. La transmisión comienza en S1P1 del ciclo de máquina siguiente al ciclo de sobrepasamiento del Timer1 dividido por 16, así pues el tiempo de bit esta sincronizado por el contador divisor por 16 y no por la señal de escritura en SBUF.

La primera operación del bloque de control TX es activar la señal de control $\overline{\text{SEND}}$ (Como se muestra en la Fig. 2.16), que pone el bit de inicio en TXD. Un tiempo de bit después se activa la señal DATA, la cual habilita la salida de los bits del registro de desplazamiento transmisor a través de TXD. El primer pulso de desplazamiento ocurre un tiempo de bit después de la salida del primer bit de datos.

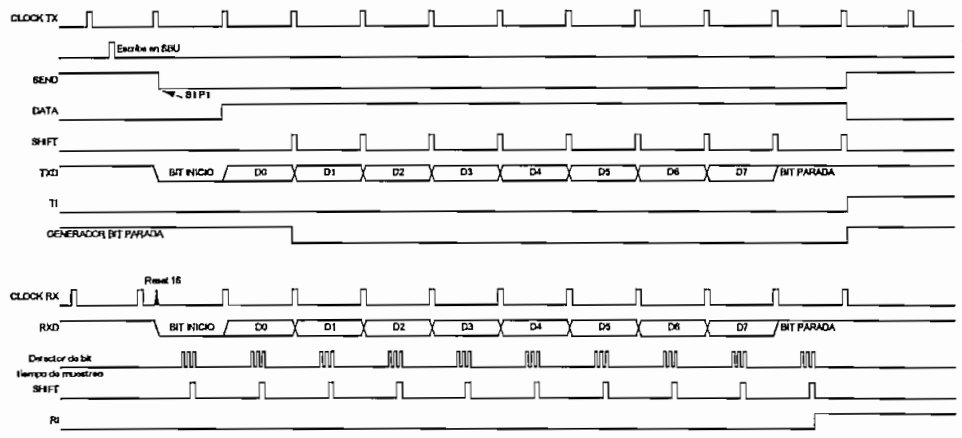


Fig.2.16 Temporización para el Modo 1, Transmisión y Recepción

Al desplazarse los bits de datos hacia la derecha, en la parte izquierda del registro de desplazamiento se colocan ceros. El momento en que MSB se encuentre en la

posición de salida del registro, en su lado izquierdo se encuentra el noveno bit seteado en 1 junto con los ceros que se fueron colocando. En esta condición la unidad de control realiza un último desplazamiento, desactiva $\overline{\text{SEND}}$ y activa TI esto ocurre en el décimo ciclo de desbordamiento del Timer dividido para 16 después de la instrucción de escritura en SBUF.

La recepción empieza por la detección de una transición de flanco descendente en RXD, por esta razón este pin es muestreado 16 veces independientemente de la velocidad que haya sido establecida. Cuando una transición es detectada el Timer 1 es puesto a cero y el dato 1FFH escrito en el registro de desplazamiento receptor, colocando en cero el Timer 1 sincroniza los ciclos de desbordamiento con los tiempos de bits entrantes. Los 16 estados del contador dividen cada tiempo de bit en 16 intervalos, en los estados 7, 8 y 9 de cada tiempo de bit el bit detector muestrea el valor de RXD y se acepta el valor que ha sido leído en al memos dos de las tres muestras, esto se hace para evitar el ruido eléctrico.

Si el valor aceptado durante el primer tiempo de bit no es cero, los circuitos receptores son reseteados y la unidad vuelve a alertarse para detectar otra transición de flanco descendente. Esto se hace para rechazar falsos bits de inicio, si el bit de inicio resulta correcto se ubica dentro del registro de desplazamiento y lo mismo sucede con el resto de la cadena de bits.

Debido a que los bits de datos ingresan en el registro de desplazamiento por la derecha, los unos son desplazados fuera por la izquierda. Cuando el bit de inicio llega a la novena posición alerta al bloque de control RX para hacer un último desplazamiento, carga SBUF y RB8 y además activa RI. La señal que se encarga de hacer esto, se generará si las siguientes condiciones se cumplen en el momento en que el pulso final de desplazamiento se da :

- RI = 0 y
- SM2 = 0 o bit de parada es igual a 1

Si una de estas dos condiciones no son satisfechas, se pierde inevitablemente la cadena de bits. En cambio si las condiciones se cumplen el bit de parada se ubica en RB8, los 8 bits de datos irán al registro SBUF y RI se activa. En este intervalo de tiempo ya sea que las condiciones indicadas se cumplan o no, la unidad vuelve a muestrear el pin RXD en busca de un flanco descendente.

Modos 2 Y 3

Para este modo de trabajo se manejan 11 bits, la transmisión se realiza a través de TXD y la recepción por RXD. Estos 11 bits están divididos en : un bit de inicio (0,start), ocho bits de datos, un noveno bit programable y un bit de parada (1, stop).

El noveno bit de datos en transmisión es la imagen del bit TB8, pudiendo asignarle el valor de 1 o 0 ; en recepción se refleja en el bit RB8.

En este modo la transmisión también se inicia con cualquier instrucción que utilice el registros SBUF como destino. La señal de escritura en SBUF carga un 1 en la novena posición del registro de desplazamiento transmisor y se activa el bloque ce control TX. La transmisión comienza en S1P1 del ciclo de máquina siguiente al ciclo de desbordamiento del Timer1 dividido por 16, así pues el tiempo de bit esta sincronizado por el contador divisor por 16 y no por la señal de escritura en SBUF.

La primera operación del bloque de control TX es activar la señal de control $\overline{\text{SEND}}$ (Como se muestra en la Fig. 2.17), que pone el bit de inicio en TXD. Un tiempo de bit después se activa la señal DATA, la cual habilita la salida de los bits del registro de desplazamiento transmisor a través de TXD. El primer pulso de desplazamiento ocurre un tiempo de bit después de la salida del primer bit de datos, también en ese instante el registro de desplazamiento envía un 1 como bit de parada que ocupa la posición del noveno bit del registro de desplazamiento, después se almacenan ceros en dicho registro ; como los bits de datos son desplazados al exterior por la derecha por la izquierda se introducen bits ceros. En la condición del último bit de salida se alerta al bloque de control TX para que haga el último desplazamiento e inmediatamente desactive $\overline{\text{SEND}}$ y active TI. Esto sucede en el undécimo ciclo de desbordamiento del Timer después de la escritura en SBUF.

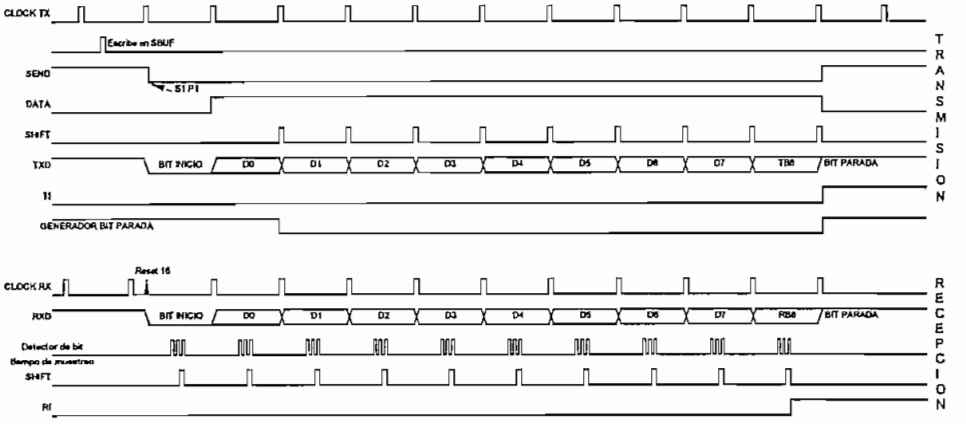


Fig. 2.17 Temporización para el Modo 2 y 3, Transmisión y Recepción

La recepción para estos Modos funciona de idéntica forma que en el Modo 1, pero las condiciones que deben ser satisfechas el momento en que el pulso final de desplazamiento se produce varían así :

- RI = 0 y
- SM2 = 0 o el noveno bit recibido igual a 1

Si una de estas dos condiciones no son satisfechas, se pierde inevitablemente la cadena de bits y RI no se activa. En cambio si las condiciones se cumplen el noveno bit de datos se ubica en RB8, los 8 bits de datos irán al registro SBUF y RI se activa. En este intervalo de tiempo ya sea que las condiciones indicadas se cumplan o no, la unidad vuelve a muestrear el pin RXD en busca de un flanco descendente.

Tome en cuenta que el valor del bit de parada recibido es irrelevante para SBUF, RB8 o RI.

2.3.2 PERIFERICOS EXTERNOS

2.3.2.1 Acceso a Memoria Externa

El acceso a memoria externa es de dos tipos:

- Acceso a Memoria Externa de Programa
- Acceso a Memoria Externa de Datos.

EL acceso a la Memoria Externa de Programa usa la señal de $\overline{PSEN}$ (Program Strobe Enable) como habilitación de lectura. En cambio para accesar a la Memoria Externa de Datos se usa $\overline{RD}$ o $\overline{WR}$ para habilitar la memoria.

La traída de datos de la Memoria Externa de Programa (que para el 8031 es la única área para memoria de programa ya que no la posee Internamente) siempre usa 16 bits de direcciones, para la Memoria Externa de Datos se puede usar 16 bits (MOVX @DPRT) o 8 bits (MOVX @Ri) para direccionarla. La señal de ALE (Address Lacth Enable) debería ser usada para capturar el byte de dirección en un latch externo al microcontrolador. Entonces en un ciclo de escritura, el byte de datos a ser escrito aparece en el pórtico 0 justo antes de que $\overline{WR}$ es activada y permanecerá allí hasta que $\overline{WR}$ sea desactivada. En un ciclo de lectura el byte que esta ingresando es

aceptado en el pórtico 0 justo antes de que la señal de habilitación lectura sea desactivada. Durante cualquier acceso a memoria externa, el CPU escribe 0FFH en el latch del pórtico 0, eliminando de esta manera cualquier información que pudo almacenarse en el SFR del pórtico 0.

PSEN

La señal de habilitación de memoria externa $\overline{\text{PSEN}}$, no se activa cuando se trabaja con la memoria interna. Cuando el CPU esta accesando a la Memoria Externa del Programa, $\overline{\text{PSEN}}$ es activada dos veces en cada ciclo de máquina (excepto durante una instrucción MOVX) sea o no que el byte traído se requiera para la instrucción que esta siendo leída. Cuando $\overline{\text{PSEN}}$ es activada su temporización no es la misma que para $\overline{\text{RD}}$. Un ciclo completo de $\overline{\text{RD}}$ incluyendo activación y desactivación de ALE y $\overline{\text{RD}}$, toma 12 periodos de oscilación. Un ciclo completo de $\overline{\text{PSEN}}$ incluyendo activación y desactivación de ALE y $\overline{\text{PSEN}}$, toma 6 períodos de oscilación. La secuencia de ejecución de estos dos tipos de ciclos de lectura se muestran en la Fig. 2.18

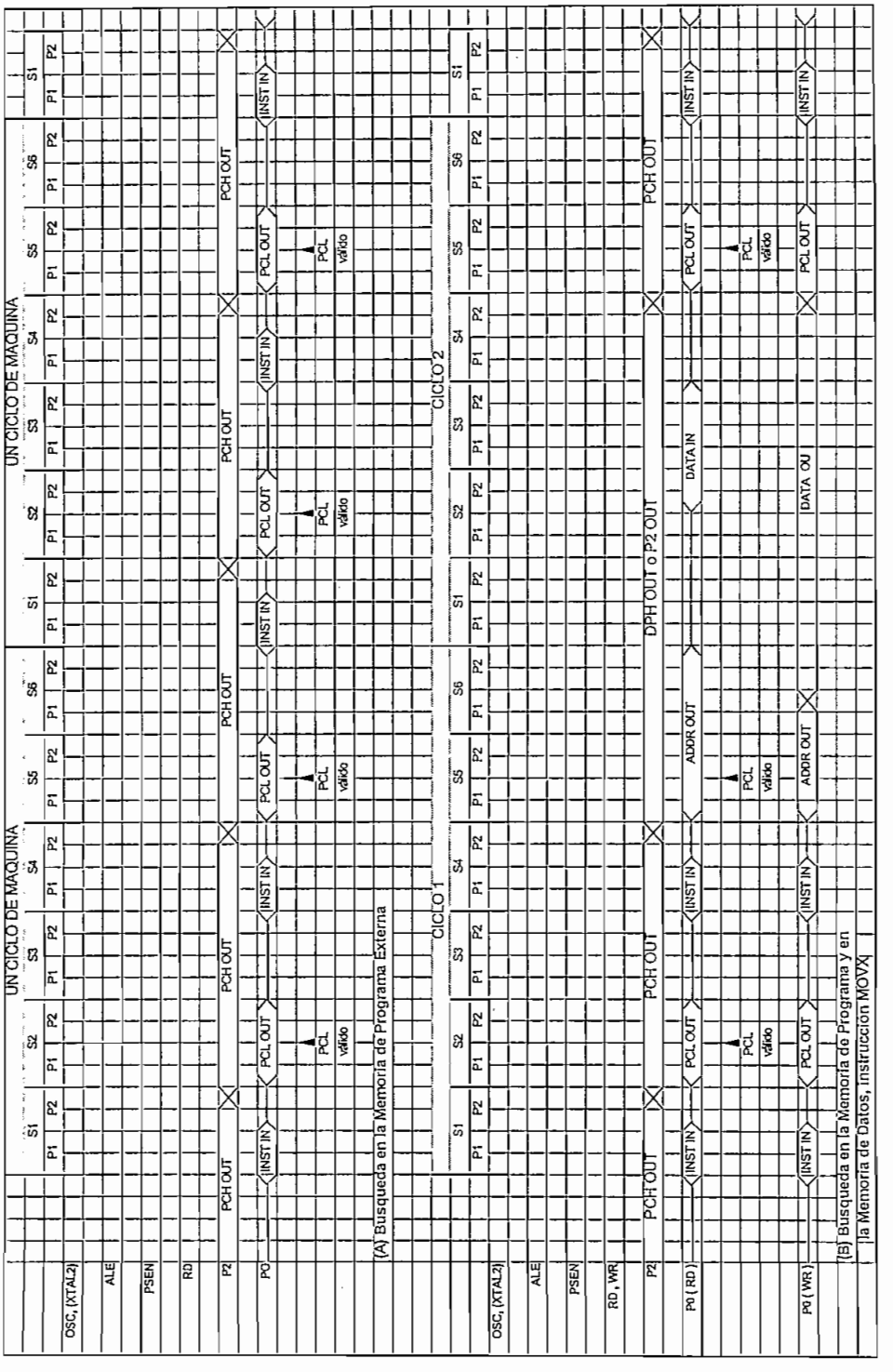


Fig.2.18 Secuencia de acceso a Memoria Externa de Programa

ALE

La función principal de ALE es permitir una temporización adecuada para el latch externo del byte bajo de direcciones del pórtico P0 durante el acceso a la Memoria Externa de Programa. Por este propósito ALE es activada dos veces en cada ciclo de máquina, esta doble activación toma lugar a un cuando involucre una traída que se ignorará.

La única vez que ALE no se habilita es durante el acceso a la memoria externa de datos. El primer cambio de estado de ALE durante el segundo ciclo e máquina de una instrucción MOVX es anulado (ver Fig.2.18). Consecuentemente en ciertos sistemas que no utilizan memoria externa de datos, la señal de ALE es activada en un taza de 1/6 de la frecuencia de oscilación, y puede ser usada como señal de reloj externa o para propósitos de sincronismo.

2.3.2.2 Dispositivos Externos

Los elementos utilizados como periféricos para el desarrollo de este programa son los siguientes :

Memorias EPROM, Memorias RAM, LATCH´s, BUFFER´s , DECODER y DIP SWITCH´s.

Memorias Eprom

Intel 2708 de (1K x 8)

Intel 2716 de (2K x 8)

Intel 2732A de (4K x 8)

Memorias Ram

Intel 6108 de (1K x 8)

Intel 6116 de (2K x 8)

Intel 6132 de (4K x 8)

Buffer´s

74LS541

Latch´s

74LS377

74LS373

Decodificador

74LS138

CAPITULO III

BASES PARA LA ELABORACION DEL PROGRAMA

3.1 CONSIDERACIONES Y LIMITACIONES

Si tomamos en cuenta la variedad de microcontroladores y elementos periféricos que en el mercado existen, y considerando que uno de los principales objetivos de esta tesis es mostrar al usuario la interacción entre ellos, es necesario definir el tipo de microcontrolador que mayores beneficios nos preste, y asi mismo definir los periféricos que comunmente se utilizan.

El microcontrolador que se presta para estos objetivos, es el 8031 de Intel. Este microcontrolador al disponer internamente solo de RAM, obliga a implementar en el diagrama como elemento periférico principal una memoria de programa externa EPROM, en la cual se cargará el programa que el usuario desee. Los dispositivos de memoria que el programa utilizará como elementos periféricos serán las : EPROM (1K, 2K y 4K), y RAM (1K, 2K y 4K).

Los periféricos que se utilizan para representar la RAM , muestran la generalidad de los diferentes dispositivos que pueden ser considerados como tal.

Esto quiere decir que elementos periféricos como teclados, display, lectores de tarjetas, Buffer, Latch, aplicaciones de control, etc., y no solamente la Ram como dispositivo de almacenamiento de datos, son reconocidos por el microcontrolador como RAM.

El puerto P0 trabajará en sus dos funciones alternas (puerto de direcciones y puerto de E/S de datos) razón por la cual requiere de la presencia del LATCH 74LS373 externo para poder interactuar con los periféricos ; esto no ocurre con el puerto P2, pues será utilizado únicamente como puerto de direcciones. Con lo que respecta al puerto P3, este trabajará en sus funciones alternas descritas en el capítulo anterior, mientras que P1 trabajará como E/S de datos, con la característica de que los 4 bits inferiores son de entrada y los 4 superiores de salida, esto se lo hace con el único objetivo de dar mayor versatilidad al usuario en el manejo de instrucciones a nivel de bits, claro está, considerando las limitaciones que esto implica en el código de programa que podrá manejar.

La presencia del LATCH 74LS377 y el BUFFER 74LS541 con sus respectivos INTERRUPTORES y LEDS, deben ser considerados como una generalidad de lo que el microcontrolador puede manejar como RAM, y no como elementos aislados con los que el microcontrolador puede interactuar ; dado que el

direccionamiento y las líneas de control del puerto P3, trabajan indistintamente de aquello.

Se ha incluido un DECODIFICADOR con el objetivo de mostrar que a través de los bits de dirección, en este caso los bits más significativos del puerto P2 del microcontrolador, se puede lograr un direccionamiento más completo y controlado de la memoria externa , esto quiere decir que cuando el microcontrolador este direccionando un determinado dispositivo, se tenga la certeza de que únicamente ese dispositivo es el direccionado.

Para una mejor apreciación de la interacción del microcontrolador con los periféricos, se han definido escalas ampliadas en base a los diagramas de tiempo real que el microcontrolador utiliza para ejecutar una determinada instrucción. Estas escalas de tiempo, el usuario será capaz de controlar, permitiendo de este modo observar y analizar la ejecución de una instrucción más detenidamente.

Analizando la variedad de posibilidades de configuraciones que podemos obtener con estos dispositivos, se ha predefinido un diagrama básico, de tal forma que el usuario pueda seleccionar cual de ellos quiere. El diagrama que se presenta ha sido concebido tratando de mostrar la mayor cantidad de

información posible en lo que se refiere a los eventos de interacción y al mismo tiempo aprovechar el espacio en pantalla. Dentro de cada uno, será factible también variar los dispositivos de memoria, incrementando la capacidad claro esta, en el orden que hemos considerado.

El programa permitirá mostrar a través de ventanas el estado en que los diferentes registros internos del microcontrolador se encuentran antes y después de la ejecución de una instrucción, igual detalle se podrá obtener de las dispositivos de RAM Y EPROM .

Para una adecuada demostración se incluirán programas de prueba , además de un editor de instrucciones, el mismo que le servirá al usuario par editar sus programas o editar una sola instrucción y ver como el microcontrolador la ejecuta.

El programa no debe ser considerado como un simulador o un depurador, ya que el objetivo no es ese, sino más bien ser considerado como una ayuda gráfica para concebir de una manera más objetiva la interacción del microcontrolador con ciertos periféricos. Por ello no es posible el uso de ayudas como el DB y DW además de que el uso de etiquetas es limitado.

3. 2 DIAGRAMAS DE CONFIGURACIÓN

Para la visualización de la interacción entre el microcontrolador y sus periféricos, el usuario debe ser capaz de definir diferentes diagramas de configuración a partir de un diagrama básico que se le presentará como preestablecido al inicio del programa. Este diagrama esta compuesto de los siguientes elementos :

- 1 Microcontrolador 8031
- 1 Decoder
- 1 EPROM de 2K
- 1 RAM de 2K
- 1 Buffer
- 2 Latch
- Dispositivos de Entrada de datos (Interruptores)
- Dispositivos de Salida de datos (Leds)

El detalle de esta configuración se lo puede observar en la Fig.3.1. Dependiendo de los dispositivos de EPROM y RAM que se seleccione de entre los valores establecidos anteriormente que son 1K, 2K o 4 K ; el diagrama presenta 9 combinaciones o configuraciones físicas diferentes.

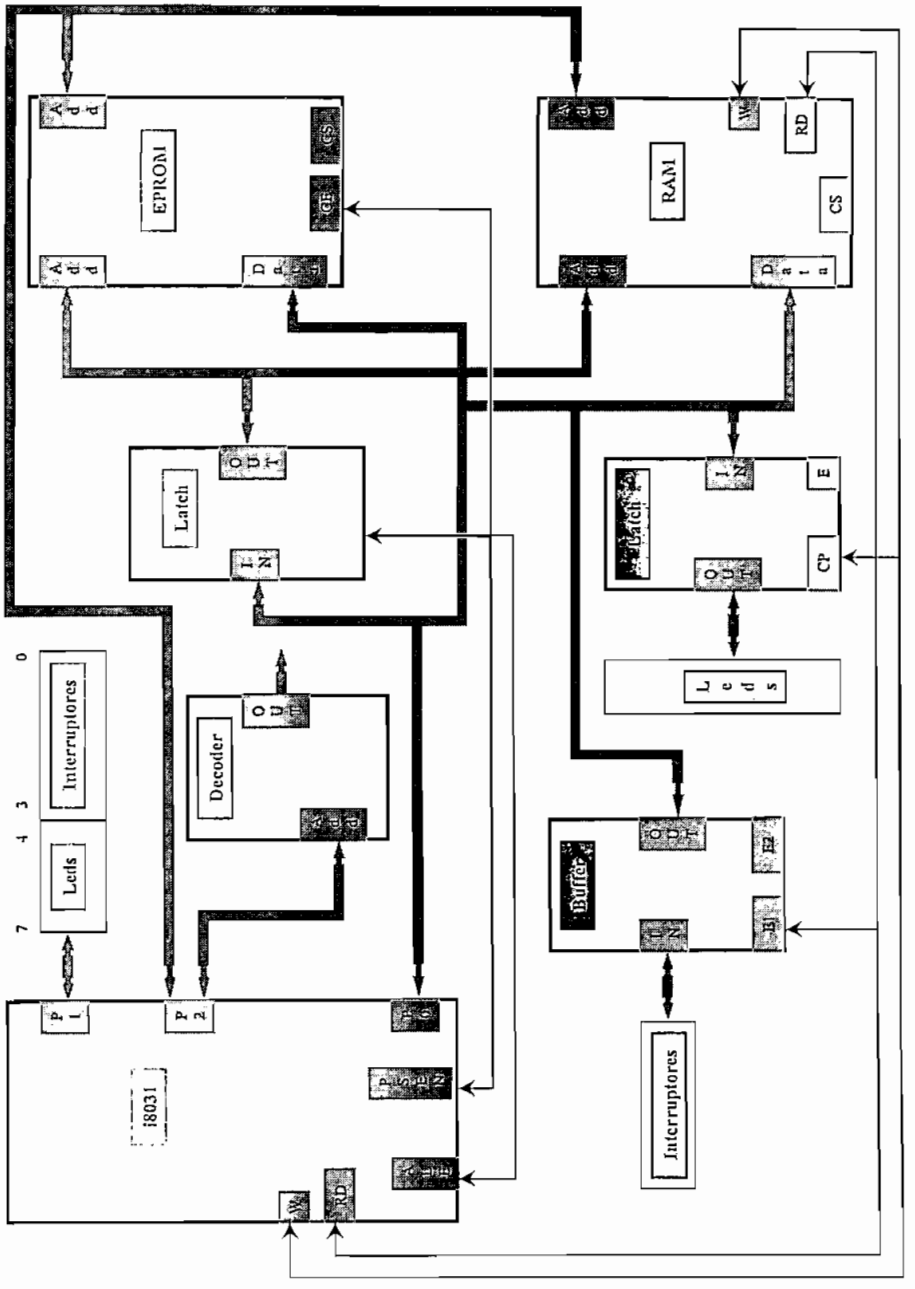


Fig.3.1 Diagrama Básico de Configuración.

Además si se considera que el DECODIFICADOR 74LS138 (3 a 8) que se está utilizando es un dispositivo con 7 líneas de salida disponibles (Y1...Y7) para direccionar en este caso a tres dispositivos como "RAM externa", ya que la primera línea (Y0) está reservada para direccionar a la EPROM ; lo cual nos abre la posibilidad de disponer diferentes configuraciones lógicas de direccionamiento.

La selección de una determinada configuración, influye tanto en la parte física como lógica de los dispositivos seleccionados, por ello el código que se utilice deberá tomar en cuenta las limitaciones que cada configuración establece.

3.3 INTERACCIÓN DEL MICROCONTROLADOR CON PERIFERICOS EXTERNOS

3. 3.1 Interacción Microcontrolador - ROM

El microcontrolador 8031, puede direccionar hasta 64K bytes de ROM externa, pero por consideraciones establecidas anteriormente, la cantidad de memoria máxima de programas que podemos direccionar es 4K, pues es la memoria de mayor capacidad que se puede escoger de las tres posibilidades

permitidas en el menú. Al ser la memoria completamente externa, el pin $\overline{EA}$ del microcontrolador debe obligatoriamente estar puesto a 0_L , para que la búsqueda de direcciones del programa se dirija en todo momento hacia ella (0000H a 0FFFH).

Las líneas de control que se van a utilizar para que el microcontrolador maneje la EPROM como memoria de programa son el $\overline{\text{PSEN}}$ y la Línea Y0 del DECODIFICADOR, cuyo estado en 0L es dado por el estado lógico de las líneas P2.5, P2.6 y P2.7 que emite el microcontrolador al momento de direccionar. Los 8 bits superiores de direccionamiento de la EPROM son dados por la salida del puerto P2, mientras que los 8 bits bajos de direccionamiento son los 8 bits que emite P0 , pero con la diferencia que entre el microcontrolador y la EPROM existe un LATCH que se controla por medio de la señal de ALE que emite el microcontrolador, este LATCH funciona como un puente para permitir únicamente el paso de direcciones de P0 hacia EPROM y no otro tipo de datos como bytes leídos tanto de la EPROM como de la RAM externa que ingresa por P0 y que podrían causar falsos direccionamientos en momentos no requeridos. El estado de cada una de estas líneas tanto de direcciones como de control se muestran en la Fig.3.2

La EPROM para su control dispone de dos entradas, $\overline{\text{CE}}$ y $\overline{\text{OE}}$ que son de lógica negativa, estas dos entradas están conectadas respectivamente con la

Salida 0 del DECODIFICADOR y la Salida $\overline{PSEN}$ del microcontrolador que también son de lógica negativa, lo cual evita utilizar dispositivos adicionales para invertir las señales.

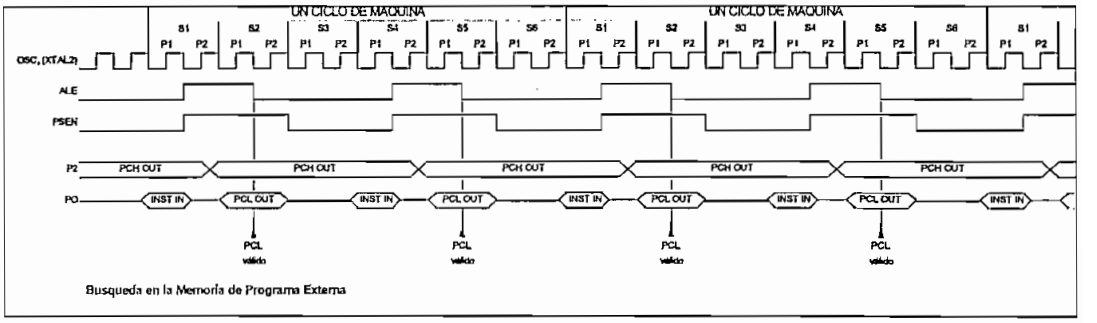


Fig.3.2 Secuencia de Lectura de una instrucción en la memoria de Programa

3.3.2 Interacción Microcontrolador - RAM Externa

La RAM externa en la ejecución de un programa no es solamente un dispositivo, sino que esta constituída por varios elementos que pueden ser manejados como tal, estos son un dispositivo de RAM propiamente dicho, un LATCH para controlar 8 Leds y un BUFFER para ingresar la información de 8 Interruptores. Al igual que en el caso de la memoria de programa, la totalidad de memoria de este tipo que podemos direccionar es de 64K.

El dispositivo de RAM estará controlado por las señales de Lectura $\overline{RD}$ y Escritura $\overline{WR}$ emitidas por el microcontrolador a través de los pines P3.7 y P3.6 respectivamente, y por una de las 7 salidas del DECODIFICADOR, este último control depende de la decodificación que el usuario haya seleccionado.

El BUFFER también dispone de dos entradas de control $\overline{E1}$ y $\overline{E2}$, la entrada $\overline{E1}$ está conectada a la señal de Lectura $\overline{RD}$ del microcontrolador y $\overline{E2}$ está conectada a otra de las 7 salidas (Y1...Y7) del DECODIFICADOR . Se debe recordar que estas entradas $\overline{E1}$ y $\overline{E2}$, están conectadas a una compuerta AND interna en el BUFFER.

El LATCH al igual que el Buffer, dispone también de dos entradas de control, CP y $\overline{E}$; CP está conectada con la señal de Escritura $\overline{WR}$ del microcontrolador, mientras que $\overline{E}$ está conectada con otra de las 7 salidas del DECODIFICADOR.

Es importante recordar y tener presente, que cada salida del DECODIFICADOR en este programa controlará a un elemento de la RAM Externa; como únicamente estamos utilizando los 3 bits superiores del pueto P2 como entradas del DECODIFICADOR, el estado del bit P2.4 es una

condición "no importa" en el direccionamiento a realizarce, pues este pin no tiene conexión alguna.

Cada salida del DECODIFICADOR, manejará un rango de memoria de 8K, por esta razón , dependiendo de la capacidad del dispositivo de RAM, únicamente ocuparán las direcciones inferiores de los 8K mencionados, esto trae como resultado que cada byte del dispositivo de RAM tendrá una única dirección, en cambio para el caso del BUFFER y del LATCH, como cada elemento depende solo de una línea de dirección del DECODIFICADOR, y cada uno de estos elementos representan únicamente un byte de memoria RAM, este byte podrá ser direccionado en cualquier dirección de las 8K disponibles, esto dependerá de la salida del DECODIFICADOR asignado al dispositivo.

Para una mejor explicación de este tema, asumamos que las salidas del DECODIFICADOR que se utilizan para controlar la RAM es la **Salida Y1**, para el BUFFER la **Salida Y2** y para el LATCH la **Salida Y3**, además consideremos que la RAM es de 2K, que es como esta predefinido en nuestro programa, de acuerdo a ello el rango de direcciones para cada elemento será como se muestra en la Tabla 3.1:

	P2.7	P2.6	P2.5	DECIMAL	HEXADEC.
Ram	0	0	1	De 8192 a 16383	De 2000 a 3FFF
BUFFER	0	1	0	De 16384 a 24575	De 4000 a 5FFF
LATCH	0	1	1	De 24576 a 32767	De 6000 a 7FFF

Tabla 3.1 Decodificación de RAM Externa

Como podemos ver claramente el rango de direcciones para cada elemento es de 8K, además para el direccionamiento de la RAM, recordemos que estamos utilizando como máximo 12 bits de direcciones compuestos de P0.0 a P0.7 y de P2.0 a P2.3.

Para este ejemplo, el dispositivo de Ram dependerá en su direccionamiento de los 11 bits inferiores de los mencionados, con ello si tomamos una dirección cualquiera del dispositivo de Ram, por ejemplo la dirección 2555H y recordamos que la decodificación para el dispositivo de Ram, es la **Salida Y1** del DECODIFICADOR y que el mismo elemento no va a depender del estado de las pines P2.4 y P2.3, podemos observar de acuerdo a la siguiente tabla que esta misma dirección 2555H es válida en cuatro posibles direccionamientos, como se puede observar en la Tabla 3.2.

P27	P26	P25	P24	P23	P22	P21	P20	P07	P06	P05	P04	P03	P02	P01	P00	ADDR
0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	2555H
0	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	2D55H
0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	3555H
0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	1	3D55H

Tabla 3.2 Validación de Direcciones

De lo anteriormente expuesto y de la tabla, podemos concluir que el rango de 8K disponible para el dispositivo de RAM, se dividirán en 4 grupos de 2K, con ello cada byte la RAM podrá ser direccionado de 4 maneras distintas.

Con este ejemplo, lo que queremos ratificar, es que dependiendo de la capacidad del dispositivo de RAM que se elija, si es de 1K, cada byte podrá ser direccionado de 8 formas distintas, si es de 2K, tendrá 4 direcciones y si es de 4K tendrá 2 direcciones ; cosa semejante sucede con las direcciones del BUFFER y del LATCH, lo que si debemos tener presente es que estas localidades de RAM tienen 1 byte, por ello se cumple que cada byte tendrá 8K de direcciones válidas.

De las 4 instrucciones que permiten leer y escribir datos en la RAM externa que son :

escritura :

MOVX @Ri , A

MOVX @DPTR , A

Lectura :

MOVX A, @Ri

MOVX A, @DPTR

Aquellas que tienen como uno de sus operandos @ Ri, las direcciones contenidas en estos bancos de registros **de acuerdo a lo que se especifica en el manual del MCS-51**, no son válidas para ningún byte de memoria RAM externa, ya que como se había explicado anteriormente, el primer byte de RAM que se puede direccionar por consideraciones en el direccionamiento a la Memoria Externa es a partir de la dirección 2000H, puesto que el máximo valor que pueden contener estos registros es FFH, por ello cuando se usa como operando @ Ri el programa de visualización mostrará un mensaje de error.

Pero de acuerdo a **información más reciente, se tiene conocimiento de que si se puede direccionar a localidades de RAM externa superiores a FFH a través del contenido de estos registros (@ Ri) si se utiliza como ayuda el pórtico P2.** Esta afirmación se sustenta en el hecho de que cuando se direcciona a localidades de memoria superiores a FFH es necesario utilizar 2

bytes, de los cuales el menos significativo es emitido a través de P0 y el más significativo a través de P2.

En la Fig.3.3 se muestra el estado de las líneas de control para el acceso a la memoria de Datos.

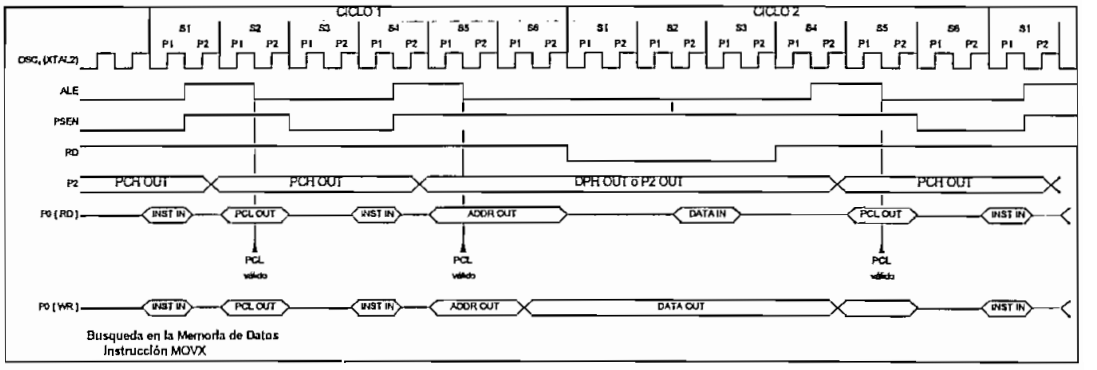


Fig.3.3 Secuencia de Lectura y Escritura en la memoria de Datos

Por todo lo anteriormente expuesto, el programa permitirá visualizar el direccionamiento a RAM externa solamente a través del registro DPTR ya que este registro es de 16 bits.

Además debemos tomar en cuenta el orden de los operandos (DPTR y A) también pueden dar mensajes de error por ejemplo, supongamos que el DPTR tiene la dirección 2000F que corresponde al buffer y que la instrucción a visualizar es MOVX @ DPTR , A, esta instrucción como vemos es de escritura en la Ram y no de lectura que es lo que podemos realizar de un buffer,

por ello este programa tambien le dará un mensaje de error. Si el contenido del DPTR tampoco corresponde a una dirección válida de RAM de la actual decodificación que se haga también tendremos un mensaje de error.

3.3.3 Interacción del Puerto P1 del Microcontrolador con Elementos Externos

El puerto P1 a través de sus 8 bits interactúa con 2 tipos de elementos externos, Leds e interruptores, la interacción con los Leds se da con los 4 bits superiores (P1.4 a P1.7) para salida de datos y con los 4 bits inferiores (P1.0 a P1.3) a interruptores para entrada de datos.

La lectura de los interruptores se da en el estado S6P2 del mismo ciclo de máquina de las instrucciones que tienen como Operando a P1 por ejemplo **MOV direct,P1** o **MOV bit,P1.n**, mientras que la escritura en los pines del microcontrolador se da en los estados del primer ciclo de máquina de la siguiente instrucción que se ejecutará.

La escritura en los pines del puerto toma lugar en dos instancias, primero en el estado S6P2 únicamente los datos de trasladan a los latch de cada pin del microcontrolador, luego en S1P1 del ciclo de máquina de la siguiente instrucción el nuevo estado de los Latch se refleja en los pines del puerto.

Los procesos de Lectura, Escritura y Lectura-Modificación-Escritura a través del Puerto P1 deben ser diferenciados si se lo hace desde el pin o del Latch del pin, ya que dependiendo de donde se lo haga, involucra un conjunto de instrucciones que permiten realizar tal proceso, estos son :

1.- Para escritura en los Latch :

MOV P1 , A

MOV P1 , Rn

MOV P1 , direct

MOV P1 , @Ri

MOV P1 , #data

MOV P1.x , C

POP P1

2.- Para Lectura de los Pines :

```
MOV  A , P1
MOV  Rn , P1
MOV  direct , P1
MOV  @Ri , P1
MOV C , P1.x
ADD A , P1
ADDC A , P1
SUBB A , P1
ANL A , P1
ORL A , P1
XRL A , P1
PUSH  P1
XCH A , P1
ANL C , P1.x
ORL C , P1.x
ANL C , /P1.x
ORL C , /P1.x
JB  P1.x,rel
JNB  P1.x,rel
CJNE  A,P1,rel
```

3.- Para Lectura de los Latch (Lectura - Modificación - Ecritura) :

```
INC P1

DEC P1

ANL P1 , A

ANL P1 , #data

ORL P1 , A

ORL P1 , #data

XRL P1 , A

XRL P1 , #data

CLR P1.x

SETB P1.x

CPL P1.x

JBC P1.x , rel

DJNZ P1 , rel
```

CAPITULO IV

ELABORACION DEL PROGRAMA

4.1 REQUERIMIENTO DE HARDWARE Y SOFTWARE PARA LA ELABORACION

El programa ha sido desarrollado en un ambiente de WINDOWS razón por la cual se ha requerido del siguiente hardware y software mínimo:

1. Microprocesador 486 DX2 o Superior

2. Memoria RAM 16MB como mínimo

3. Monitor SVGA o Superior

4. Sistema Operativo WINDOWS 95

5. Lenguaje de Programación VISUAL BASIC 4.0, como principal herramienta para la creación de los códigos y pantallas del programa

6. Herramienta de manejo de Bases de Datos ACCES, la misma que ha permitido crear una base llamada BaseInsOpe1.mdb, utilizada para editar instrucciones con el programa.

7. Editor de Texto EDIT bajo DOS

4.2 DIAGRAMA DE FLUJO

Para el desarrollo del programa de visualización, se ha utilizado como herramienta un software que se aparta de la programación estructurada y que más bien se acoge a la **Técnica de Programación Orientada a Objetos** ; por esta razón, se mostrará un único diagrama de flujo general (Fig.4.1) del proceso que involucra el ejecutar una instrucción ; y más bien se describirá como está estructurado el Programa bajo esta Técnica.

El siguiente diagrama de flujo muestra en forma general como se llega a la ejecución de la instrucción o las instrucciones, dependiendo de las opciones seleccionadas :

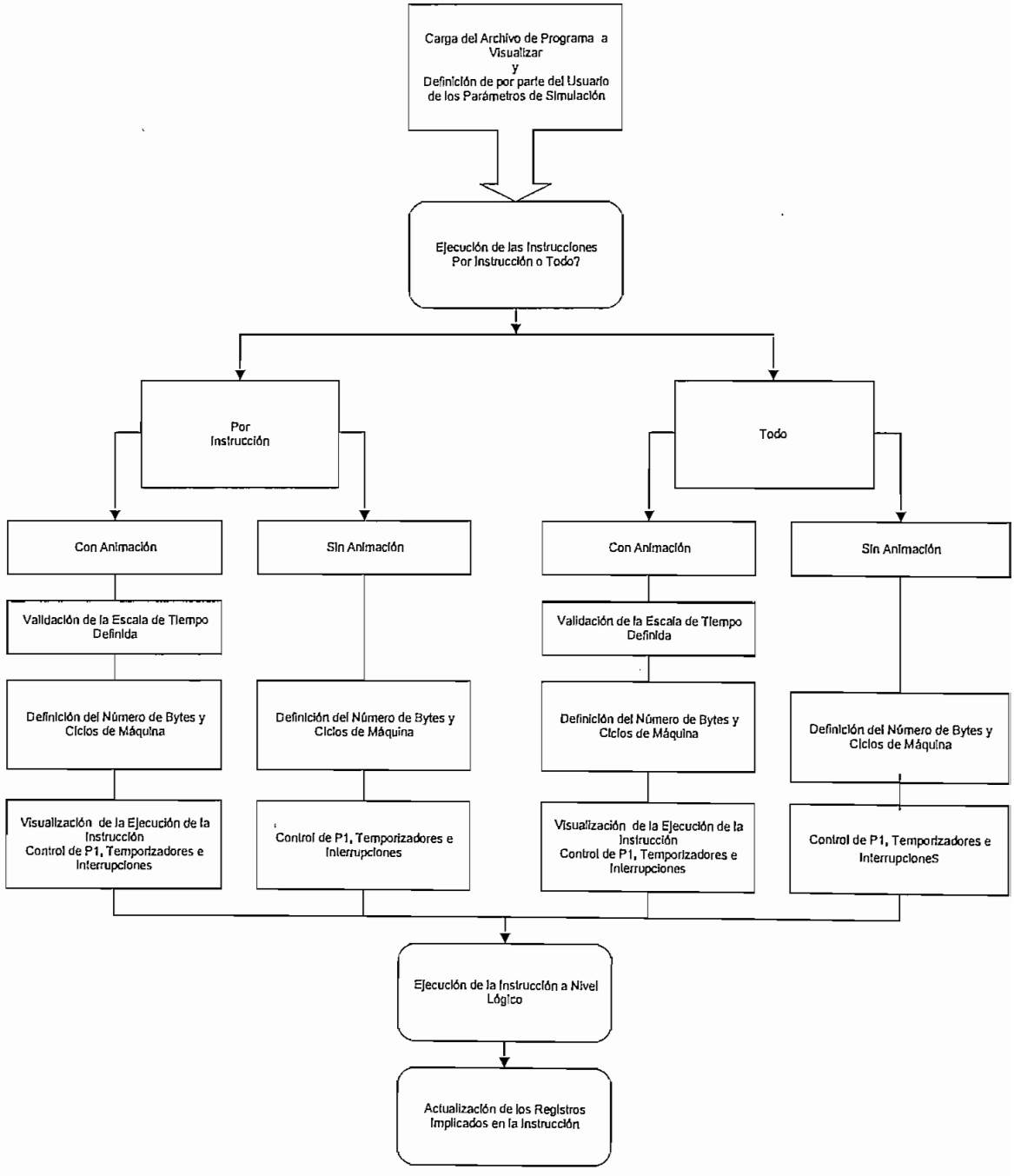


Fig.4.1 Diagrama de Flujo General

4.3 ESTRUCTURA Y DESARROLLO DEL PROGRAMA DE VISUALIZACION

Como ya se indicó anteriormente Visual Basic es el lenguaje de programación bajo el cual se ha desarrollado el código del programa que permite visualizar la interacción entre el microcontrolodor y sus periféricos, dicho código ha sido implementado tomando como base el diagrama de configuración básico propuesto en el capítulo anterior, así como las diferentes consideraciones establecidas respecto de la interacción del microcontrolador con la EPROM, Ram, Buffer, Latch, y demás dispositivos de entrada y salida de datos.

La palabra que representa a la Memoria de Acceso Aleatorio será escrita de manera que permita diferenciar cuando representa a la memoria como dispositivo (**Ram**) y cuando hace referencia a la memoria de datos en general (**RAM**), como el mismo dispositivo de Ram, buffer, latch etc.

La clasificación y explicación del programa se hace en base a la manera en que Visual Basic permite crear los diferentes objetos, asi como la estructura manejada en el programa de acuerdo a la subdivisión que se realiza de subrutinas, temporizadores, interrupciones, etc.

4.3.1 Descripción General

Las **Formas** mas representativas utilizadas en el desarrollo del programa se muestran a continuación en la Tabla 4.1

NOMBRE DE LA FORMA	DESCRIPCION DE SU FUNCION
Form2	Contiene el área de código principal, para la ejecución de cada una de las instrucciones
frm2KRam	Se lo utiliza para visualizar el circuito esquemático de la Ram seleccionada en el diagrama en ese momento
frm2KRom	Se lo utiliza para visualizar el circuito esquemático de la Eprom seleccionada en el diagrama en ese momento
frmBuffer244	Se lo utiliza para visualizar el circuito esquemático del Buffer presente en el diagrama
frmCodigo	Permite al usuario ver el código de su programa, según se de el avance en la ejecución de las instrucciones
frmDecodificador138	Se lo utiliza para visualizar el circuito esquemático del Decodificador presente en el diagrama
frmDiagrama1	Contiene el diagrama para la visualización del puerto serial
frmDiagrama2	Contiene el diagrama principal de la distribución de cada uno de los elementos, en el cual el usuario podrá hacer los cambios del mismo de acuerdo a sus necesidades
frmDirec_Ram	Es un formulario usado para desplegar una pantalla en la cual el usuario designará los pines del decodificador corrrespondiente a cada periférico de memoria RAM externa indicandole además el área de direcciones que le corresponde
frmEdit_Ins	Este formulario se lo agrega al programa con la finalidad de facilitar al usuario la edición de programas de prueba, dandole al programa desarrollado menor dependencia de otras herramientas, como el **Edit** del DOS
frmImagenes	Es simplemente un formulario que actuará como contenedor de las diferentes imagenes usadas en el programa
frmLatch	Sirve para visualizar el circuito esquemático del Latch 377 presente en el diagrama
frmLatch373	Sirve para visualizar el circuito esquemático del Latch 373 presente en el diagrama
frmMemoriaROM	Es un formulario en el cual se le indica al usuario el contenido de la memoria EPROM de acuerdo al programa por él cargado
frmMemorias	Indica el contenido de la memoria Ram Interna
frmMicro	Se lo utiliza para visualizar el circuito esquemático del Microcontrolador i8031 utilizado en el diagrama
frmSwitcheo_P1	Formulario para ingresar el dato en forma decimal, hexadecimal o binario en los interruptores del frmDiagrama2
MDIForm1	Hace la función de formulario MDI (Principal) bajo la definición de Windows
Módulo1	Contine todas la variables y subrutinas globales usadas en diferentes formularios.

Tabla 4.1 Principales Formas de Programa

Los nombres que se utilizan para las Formas y Funciones del programa estan escritas bajo las recomendaciones que hace Visual Basic para manejar de manera más sencilla las diferentes áreas de código. De las diferentes Formas mencionadas, la mayoría de ellas serán vistas por el usuario en ciertas fases de uso del programa.

A continuación, y considerando el diagrama de flujo general presentado al inicio del capítulo, se describirá en forma general la manera en que el programa desarrollado hace uso de cada una de las Formas antes mencionadas.

Para hacer referencia al programa desarrollado, se lo hará utilizando las siglas **V.I.M.P.**

a) Carga de Programa a Visualizar

Cuando el programa V.I.M.P. es abierto por el usuario, la primera área de codigó ejecutada es Form_Load del formulario Form2, la misma que permite inicializar todas las variables usadas, tanto las simples como las matriciales. Además se inicializa la RAM interna del microcontrolador de acuerdo a las especificaciones dadas cuando se realiza un reset del microcontrolador, como por ejemplo poner todos los bits del puerto P1 a 1_L.

El area de código correspondiente a la carga del programa del usuario para visiualizar su funcionamiento se llama cmdGenerarMatriz_Click ; aquí en primera instancia se

utiliza uno de los menues de Windows para abrir programas, luego el archivo cargado es analizado línea por línea, verificando que estas sean líneas de código válidas dividiendolas en etiquetas, opcode de la operación, y operandos ; cada uno de estos datos se almacenan en filas y columnas de una matriz llamada sInstruccionesOperandos, matriz que luego se utiliza para ir ejecutando cada una de las instrucciones, con esto, el archivo de programa del usuario no permanece abierto todo el tiempo. Esto permite hasta cierto modo ayudar al usuario a verificar la correcta sintaxis usada en el archivo cargado; de encontrarse errores, el usuario pueda abrir su código en una sesión de edición bajo DOS y correjirlos para nuevamente cargarlo con las correcciones necesarias.

A continuación se procede a cambiar ciertas etiquetas usadas para dejar un único formato de código y no estarlo averiguando durante la ejecución de las instrucciones, esto se hace en base a al contenido de las siguientes variables y constantes matriciales : sEQU_diDA (contiene los equivalentes de las etiquetas) y sSFR_Operandos (etiquetas definidas para el 8031 como TR0, TF1, C, P1, P3, etc.)

En la misma matriz sInstruccionesOperandos se guarda en las siguientes columnas los códigos de las instrucciones pero en forma binaria y hexadecimal, que luego serán usados para indicarlos en el contenido de la memoria de programa externa ; también se guarda en la misma matriz otros valores como el número de bytes y el

número de ciclos de máquina de cada intrucción presente en el archivo, esta información sirve tanto para la ejecución con animación como para presentarla a manera de información cuando una determinada instrucción se está ejecutando.

b) Ejecución de las Instrucciones del Programa

Una vez que el usuario a abierto el archivo de código que desea revizar, él puede proceder a ejecutar cada una de las instrucciones que conforman el programa ; ya sea de instrucción en instrucción o todo, o en modo animado o no animado.

Cada uno de estos modos de ejecución esta identificado por las siguientes areas de código : cmdEjecutarProgra,a_Click (todo el programa) y cmdStep_Click (de instrucción en instrucción). Estas areas de código contienen algunas subrutinas como Frecuencia_Simulación, la cual determina la escala de tiempo que el usuario haya elegido, pudiendo ser esta de 1, 2, 3 o 4 segundos de duración de cada oscilación de reloj.

Otras subrutinas ya mencionadas y de extrema importancia son la relacionadas a la ejecución misma de las instrucciones tanto a nivel gráfico como lógico, el nombre de estas subrutinas esta conformado así : Instrucción*Opcode*, por ejemplo InstruccionADD es la subrutina de ejecución de instrucción ADD.

La parte gráfica de la instrucción es la primera en realizarse (si el usuario a elegido este modo), debido a que se utiliza los registros actuales para indicar el flujo de datos entre el microcontrolador y sus periféricos. En segundo lugar se realiza la parte lógica , es decir, la actualización de los registros involucrados en la instrucción. Esta segunda parte utiliza los datos en forma decimal o binaria de los operandos involucrados de acuerdo a la operación que tenga que realizarse con ellos. Para el tratamiento de los datos en forma decimal, binaria o hexadecimal se dispone de diferentes subrutinas como : ConversionD_B (cambia los datos de formato decimal a binario si son de 8 bits), ConversionDPTRH_D (convierte los datos de formato hexadecimal a decimal si son de 16 bits), etc.

Para actualizar los datos de Ram interna tanto en forma decimal, binaria y hexadecimal se tiene las subrutinas ActualizacionRAM y ActualizaciónRAMD_BH ; como el DPTR maneja datos de 16 bits en sus dos bytes DPH y DPL se dispone da las subrutinas ActualizacionDPTRB_DH y ActualizacionDPTRD_BH.

Vale la pena indicar que la actualización de los diferentes operandos de una instrucción, sean estos de Ram interna, Ram externa, leds del pórtico P1, leds del Latch , etc; se realizan sin depender del modo que el usuario haya elegido, modos que pueden ser con animación o sin animación.

En el modo con animación, el gráfico que se presenta al usuario mostrará en base al cambio de colores, el estado de los buses y de cada línea de control en el desarrollo de cada ciclo de máquina que este en progreso en la ejecución de una determinada instrucción, para ello el V.I.M.P. hace uso de las subrutinas que manejan el color como son P0_Eprom_color1, P2_Ram_color1, etc.

Este gráfico, como se indica en el capítulo anterior puede ser cambiado de acuerdo a las necesidades del usuario partiendo de un gráfico base como el que se muestra en la Fig. 4.2.

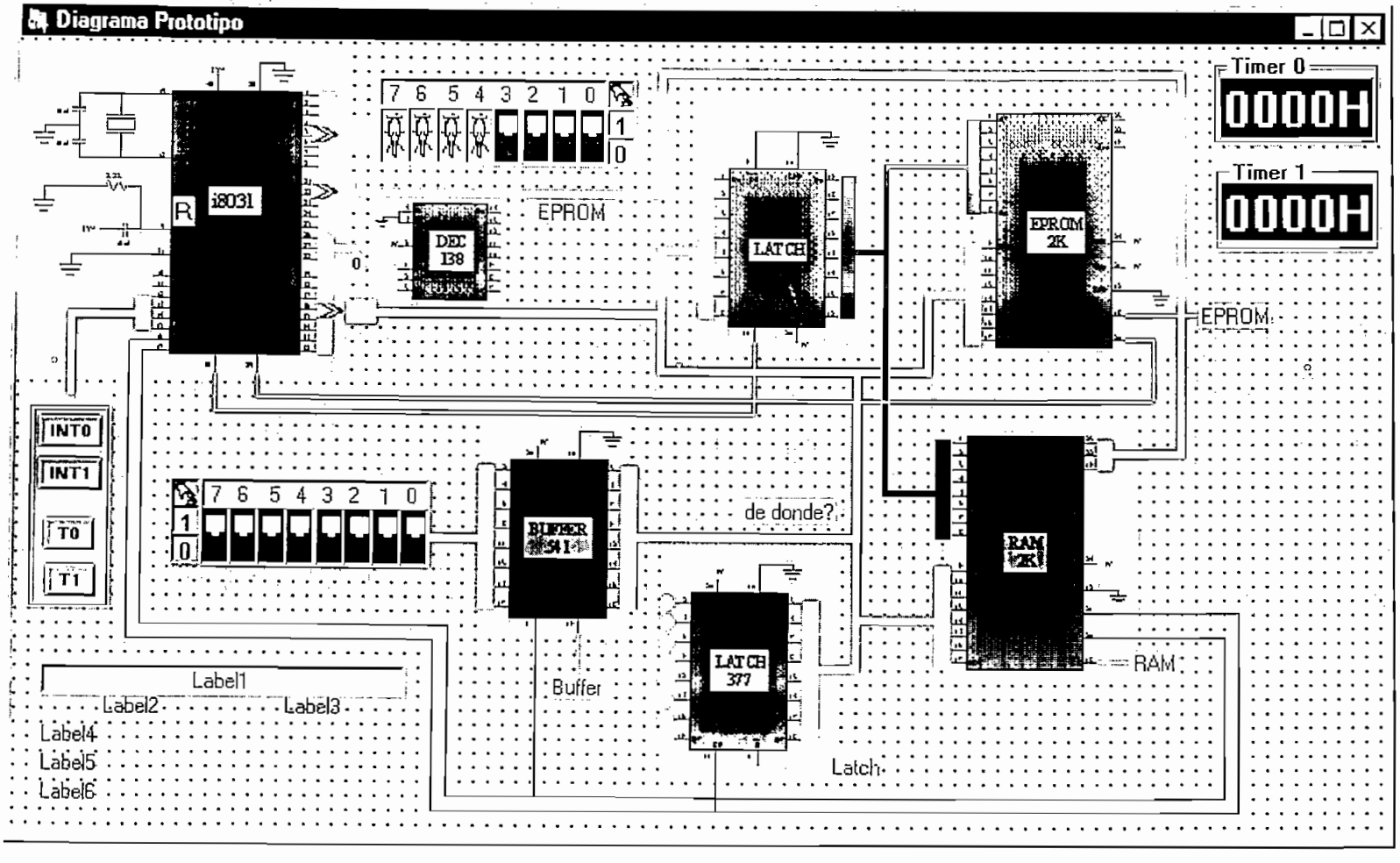


Fig.4.2 Diagrama Básico en Modo de Diseño

c) Descripción del Diagrama Básico en Modo de Diseño

El diagrama básico en modo de diseño está constituído por elementos como el microcontrolador 8031, Ram de 2K, Eprom de 2K, Decoder, 2 Latch, Buffer, Interruptores y Leds como dispositivos de entrada y salida de datos, buses y líneas de control ; junto con estos elementos, existen labels o etiquetas asociadas o independientes de ellos que cumplen una determinada función, y que en el modo de ejecución son o no presentados al usuario.

A continuación se detalla la función que cumple cada uno de los labels o etiquetas que se muestran en el diagrama.

Los siguientes son campos que en el modo de ejecución se los verá únicamente cuando se ejecuta una instrucción

Label1 : Mostrará el código de la instrucción que se está ejecutando

Label2 : Mostrará la información acerca de el número de Bytes de la instrucción.

Label3: Mostrará la información acerca de el número de ciclos de máquina de la instrucción.

Label4 : Mostrará la información acerca de hacia que tipo de memoria está direccionando el microcontrolador.

Label5 : Mostrará la información acerca de lo que el microcontrolador está realizando en un determinado ciclo de máquina de la instrucción, por ejemplo una lectura de la instrucción o una lectura no válida.

Etiqueta **EPROM , RAM, Buffer, Latch** : Estas etiquetas servirán para indicar que el microcontrolador está direccionando al elemento correspondiente de acuerdo a la decodificación de direcciones que se haya realizado.

Etiqueta **de donde ?** o **a donde ?** : Serán mostradas cuando se ha direccionado la Ram de escritura para una operación de lectura, o la Ram de lectura para una operación de escritura respectivamente, y adicionalmente si la dirección contenida en el DPTR no corresponde a una dirección de un elemento de Ram externa.

Timer 0 , Timer 1 : Serán mostrados cuando los bits de arranque TR0 o TR1 del registro TCON de dirección 88H son puestos en 1_L respectivamente, sea que los Timers esten trabajando como temporizadores o contadores.

Los siguientes son campos que en el modo de ejecución se los verá en el arranque del programa :

INT0 , INT1 : Permitiran generar interrupciones externas a través de los pines P3.2 y P3.3 del microcontrolador.

T0 , T1 : Permitirán introducir una señal de reloj externa de frecuencia no fija cuando el Timer trabaje como contador.

d) Control de P1, Timers e Interrupciones

- Para el control de lectura y escritura en los dispositivos de entrada/salida de datos del puerto P1, se dispone de diferentes areas de código, como :

en la Form2 :

Puerto_P1RD (para lectura)

Puerto_P1WR (para escritura)

en el frmDiagrama2 :

EscrituraP1

y cierta parte de código de las interrupciones que permiten controlar la visualización de la ejecución de las instrucciones, tales como :

Byte1_1Ciclo_1

Byte1_2Ciclo_1

Byte1_2Ciclomovc_1

Byte1_2CiclomovxR_1

Byte1_2CiclomovxR_Buffer

En base a la utilización de estas subrutinas se mostrará simplemente el dato hexadecimal que se lee de los 4 interruptores de entrada ubicados en los bits menos significativos como la escritura el los 4 leds ubicados en los bits más significativos. Si la instrucción es a nivel de bits como MOV P1.5,C (en caso de escritura) o MOV C,P1.2 (en caso de lectura) se indicará únicamente el estado del bit que esta involucrado en la operación.

La actualización del los registros del puerto P1 se hará sea que el usuario haya seleccionado el modo con animación o sin animación.

- Para el control de los Timers 0 y 1 del microcontrolador 8031 se utilizan varias subrutinas, las cuales siguen una secuencia que permiten cumplir este objetivo; dicha secuencia se da de la siguiente manera :

Las subrutinas Temp_Cont_0 y Aumento_MIM0_0 controlan el Timer0, y, Temp_Cont_1 y Aumento_MIM0_1 controlan el Timer1.

En primer lugar, estas subrutinas revisan si el Timer arrancó o no verificando el estado de los bits TR0 y TR1.

Luego se averigua si los Timers estan trabajando como temporizadores o contadores, y si el control por software (bit GATE) o hardware (bit del pin T0 o T1 respectivamente), estan habilidatos.

Si las condiciones antes mencionadas se cumple para cualquiera de ellos, entonces se llama a las subrutinas Aumento_MIM0_0 y Aumento_MIM0_1 para hacer el incremento en el contenido de los registros correspondientes a cada timer ; esto lo realiza no sin antes haber establecido en que Modo estan configurados cada temporizador (por ejemplo Modo 0 contador/temporizador de 13 bist).

Luego de haber realizado el incremento, se debe consultar si se ha dado o no el desbordamiento de los Timers ; si el desbordamiento ocurre, se debe cambiar el contenido del PC (contador del programa) para que la siguiente instrucción a ser ejecutada sea la primera de la subrutina de atención a la interrupción generada..

El valor de incremento en los Timers en el programa se da de dos maneras. Si se está ejecutando la instrucción en el modo con animación, el Timer será incrementado cada ciclo de máquina, en cambio si la instrucción se ejecuta en el modo sin animación, el incremento en los Timers será igual al número de ciclos de máquina de

la instrucción, puesto que aquí no se controla cuando comienza y cuando culmina cada ciclo de máquina.

- El control de las interrupciones se hace en base a dos subrutinas Atencion_Interrupciones y Prioridad_Interrupcion .

La subrutina Prioridad_Interrupcion establece el orden en que se atenderá cada interrupcion de acuerdo al estado del registro IP del microcontrolador i8031 y a las definiciones de prioridad que establece la teoría del mismo ; esta verificación es necesaria dado que se puede generar más de una interrupción a la vez

Una vez establecida la prioridad de las interrupciones coresponde verificar cuales de las interrupciones han sido habilitadas tanto individualmente como generalmente consultando el registro IE. Finalmente el salto hacia la subrutina de atención a alguna interrupción se dará solo si la bandera de aviso (IE0,TF0,IE1,TF1) esta activada.

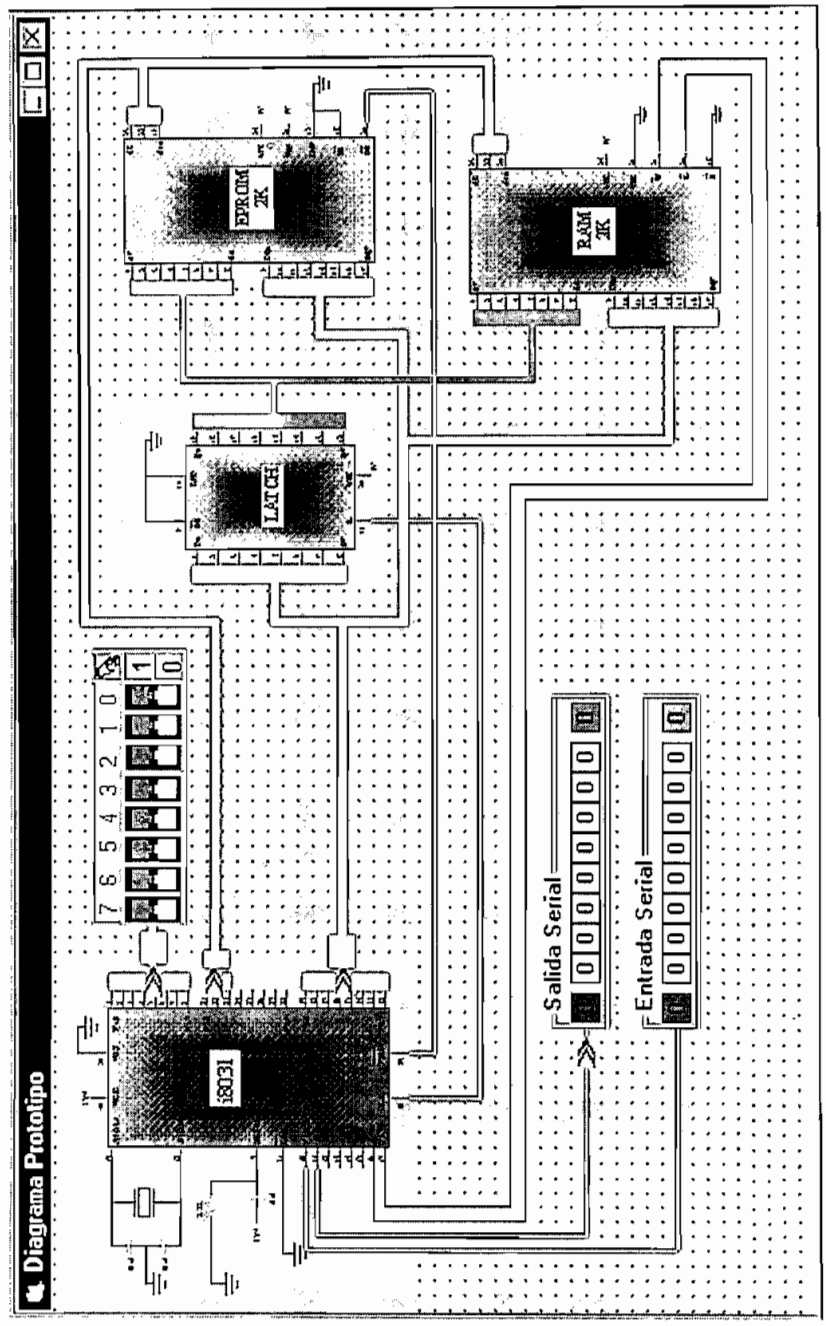


Fig.4.3 Diagrama de Configuración para el Pórtico Serial

La velocidad de transmisión que se utilizará es el resultado de considerar los siguientes valores para los parámetros que la determinan en este Modo :

a) SMOD = 1 fosc = 12MHz TH1 = 254

entonces la Velocidad M1 = 31250 bps

b) SMOD = 0 fosc = 12MHz TH1 = 255

entonces la Velocidad M1 = 31250 bps

En los dos casos se puede observar que para transmitir 31250 bits se requiere un segundo, entonces para transmitir 1 bit se empleará 32 microsegundos.

Si tenemos presente que cada ciclo de máquina con esta frecuencia de oscilación equivale a 1 microsegundo y que en nuestra escala de tiempo la opción 2 hace que un ciclo de máquina tarde aproximadamente 48 segundos, el tiempo que se tardaría en ver la transmisión de un bit sería demasiado grande (aprox. 25minutos).

De las consideraciones antes mencionadas ya se puede ver claramente la razón por la que se le ha dado el tratamiento úmicamente de demostración (Demo) al Pórtico Serial.

CAPITULO V

PRUEBAS DEL PROGRAMA

5.1 PROGRAMAS DE PRUEBA

Los programas de prueba que ha continuación se presentan, permitirán verificar, validar e indicar el funcionamiento del V.I.M.P.

El éxito de estas pruebas se basan en que los **resultados esperados** que serán de antemano conocidos en la definición de los programas de prueba en base a la teoría de los microcontroladores, concuerden con los **resultados obtenidos** al cargarlos en el V.I.M.P.

5.1.1 DEFINICION DE LOS PROGRAMAS DE PRUEBA

Para cumplir el objetivo antes planteado y a la vez mostrar la mayor cantidad de información posible que se pueda obtener del V.I.M.P. se han definido 4 Programas Ejemplo :

5.1.1.1 Programa Ejemplo de Acceso a Memoria Externa.

Objetivos :

Con este programa se pretende visualizar y probar lo siguiente :

- Configurar el Diagrama
- Decodificar la RAM Externa
- Probar ciertas instrucciones lógicas, aritméticas y de movimiento.
- Direccionar y Escribir en el Dispositivo de Ram Externa
- Direccionar y Escribir en el Latch
- Direccionar y Leer del Buffer
- Escribir en una Dirección no válida de RAM
- Direccionar y leer de la ROM

5.1.1.2 Programa Ejemplo de Llamado a Subrutinas y Saltos.

Objetivos :

Con este programa se pretende visualizar y probar lo siguiente :

- Mostrar como se ejecuta el Llamado a una Subrutinas
- Mostrar como se ejecutan los Saltos.
- Utilizar ciertas ayudas en el desarrollo de un programa como son la Etiquetas.
- Mostrar el funcionamiento de ciertas instrucciones de:

- Intercambio
- Movimiento
- Aritméticas
- Rotación
- Lógicas

5.1.1.3 Programa Ejemplo de Pórticos,Temporizadores e Interrupciones.

Objetivos :

Este programa pretende visualizar y probar lo siguiente :

- Utilizar el temporizador para generar una espera de un tiempo determinado.
- Generar una interrupción Externa y dar atención a ella.
- Leer del Pórtico P1.
- Escribir en el Pórtico P1.
- Probar ciertas instrucciones a nivel de Bits

5.1.1.4 Programa Ejemplo del Pórtico Serial.

Objetivos :

El programa ejemplo que ha sido cargado en la demostración del funcionamiento del Pórtico Serial, pretende visualizar y probar lo siguiente:

- Ingresar datos y mostrarlos en los leds conectados al Pórtico P1.

- Recuperar los datos leídos del Pórtico P1 que han sido ingresados a través de los interruptores conectados a él.

5.2 PRUEBAS Y RESULTADOS

5.2.1 Programa Ejemplo de Acceso a Memoria Externa.

a) Código y Resultados Esperados.

CODIGO DEL PROGRAMA	RESULTADOS ESPERADOS
MOV A,#0FH	Número de C.M. = 1 Número de Bytes = 2 1erByte = 74H 2doByte = 0FH El Acumulador deberá contener el valor de 0FH luego de realizar esta operación.
ANL A,#04H	Número de C.M. = 1 Número de Bytes = 2 1erByte = 54H 2doByte = 04H El Acumulador deberá contener el valor de 04H luego de realizar esta operación.
MOV DPRT,#2010H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 90H 2doByte = 20H 3erByte = 10H El registro DPTR deberá contener el valor 2010H que es un valor de dirección válida del dispositivo de Ram Externa.
MOVX @DPTR,A	Número de C.M. = 2 Número de Bytes = 1 1erByte = F0H La dirección 2010H del dispositivo de Ram Externa deberá contener el valor 04H que fue cargado en el Acumulador.
ADD A,#0AH	Número de C.M. = 1 Número de Bytes = 2 1erByte = 24H 2doByte = 0AH El Acumulador deberá presentar el valor de 0EH luego de realizar esta operación.

MOV DPRT,#6040H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 90H 2doByte = 60H 3erByte = 40H El registro DPTR deberá contener este valor de dirección que corresponde al Latch.
MOVX @DPTR,A	Número de C.M. = 2 Número de Bytes = 1 1erByte = F0H El Latch deberá mostrar a través de los leds conectados a el, el valor 0EH, para esto encenderá 3 leds conectados a salidas que representen a estos bits.
MOV DPRT,#4020H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 90H 2doByte = 40H 3erByte = 20H El registro DPTR deberá contener este valor de dirección que corresponde al Buffer.
MOVX A,@DPTR	Número de C.M. = 2 Número de Bytes = 1 1erByte = E0H El Acumulador deberá contener el valor que se ingrese a través de los interuptores conectados al Buffer, para este ejemplo se ingresará el valor 09H, por tanto el registro el Acumulador deberá contener este valor.
MOVX @DPTR,A	Número de C.M. = 2 Número de Bytes = 1 1erByte = F0H El Latch debería mostrar a través de los leds conectados a el, el valor 09H, pero debido a que se cambiará la decodificación de Ram haciendo que esta sea una dirección no válida, el Latch no deberá encender ningún led.
MOV DPTR,#01H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 90H 2doByte = 00H 3erByte = 01H El DPTR deberá contener el valor de 01H que es la dirección de una localidad de la ROM externa.
CLR A	Número de C.M. = 1 Número de Bytes = 1 1erByte = E4H El Acumulador deberá contener el número 00H

MOVC A, @A+DPTR	Número de C.M. = 2 Número de Bytes = 1 1erByte = 93H El Acumulador deberá contener el valor contenido en la dirección 01H de la ROM que en este caso es el segundo byte de la primera instrucción del programa cargado(0FH)
SJMP $	Número de C.M. = 2 Número de Bytes = 2 1erByte = 80H 2doByte = FEH Luego de esta instrucción el microntrolador ejecutará el programa una y otra vez, es decir entrará en un lazo de ejecución infinito.

b) Ejecución y Resultados Obtenidos

La ejecución del programa comienza con el proceso de carga del archivo Ejemplo1.asm como se muestra en la Fig 5.1

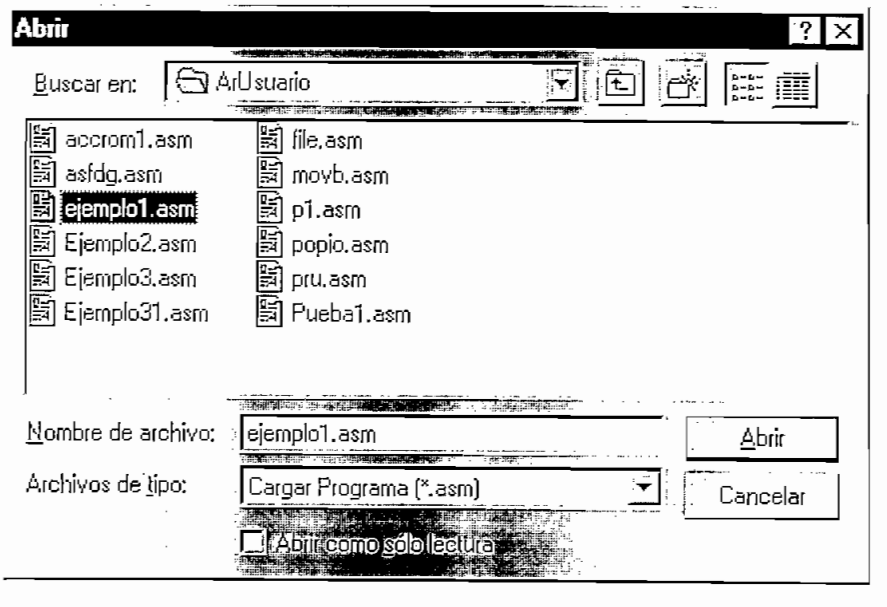


Fig.5.1 Carga del Archivo ejemplo1.asm

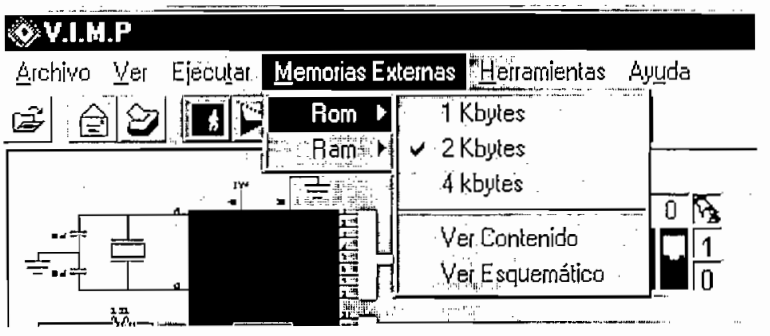


Fig.5.3 Selección de la capacidad de los dispositivos de Memorias Externas

Una vez que se ha configurado el gráfico se debe decodificar la RAM externa (Fig.5.4) tomando en cuenta las necesidades del código y las consideraciones establecidas en un capítulo anterior.

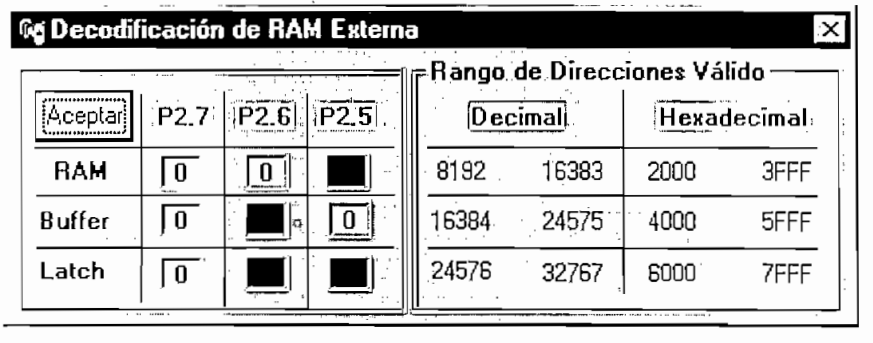


Fig.5.4 Decodificación de RAM Externa.

Ahora se tiene la posibilidad de ejecutar las instrucciones con animación o sin animación, para la segunda opción, se puede seleccionar una escala de tiempo que permite aumentar o disminuir la duración de un ciclo de máquina (Fig.5.5).

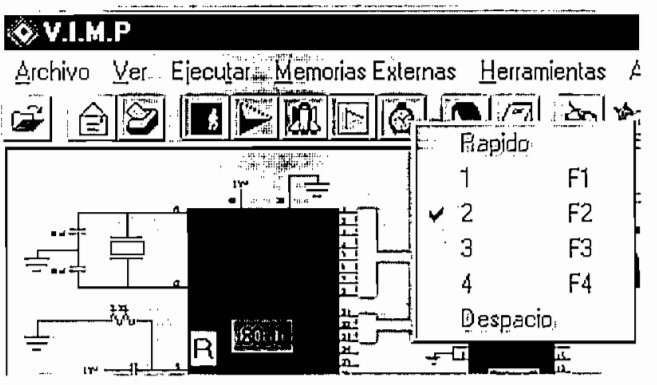


Fig.5.5 Selección de la Escala de Tiempo.

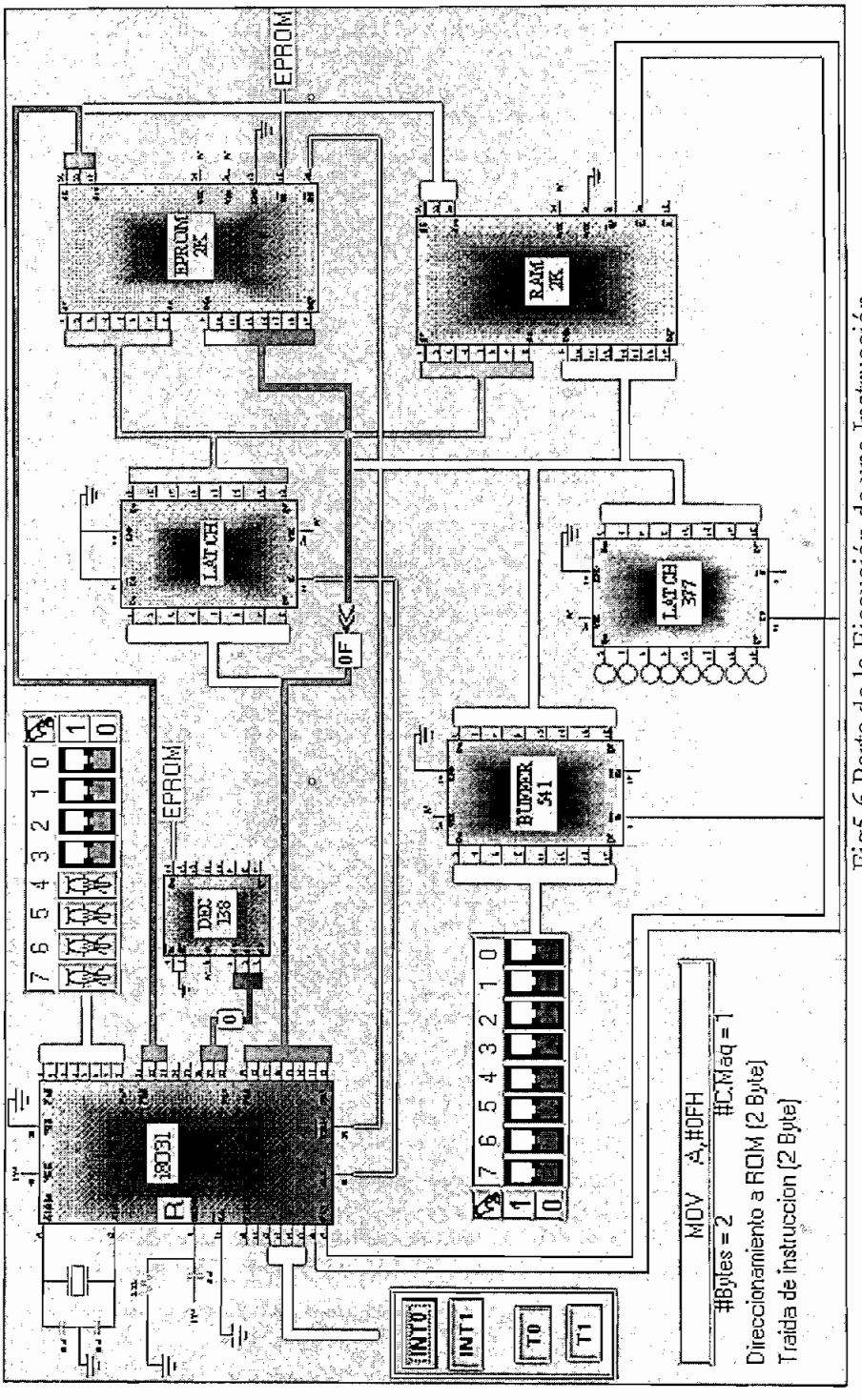
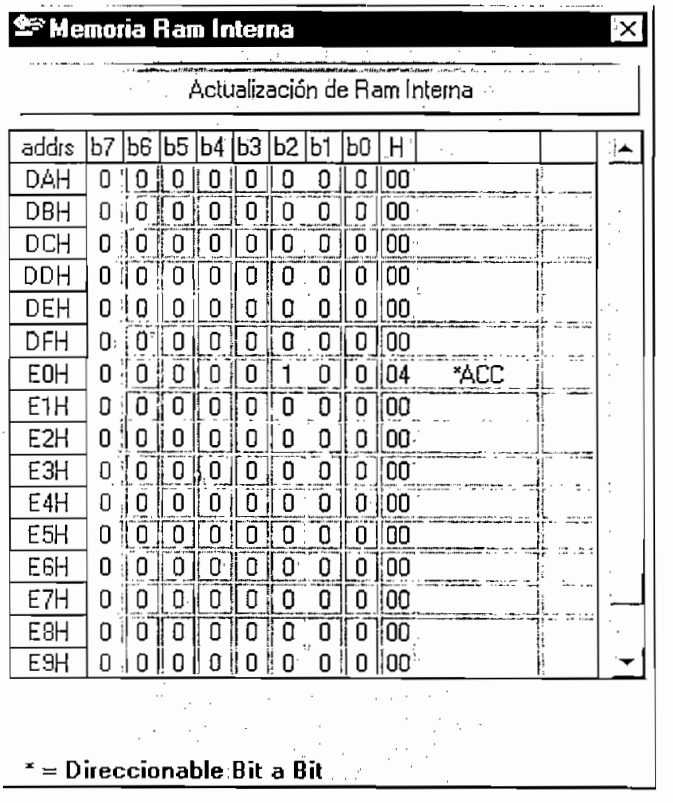


Fig5.6 Parte de la Ejecución de una Instrucción

En la Fig.5.6 se puede observar como el V.I.M.P. ejecuta parte de la primera instrucción del programa. La información del desarrollo de la ejecución de cada instrucción se la puede observar en la parte inferior izquierda de la pantalla a medida que el microcontrolador las ejecuta. Gráficamente se puede observar como el microcontrolador a través del bus de datos está recuperando el segundo byte de la instrucción (0FH).

Como se dijo anteriormente, estos resultados obtenidos también se los puede ir comparando con los resultados esperados especificados en la definición del programa.

A continuación se ejecuta la siguiente instrucción (ANL A,#04H) y revisamos el contenido del acumulador (Fig.5.7), el cual corresponde al resultado esperado.



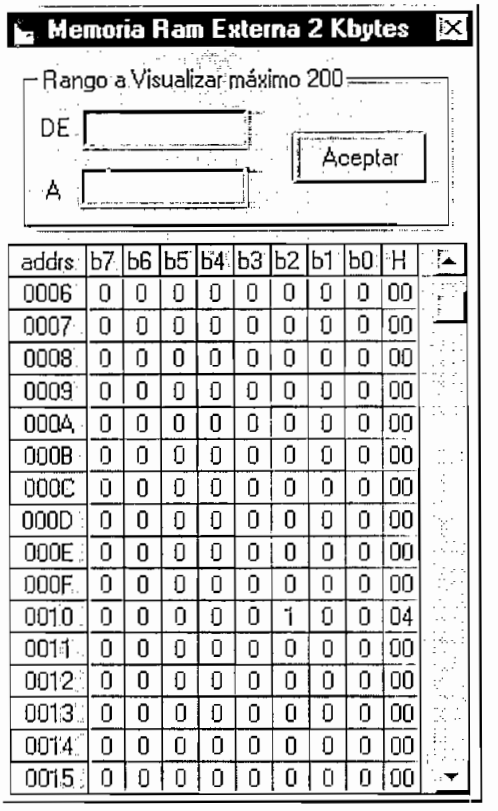
addrs	b7	b6	b5	b4	b3	b2	b1	b0	H	
DAH	0	0	0	0	0	0	0	0	00	
DBH	0	0	0	0	0	0	0	0	00	
DCH	0	0	0	0	0	0	0	0	00	
DDH	0	0	0	0	0	0	0	0	00	
DEH	0	0	0	0	0	0	0	0	00	
DFH	0	0	0	0	0	0	0	0	00	
E0H	0	0	0	0	0	1	0	0	04	*ACC
E1H	0	0	0	0	0	0	0	0	00	
E2H	0	0	0	0	0	0	0	0	00	
E3H	0	0	0	0	0	0	0	0	00	
E4H	0	0	0	0	0	0	0	0	00	
E5H	0	0	0	0	0	0	0	0	00	
E6H	0	0	0	0	0	0	0	0	00	
E7H	0	0	0	0	0	0	0	0	00	
E8H	0	0	0	0	0	0	0	0	00	
E9H	0	0	0	0	0	0	0	0	00	

Fig.5.7 Contenido del Acumulador

Luego se ejecuta las instrucciones que permiten direccionar y escribir en el dispositivo de memoria Ram Externa:

```
MOV     DPTR,#2010H
MOVX    @DPTR,A
```

Al realizar una consulta en el contenido del dispositivo de Ram externa (Fig.5.8) se comprueba que se ha realizado la escritura en la dirección física 010H, que corresponde de acuerdo a la decodificación a la dirección lógica 2010H.



addrs	b7	b6	b5	b4	b3	b2	b1	b0	H
0006	0	0	0	0	0	0	0	0	00
0007	0	0	0	0	0	0	0	0	00
0008	0	0	0	0	0	0	0	0	00
0009	0	0	0	0	0	0	0	0	00
000A	0	0	0	0	0	0	0	0	00
000B	0	0	0	0	0	0	0	0	00
000C	0	0	0	0	0	0	0	0	00
000D	0	0	0	0	0	0	0	0	00
000E	0	0	0	0	0	0	0	0	00
000F	0	0	0	0	0	0	0	0	00
0010	0	0	0	0	0	1	0	0	04
0011	0	0	0	0	0	0	0	0	00
0012	0	0	0	0	0	0	0	0	00
0013	0	0	0	0	0	0	0	0	00
0014	0	0	0	0	0	0	0	0	00
0015	0	0	0	0	0	0	0	0	00

Fig.5.8 Resultado de la Escritura en el dispositivo de Ram Externa

Ejecutar las siguientes 3 instrucciones que permiten cambiar el valor del contenido del acumulador a 0EH, direccionar y escribir en el Latch este mismo valor :

```
ADD     A,#0AH
MOV     DPTR,#6040H
MOVX    @DPTR,A
```

El resultado se puede observar en la Fig.5.9, en donde se puede ver los leds encendidos y el valor mismo que se encuentra retenido.

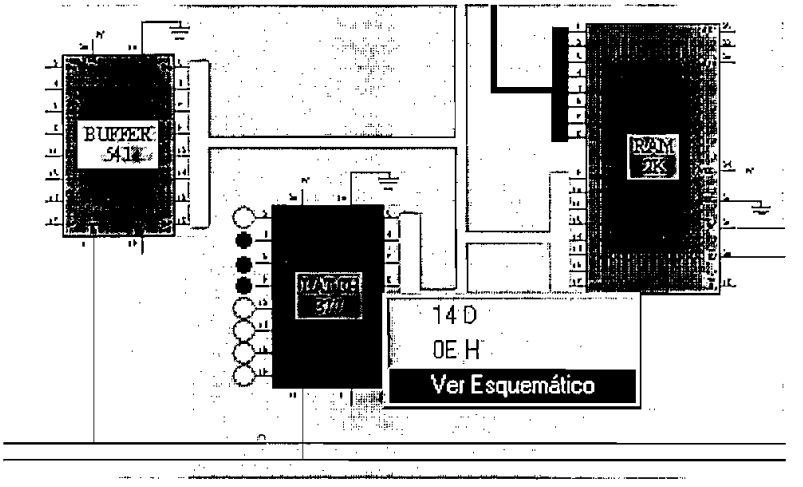


Fig.5.9 Resultado de la Escritura en el Latch

Antes de ejecutar las instrucciones que permiten leer del Buffer, se configura los interruptores (Fig.5.10) de tal forma que el valor leído sea 09H

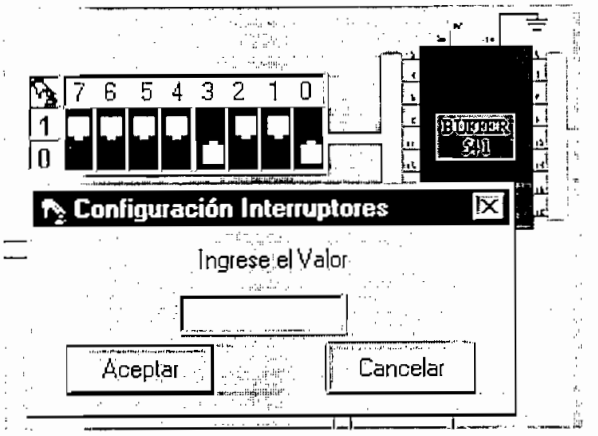
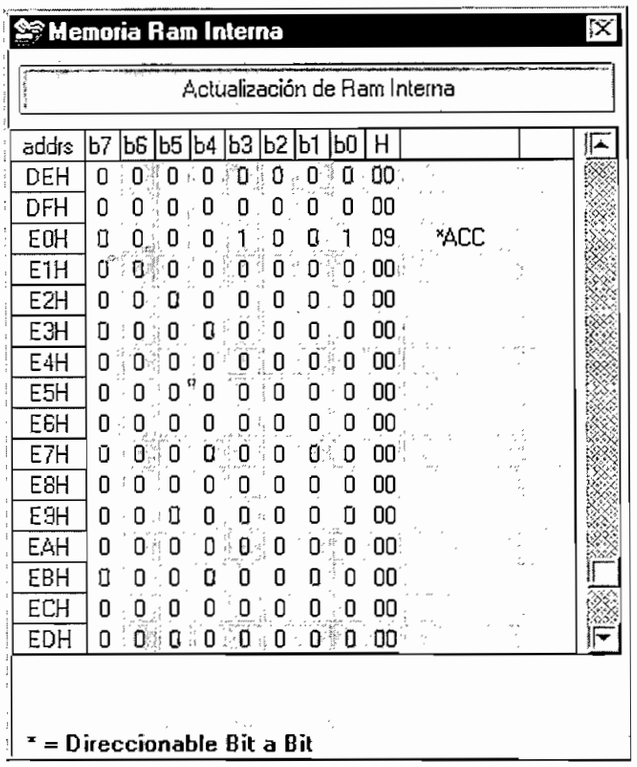


Fig.5.10 Configuración de los Interruptores del Buffer

Ejecutar las siguientes 2 instrucciones que permiten direccionar y leer del Buffer :

```
MOV     DPTR,#4020H
MOVX    A,@DPTR
```

El resultado se puede observar en la Fig.5.11, en donde el contenido del acumulador es 09H, que fue el valor configurado en los interruptores conectados al Buffer.



Memoria Ram Interna

Actualización de Ram Interna

addrs	b7	b6	b5	b4	b3	b2	b1	b0	H	
DEH	0	0	0	0	0	0	0	0	00	
DFH	0	0	0	0	0	0	0	0	00	
E0H	0	0	0	0	1	0	0	1	09	*ACC
E1H	0	0	0	0	0	0	0	0	00	
E2H	0	0	0	0	0	0	0	0	00	
E3H	0	0	0	0	0	0	0	0	00	
E4H	0	0	0	0	0	0	0	0	00	
E5H	0	0	0	0	0	0	0	0	00	
E6H	0	0	0	0	0	0	0	0	00	
E7H	0	0	0	0	0	0	0	0	00	
E8H	0	0	0	0	0	0	0	0	00	
E9H	0	0	0	0	0	0	0	0	00	
EAH	0	0	0	0	0	0	0	0	00	
EBH	0	0	0	0	0	0	0	0	00	
ECH	0	0	0	0	0	0	0	0	00	
EDH	0	0	0	0	0	0	0	0	00	

* = Direccionable Bit a Bit

Fig.5.11 Resultado de la Lectura del Buffer

Al ejecutar la siguiente instrucción (MOV @DPTR,A) se está tratando de escribir en una dirección no válida, ya que el DPRT está apuntando a una dirección decodificada como de Lectura, el V.I.M.P. mostrará un mensaje indicando que no puede escribir en esa dirección y terminará de ejecutar la instrucción.

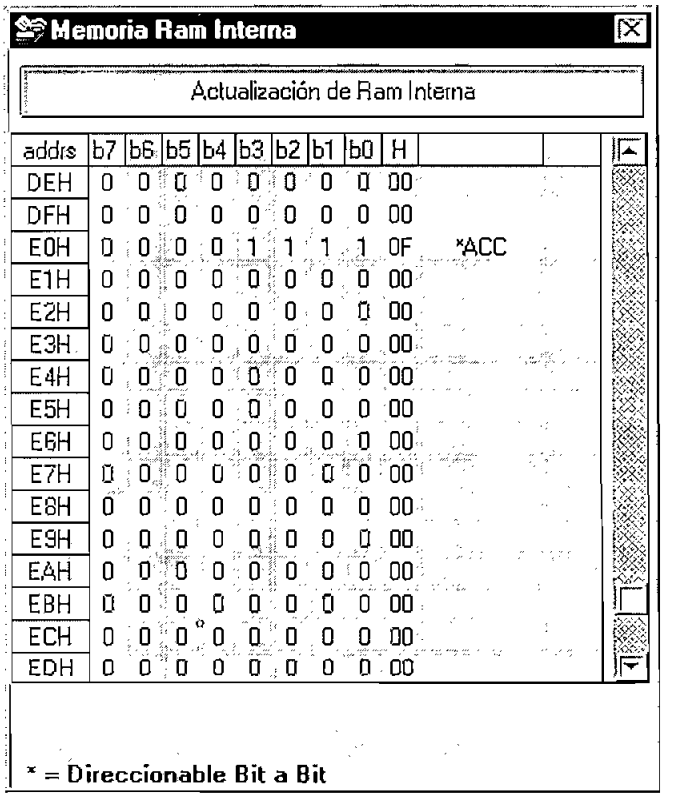
Ejecutar las siguientes 3 instrucciones que permiten encerar el contenido del acumulador, direccionar y leer de la ROM :

```
MOV     DPTR,#01H
CLR     A
MOVC    A,@A+DPTR
```

Ejecutar las siguientes 3 instrucciones que permiten encerar el contenido del acumulador, direccionar y leer de la ROM :

```
MOV     DPTR,#01H
CLR     A
MOVC    A,@A+DPTR
```

El resultado se puede observar en la Fig.5.12, en donde el contenido del acumulador es el valor contenido en la dirección 01H de la ROM que en este caso es el segundo byte de la primera instrucción del programa cargado(0FH)



addrs	b7	b6	b5	b4	b3	b2	b1	b0	H	
DEH	0	0	0	0	0	0	0	0	00	
DFH	0	0	0	0	0	0	0	0	00	
E0H	0	0	0	0	1	1	1	1	0F	*ACC
E1H	0	0	0	0	0	0	0	0	00	
E2H	0	0	0	0	0	0	0	0	00	
E3H	0	0	0	0	0	0	0	0	00	
E4H	0	0	0	0	0	0	0	0	00	
E5H	0	0	0	0	0	0	0	0	00	
E6H	0	0	0	0	0	0	0	0	00	
E7H	0	0	0	0	0	0	0	0	00	
E8H	0	0	0	0	0	0	0	0	00	
E9H	0	0	0	0	0	0	0	0	00	
EAH	0	0	0	0	0	0	0	0	00	
EBH	0	0	0	0	0	0	0	0	00	
ECH	0	0	0	0	0	0	0	0	00	
EDH	0	0	0	0	0	0	0	0	00	

Fig.5.12 Resultado de la Lectura de ROM

5.2.2 Programa Ejemplo de Llamado a Subrutinas y Saltos.

a) Código y Resultados Esperados.

CODIGO DEL PROGRAMA	RESULTADOS ESPERADOS
DATO EQU,0CH	Número de C.M. = 00 Número de Bytes = 00 Al igual que en los programas ensambladores, esta línea de código no es mas que una ayuda, y lo que hará es asignar a la etiqueta DATO el valor de 0CH.
MOV R0,#60H	Número de C.M. = 1 Número de Bytes = 2 1erByte = 78H 2doByte = 60H El Registro R0 del Banco 0 deberá contener el número 60H, que es una dirección de RAM interna
MOV @R0,#DATO	Número de C.M. = 1 Número de Bytes = 2 1erByte = 76H 2doByte = 0CH En la dirección 60H de la RAM interna debe cargarse el valor 0CH.
NOP	Número de C.M. = 1 Número de Bytes = 1 1erByte = 00H Se ejecuta la instrucción sin realizar operación alguna.
MOV A,@R0	Número de C.M. = 1 Número de Bytes = 1 1erByte = E6H El Acumulador deberá cargarse con el mismo valor de la dirección de memoria interna, es decir 0CH.
SALTO2 JZ FIN	Número de C.M. = 2 Número de Bytes = 2 1erByte = 60H 2doByte = 15H Esta instrucción revisa el contenido del Acumulador, si es 00H saltará a la etiqueta FIN.

PUSH A	Número de C.M. = 2 Número de Bytes = 2 1erByte = C0H 2doByte = E0H Luego de esta instrucción el registro puntero del stack que se inicializa en 2FH se incrementa en 1, es decir deberá contener el valor 30H que es la primera localidad de memoria interna reservada para el STACK ; y en esta localidad deberá guardarse el valor que contiene el acumulador (0CH).
LCALL SUBR1	Número de C.M. = 2 Número de Bytes = 3 1erByte = 12H 2doByte = 00H 3erByte = 21H Luego de esta instrucción el registro puntero del stack deberá incrementarse en 2, es decir deberá contener el valor 32H, y en las localidades 31H y 32H del STACK deberá guardarse el byte bajo (0EH) y el byte alto (00H) del contador de programa respectivamente.
POP A	Número de C.M. = 2 Número de Bytes = 2 1erByte = D0H 2doByte = E0H Luego de esta instrucción el registro puntero del stack deberá disminuir en 1, es decir deberá contener el valor 2FH, y el acumulador deberá recuperar el valor 0CH que teníamos guardado en 30H ; además esta localidad deberá mantener ese valor.
MOV R1,#61H	Número de C.M. = 1 Número de Bytes = 2 1erByte = 71H 2doByte = 61H El contenido del Registro R1 del Banco 0 deberá contener el número 61H, que es una dirección de RAM interna
MOV @R1,#10001111B	Número de C.M. = 1 Número de Bytes = 2 1erByte = 77H 2doByte = 8FH En la dirección 61H de la RAM interna debe cargarse el valor 10001111B.
XCHD A, @R1	Número de C.M. = 1 Número de Bytes = 1 1erByte = D7H El Acumulador contenía el valor 00001100B,después de esta operación deberá intercambiar sus 4 bits menos significativos con los de la localidad 61H obteniendose en el Acumulador 00001111B y en la localida 61H el valor 10001100B.

XCH A, @R1	Número de C.M. = 1 Número de Bytes = 1 1erByte = C7H El Acumulador ,después de esta operación deberá intercambiar sus 8 bits con los de la localidad 61H obteniendose en el Acumulador 10001100B y en la localida 61H el valor 00001111B.
ANL A,04H	Número de C.M. = 1 Número de Bytes = 2 1erByte = 55H 2doByte = 04H El Acumulador ,después de esta operación deberá contener el valor 00000100B.
SALTO1 DEC A	Número de C.M. = 1 Número de Bytes = 1 1erByte = 14H El microcontrolador decrementará el valor del Acumulador en 1 bit.
JNZ SALTO1	Número de C.M. = 2 Número de Bytes = 2 1erByte = 70H 2doByte = FDH Esta instrucción revisa el contenido del Acumulador, si no es 00H saltará a la etiqueta SALTO1. Cuando el Acumulador sea 00H el microcontrolador ejecutará la siguiente instrucción.
LJMP SALTO2	Núnero de C.M. = 2 Número de Bytes = 3 1erByte = 02H 2doByte = 00H 3erByte = 07H Luego de esta instrucción, el microcontrolador deberá ir a ejecutar la instrucción que se encuentra bajo esta etiqueta.
FIN CPL A	Número de C.M. = 1 Número de Bytes = 1 1erByte = F4H El Acumulador ,después de esta operación deberá contener el valor 11111111B.
SJMP $	Número de C.M. = 2 Número de Bytes = 2 1erByte = 80H 2doByte = FEH Luego de esta instrucción el microntrolador ejecutará el programa una y otra vez, es decir entrará en un lazo de ejecución infinito.

SUBR1 SWAP A	Número de C.M. = 1 Número de Bytes = 1 1erByte = C4H El Acumulador que contenía el valor binario 00001100B deberá tener el valor 11000000B.
LCALL SUBR2	Número de C.M. = 2 Número de Bytes = 3 1erByte = 12H 2doByte = 00H 3erByte = 26H Luego de esta instrucción el registro puntero del stack deberá incrementarse en 2, es decir deberá contener el valor 34H, y en las localidades 33H y 34H del STACK deberá guardarse el byte bajo (23H) y el byte alto (00H) del contador de programa respectivamente.
RET	Número de C.M. = 2 Número de Bytes = 1 1erByte = 12H Luego de esta instrucción el registro puntero del stack deberá disminuir en 2, es decir deberá contener el valor 32H, y el contador de programa tomará la dirección contenida en las localidades 33H y 34H del STACK (23H byte bajo y 00H byte alto), además estas localidades deberán mantener esos valores.
SUBR2 RL A	Número de C.M. = 1 Número de Bytes = 1 1erByte = 23H El Acumulador que contenía el valor binario 00001100B deberá tener el valor 10000001B
MOV B,#F0H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = F0H 3erByte = F0H El Registro B deberá tener el valor F0H
MUL AB	Número de C.M. = 4 Número de Bytes = 1 1erByte = A4H Luego de la operación (A=81H X B=F0H) el Acumulador contendrá el byte bajo del resultado (FDH) y el registro B el byte alto (78H) ; además la banderá de desborde OV deberá ser activada para indicar que el resultado es de más de 8 bits.

RET	Número de C.M. = 2 Número de Bytes = 1 1erByte = 22H Luego de esta instrucción el registro puntero del stack deberá disminuir en 2, es decir deberá contener el valor 30H, y el contador de programa tomará la dirección contenida en las localidades 31H y 32H del STACK (0EH byte bajo y 00H byte alto), además estas localidades deberán mantener esos valores.

b) Ejecución y Resultados Obtenidos

En primer lugar se deben seguir los pasos explicados en el ejemplo anterior antes de comenzar a ejecutar las instrucciones del archivo Ejemplo2.asm.

Como se puede observar en la consulta del código detallado del programa (Fig.5.13 y Fig.5.14), el número de ciclos de máquina, el número y los bytes de cada instrucción que han sido asignados por el V.I.M.P. a cada una de ellas, concuerdan con los resultados que anteriormente se había determinado como resultados esperados.

Para verificar el contenido del registro involucrado después de la ejecución de cada instrucción, como se hizo en el ejemplo anterior, el usuario puede cargar en el programa V.I.M.P. el Archivo **Ejemplo2.asm.** que se encuentra en el directorio **ArUsuario.**

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO1	OPERANDO2	OPERANDO3	#BYTES	#C.M	BYTE 1	BYTE 2	BYTE 3
2		MOV	R0	#60H		2	1	78	60	00
3		MOV	@R0	#0CH		2	1	76	0C	00
4		MOV	A	0CH		2	1	E5	0C	00
5		MOV	A	@R0		1	1	E6	00	00
6	SALTO2	JZ	FIN			2	2	60	15	00
7		PUSH	A			2	2	C0	E0	00
8		LCALL	SUBR1			3	2	12	00	21
9		POP	A			2	2	D0	E0	00
10		MOV	R1	#61H		2	1	79	61	00
11		MOV	@R1	#10001111B		2	1	77	8F	00
12		XCHD	A	@R1		1	1	D7	00	00
13		XCH	A	@R1		1	1	C7	00	00
14		ANL	A	04H		2	1	55	04	00
15	SALTO1	DEC	A			1	1	14	00	00
16		JNZ	SALTO1			2	2	70	FD	00

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO1	OPERANDO2	OPERANDO3	#BYTES	#C.M	BYTE 1	BYTE 2	BYTE 3
14		ANL	A	04H		2	1	55	04	00
15	SALTO1	DEC	A			1	1	14	00	00
16		JNZ	SALTO1			2	2	70	FD	00
17		LJMP	SALTO2			3	2	02	00	07
18	FIN	CPL	A			1	1	F4	00	00
19		SJMP	$			2	2	80	00	00
20								00	00	00
21	SUBR1	SWAP	A			1	1	C4	00	00
22		LCALL	SUBR2			3	2	12	00	26
23		RET				1	2	22	00	00
24	SUBR2	RL	A			1	1	23	00	00
25		MOV	240	#F0H		3	2	75	F0	F0
26		MUL	A	240		1	4	A4	00	00
27		RET				1	2	22	00	00
28								00	00	00

Fig5.13 Código Detallado del Programa

5.2.3 Programa Ejemplo de Pórticos,Temporizadores e Interrupciones.

a) Código y Resultados Esperados.

CODIGO DEL PROGRAMA	RESULTADOS ESPERADOS
CERO EQU,00H	Número de C.M. = 00 Número de Bytes = 00 Al igual que en los programas ensambladores, esta línea de código no es mas que una ayuda, y lo que hará es asignar a la etiqueta CERO el valor de 00H.
LJMP PROGP	Número de C.M. = 2 Número de Bytes = 3 1erByte = 02H 2doByte = 00H 3erByte = 2BH Esta instrucción permite al microcontrolador ir a ejecutar la primera instrucción del programa principal que se encuentra bajo la etiqueta PROGP
ORG EXTI0	Número de C.M. = 00 Número de Bytes = 00 Al igual que en los programas ensambladores, esta línea de código no es mas que una ayuda, y lo que hace es indicar el comienzo del área de memoria asignada para la interuupción externa 0.
LJMP RUTEX0	Número de C.M. = 2 Número de Bytes = 3 1erByte = 02H 2doByte = 00H 3erByte = 5AH Esta instrucción se ejecuta si se ha producido una interrupción externa, y permite al microcontrolador ir a ejecutar la primera instrucción de la subrutina RUTEX0
PROGP SETB EA	Número de C.M. = 1 Número de Bytes = 2 1erByte = D2H 2doByte = AFH Primera instrucción del programa principal, y permite activar el registro de habilitación de interrupciones. El bit EA deberá estar en 1_L

MOV P1,#CERO	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 90H 3erByte = 00H Mueve al puerto P1 el valor 00H, En la dirección 90H de la RAM interna debe cargarse el valor 00H, y el resultado en los 4 Leds, se lo podrá obsevar cuando se ejecute el S1P1 de la siguiente intrucción
SETB C	Número de C.M. = 1 Número de Bytes = 1 1erByte = D3H El valor del Carry es puesto a 1_L.
MOV P1.4,C	Número de C.M. = 2 Número de Bytes = 2 1erByte = 92H 2doByte = 94H Instucción de modificación y escritura que permite mover a P1.4 el valor 1_L que debe contener el Carry. El resultado puede ser consultado en la dirección 90H de la RAM interna y se lo podrá observar gráficamente cuando se ejecute el S1P1 de la siguiente intrucción
MOV TMOD,#00000001B	Número de C.M. = 2 Número de Bytes = 2 1erByte = 75H 2doByte = 81H 3erByte = 01H Luego de esta instrucción, el registro TMOD deberá contener el valor 01H habilitando el Temporizador 0 en Modo 0.
LCALL ESPERA	Número de C.M. = 2 Número de Bytes = 3 1erByte = 12H 2doByte = 00H 3erByte = 3FH Luego de esta instrucción el registro puntero del stack deberá incrementarse en 2, es decir deberá contener el valor 31H, y en las localidades 30H y 31H del STACK deberá guardarse el byte bajo (39H) y el byte alto (00H) del contador de programa respectivamente.
MOV C,P1.3	Número de C.M. = 1 Número de Bytes = 2 1erByte = A2H 2doByte = 13H Luego de esta instucción el Carry deberá contener el valor al que se ha configurado el interruptor conectado a este pin; para este ejemplo será 1_L.

CLR TR0	Número de C.M. = 1 Número de Bytes = 2 1erByte = C2H 2doByte = 8CH El bit TF0 deberá estar en 1_L .
RET	Número de C.M. = 2 Número de Bytes = 1 1erByte = 22H Luego de esta instrucción el registro puntero del stack deberá disminuir en 2, es decir deberá contener el valor 2FH, y el contador de programa tomará la dirección contenida en las localidades 30H y 31H del STACK (39H byte bajo y 00H byte alto), además estas localidades deberán mantener esos valores.
RUTEX0 SETB RS0	Número de C.M. = 1 Número de Bytes = 2 1erByte = D2H 2doByte = D3H Subrutina de atención a la Interrupción externa 0. El bit RS0 del PSW (bit.3) deberá estar en 1_L , y el Banco activo será el Banco 1.
MOV R0,#01110001B	Número de C.M. = 2 Número de Bytes = 2 1erByte = 80H 2doByte = FEH EL Registro R0 del Banco 1 deberá contener el valor 01110001B.
CLR RS0	Número de C.M. = 1 Número de Bytes = 1 1erByte = C4H El bit RS0 del PSW (bit.3) deberá estar en 0_L , y el Banco activo será el Banco 0.
CLR EX0	Número de C.M. = 2 Número de Bytes = 3 1erByte = 12H 2doByte = 00H 3erByte = 26H El bit EX0 del registro de habilitación de insterrupciones externas IE (bit.0) deberá estar en 0_L , deshabilitando por tanto la atención a la interrupción externa 0.
CLR IE0	Número de C.M. = 2 Número de Bytes = 1 1erByte = 12H La bandera de interrupción IE0 que fue activada por la interrupción externa 0 del registro TCON (bit.1) deberá estar en 0_L .

MOV R2,#1D	Número de C.M. = 1 Número de Bytes = 1 1erByte = 23H Se comprobará que el banco activo es el Banco 0 nuevamente. El registro R2 se cargará con el valor 00000001B.
RETI	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = F0H 3erByte = F0H Sale de la subrutina que atiende a la interrupción y continúa con la ejecución de la Registro B deberá tener el valor F0H.

b) Ejecución y Resultados Obtenidos

Este ejemplo, al igual que los anteriores deben seguir los pasos necesarios antes de comenzar a ejecutar las instrucciones del archivo Ejemplo3.asm.

Nuevamente, se puede observar en la consulta del código detallado del programa (Fig.5.15 a Fig.5.18), que el número de ciclos de máquina, el número y los bytes de cada instrucción que han sido asignados por el V.I.M.P. a cada una de ellas, concuerdan con los resultados que anteriormente se había determinado como resultados esperados. Las instrucciones NOP en el código detallado del programa permiten reservar los espacios de memoria de programa asignados por definición a

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO1	OPERANDO2	OPERANDO3	#BYTES	#CIM	BYTE 1	BYTE 2	BYTE 3
1		LJMP	PROGP			3	2	02	00	2B
2	ORG	EXTI0						00	00	00
3		LJMP	RUTEX0			3	2	02	00	4F
4		NOP				1	1	00	00	00
5		NOP				1	1	00	00	00
6		NOP				1	1	00	00	00
7		NOP				1	1	00	00	00
8		NOP				1	1	00	00	00
9	ORG	TIMER0						00	00	00
10		LJMP	RUTT0			3	2	02	00	5C
11		NOP				1	1	00	00	00
12		NOP				1	1	00	00	00
13		NOP				1	1	00	00	00
14		NOP				1	1	00	00	00
15		NOP				1	1	00	00	00
16	ORG	EXTI1						00	00	00

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO1	OPERANDO2	OPERANDO3	#BYTES	#CIM	BYTE 1	BYTE 2	BYTE 3
17		LJMP	RUTEX1			3	2	02	00	5D
18		NOP				1	1	00	00	00
19		NOP				1	1	00	00	00
20		NOP				1	1	00	00	00
21		NOP				1	1	00	00	00
22		NOP				1	1	00	00	00
23	ORG	TIMER1						00	00	00
24		LJMP	RUTT1			3	2	02	00	5E
25		NOP				1	1	00	00	00
26		NOP				1	1	00	00	00
27		NOP				1	1	00	00	00
28		NOP				1	1	00	00	00
29		NOP				1	1	00	00	00
80	ORG	SINT						00	00	00
81		LJMP	RUTPS			3	2	02	00	5F
82		NOP				1	1	00	00	00

Fig5.15 Código Detallado del Programa

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO 1	OPERANDO 2	OPERANDO 3	#BYTES	#CM	BYTE 1	BYTE 2	BYTE 3
33		NOP				1	1	00	00	00
34		NOP				1	1	00	00	00
35		NOP				1	1	00	00	00
36		NOP				1	1	00	00	00
37	PROGP	SETB	EA			2	1	D2	AF	00
38		MOV	144	#0D		3	2	75	90	00
39		SETB	C			1	1	D3	00	00
40		MOV	P1.4	C		2	2	92	94	00
41		MOV	137	#00000001B		3	2	75	89	01
42		LCALL	ESPERA			3	2	12	00	3F
43		MOV	C	P1.3		2	1	A2	93	00
44		MOV	R0	#01110000B		2	1	78	70	00
45		SJMP	$			2	2	80	FE	00
46	ESPERA	MOV	140	#0FFH		3	2	75	8C	FF
47		MOV	138	#0FBH		3	2	75	8A	FB
48		SETB	TR0			2	1	D2	8C	00

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO 1	OPERANDO 2	OPERANDO 3	#BYTES	#CM	BYTE 1	BYTE 2	BYTE 3
49	WAIT	JNB	TF0	WAIT		3	2	30	8D	FD
50		CLR	TF0			2	1	C2	8D	00
51		CLR	TR0			2	1	C2	8C	00
52		RET				1	2	22	00	00
53	RUTEX0	SETB	RS0			2	1	D2	D3	00
54		MOV	R0	#011100018		2	1	78	71	00
55		CLR	RS0			2	1	C2	D3	00
56		CLR	EX0			2	1	C2	A8	00
57		CLR	IE0			2	1	C2	89	00
58		MOV	R2	#1D		2	1	7A	01	00
59		RETI				1	2	32	00	00
60	RUTT0	RETI				1	2	32	00	00
61	RUTEX1	RETI				1	2	32	00	00
62	RUTT1	RETI				1	2	32	00	00
63	RUTPS	RETI				1	2	32	00	00

Fig5.15 Código Detallado del Programa (Continuación)

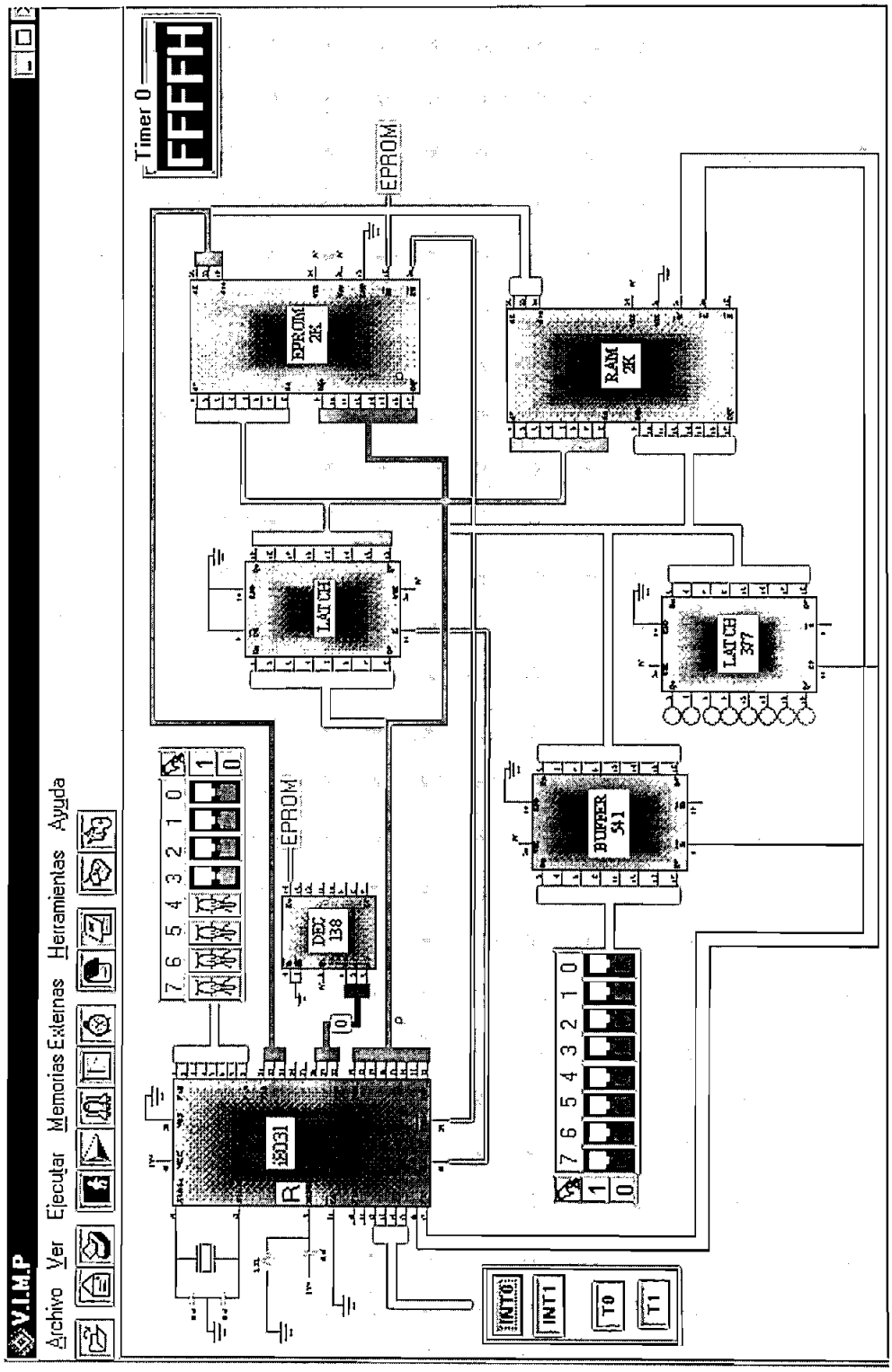


Fig5.19 Visualización del Temporizador

5.2.4 Programa Demo del Pórtico Serial

a) Código y Resultados Esperados.

CODIGO DEL PROGRAMA	RESULTADOS ESPERADOS
;Para Transmisión MOV SCON,#60H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 98H 3erByte = 60H Permite configurar al Pórtico Serial para que trabaje en Modo 1.
MOV TH1,#254D	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 8DH 3erByte = FEH Se configura el byte alto del Timer 1 en el valor 254 para definir la velocidad a la que trabajará el Pórtico Serial.
MOV PCON,#80H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 87H 3erByte = 80H Se pone el bit SMOD en 1, y junto con el valor contenido en TH1 se tiene la velocidad de transmisión en 31250 bps
MOV A,P1	Número de C.M. = 1 Número de Bytes = 2 1erByte = E5H 2doByte = 90H Se toma los datos de los interruptores del Pórtico P1, que luego serán enviados como salida serial.
MOV SBUF,A	Número de C.M. = 1 Número de Bytes = 2 1erByte = F5H 2doByte = 99H Se establece la condición para iniciar la transmisión serial del byte.

TX JNB TI,TX	Número de C.M. = 2 Número de Bytes = 3 1erByte = 30H 2doByte = 99H 3erByte = FDH Se investiga si la transmisión concluyó con la salida del bit de parada, verificando la habilitación de la bandera de interrupción TI.
NOP	Número de C.M. = 1 Número de Bytes = 1 1erByte = 00H Instrucción de retardo para espaciar envíos de información.
NOP	Número de C.M. = 1 Número de Bytes = 1 1erByte = 00H Instrucción de retardo para espaciar envíos de información.
CLR TI	Número de C.M. = 1 Número de Bytes = 2 1erByte = C2H 2doByte = 99H Se borra la bandera de interrupción TI para que no quede bloqueada la transmisión por el Pórtico Serial.
;Para Recepción MOV TH1,#255D	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 8DH 3erByte = FFH Permitirá configurar el byte alto del Timer 1 en el valor 255 para definir junto con el bit SMOD del registro PCON la velocidad a la que trabajará el Pórtico Serial y que es 62500 bps
MOV SCON,#70H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 98H 3erByte = 70H Permite configurar al Pórtico Serial para que trabaje en Modo 1, además los bits SM2 y REN son puestos en 1, con ello se está a la espera de que se detecte un flanco descendente (bit de inicio) en el pin P3.0 (RXD) para dar inicio a la recepción .

RX JNB RI,RX	Número de C.M. = 2 Número de Bytes = 3 1erByte = 30H 2doByte = FDH 3erByte = 98H Igualmente se investiga si la recepción concluyó con la llegada del bit de parada, verificando la habilitación de la bandera de interrupción RI.
NOP	Número de C.M. = 1 Número de Bytes = 1 1erByte = 00H Instrucción de retardo.
NOP	Número de C.M. = 1 Número de Bytes = 1 1erByte = 00H Instrucción de retardo.
MOV SCON,#60H	Número de C.M. = 2 Número de Bytes = 3 1erByte = 75H 2doByte = 98H 3erByte = 60H Se deja el Pórtico Serial en Modo 1 con el bit REN deshabilitado evitando con ello la disponibilidad de pórtico para recibir datos.
CLR RI	Número de C.M. = 1 Número de Bytes = 2 1erByte = C2H 2doByte = 98H Se borra la bandera de interupción RI.
MOV A,SBUF	Número de C.M. = 1 Número de Bytes = 2 1erByte = E5H 2doByte = 99H El byte que llegó por el Pórtico Serial se mueve al Acumulador, para la verificación del byte.
SJMP $	Número de C.M. = 2 Número de Bytes = 2 1erByte = 80H 2doByte = FEH Se establece un lazo infinito en el programa.

b) Ejecución y Resultados Obtenidos

Las líneas de código antes indicadas, permitirán al usuario realizar un visualización del trabajo del Pórtico Serial tanto en transmisión como en recepción.

En cuanto a la transmisión se ha establecido los valores en los registros necesarios para que la velocidad de transmisión sea de 31250 bps, con ello cada bit transmitido necesitará 32 microseg para ser enviado, lo cual indica si se asume una frecuencia de oscilador de 12 Mhz que cada bit empleará al menos 32 ciclos de máquina completos. Para la parte de recepción se establece una velocidad de 62500 bps , es decir, 16 microseg por bit; traduciendo a ciclos de máquina asumiendo asi mismo una frecuencia de oscilador de 12 Mhz será igual a 12 ciclos de máquina por bit

La habilitación de las banderas de interrupción, para transmisión TI y recepción RI del Pórtico Serial no permitirán saltar a la rutina de atención a la interrupción del pórtico ya que el registro de habilitación de interrupciones IE no esta configurado para ello.

Las Fig.5.20 muestra el código detallado usado para indicar el funcionamiento del Pórtico Serial en Modo 1.

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO 1	OPERANDO 2	OPERANDO 3	# BYTES	# C.M	BYTE 1	BYTE 2	BYTE 3
1								00	00	00
2		MOV	152	#60H		3	2	75	98	60
3		MOV	141	#254D		3	2	75	8D	FE
4		MOV	135	#80H		3	2	75	87	80
5		MOV	A	144		2	1	E5	90	00
6		MOV	153	A		2	1	F5	99	00
7	TX	JNB	TI	TX		3	2	30	99	FD
8		NOP				1	1	00	00	00
9		NOP				1	1	00	00	00
10		CLR	TI			2	1	C2	99	00
11								00	00	00
12		MOV	141	#255D		3	2	75	8D	FF
13		MOV	152	#70H		3	2	75	98	70
14	RX	JNB	RI	RX		3	2	30	98	FD
15		NOP				1	1	00	00	00

Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO 1	OPERANDO 2	OPERANDO 3	# BYTES	# C.M	BYTE 1	BYTE 2	BYTE 3
6		MOV	153	A		2	1	F5	99	00
7	TX	JNB	TI	TX		3	2	30	99	FD
8		NOP				1	1	00	00	00
9		NOP				1	1	00	00	00
10		CLR	TI			2	1	C2	99	00
11								00	00	00
12		MOV	141	#255D		3	2	75	8D	FF
13		MOV	152	#70H		3	2	75	98	70
14	RX	JNB	RI	RX		3	2	30	98	FD
15		NOP				1	1	00	00	00
16		NOP				1	1	00	00	00
17		MOV	152	#60H		3	2	75	98	60
18		CLR	RI			2	1	C2	98	00
19		MOV	A	153		2	1	E5	99	00
20		SJMP	$			2	2	80	FE	00

Fig.5.20 Código Detallado del Programa

Los interruptores de entrada al Pórtico P1 (usuados en la parte de transmisión de la demostración) y las etiquetas de entrada por el pin P3.0 (RXD) (usadas en la parte de recepción) pueden ser modificados sus contenidos por el usuario de acuerdo al progreso de la simulación, permitiendole trabajar con los bits que el usuario desee.

5.3. CONCLUSIONES Y RECOMENDACIONES

- El Programa desarrollado en esta tesis, constituye un software a través del cual futuros diseñadores de sistemas basados en microntroladores puedan disponer de una herramienta que les permita iniciarse en el conocimiento de la comunicación del microcontrolador con diferentes periféricos.

- La funcionalidad y utilidad de esta herramienta se pone de manifiesto cuando el usuario se de cuenta que el funcionamiento e interacción del microcontrolador no solamente puede ser asimilado como un proceso subjetivo que puede ser entendido de manera intuitiva, sino más bien como un proceso real que puede ser visto en cualquiera de sus etapas; permitiéndole de esta forma asimilar el concepto de eficiencia en el desarrollo de un programa.

- El desarrollo de este programa se justifica ampliamente si se considera la funcionalidad del mismo y el hecho de que en los actuales momentos los laboratorios donde se debe impulsar y consolidar el conocimiento de los

microcontroladores no disponen de los recursos suficientes para adquirir el equipo necesario.

- El programa desarrollado no debe ser considerado como un simulador, aunque se pretende que su filosofía sea esa, sino mas bien debe ser concebido como una herramienta visual que permite asimilar el funcionamiento e interacción del microcontrolador con distintos elementos periféricos.

- La selección de los elementos periféricos utilizados, se la ha realizado tomando en cuenta su representatividad, ya que obviamente no es posible mostrar la comunicación con todos los dispositivos existentes que pueden ser considerados como elementos periféricos; además, lo que se pretende es que el usuario pueda relacionar la interacción con ellos, en base a lo disponible.

- La potencialidad de la herramienta se la ve reflejada en el hecho de que no solamente se puede visualizar la interacción del microcontrolador con elementos periféricos, como un evento aislado que únicamente involucra a instrucciones de acceso a memoria externa, sino que también es posible ejecutar cualquiera de las instrucciones presentadas para el 8031, claro esta con las limitaciones en su momento establecidas.

- Otro aspecto que muestra la potencialidad del programa es el hecho de que se puede consultar el estado tanto de las memorias como de los registros de funciones especiales luego de la ejecución de una instrucción, permitiendo de este modo verificar si el microcontrolador opera adecuadamente cada instrucción leída de la ROM.

- La selección de una determinada configuración, ya sea física o lógica, influye en el funcionamiento del programa, por ello el código que se utilice deberá tomar en cuenta las limitaciones propias que cada configuración establece.

- Parte del código fuente tratando de hacerlo independiente de otros programas fundamento de peso que ha hecho posible el desarrolo de la presente tesis.

  Para hacer uso de este programa, No se requiere de que el usuario disponga del código de máquina de la instrucción o programa que desea probar, ya que lo que lee V.I.M.P. , es el archivo.asm , es decir los nemónicos de cada instrucción, razón por la que en parte ayudamos a interpretar la correcta sintaxis del código de prueba, puesto que analizamos palabra por palabra reconociendo si es o no una instrucción válida. Esta caractrerística lo hace diferente de otros programas que obligan primeramente a ensamblar el archivo que se quiere probar.

- El considerar mostrar el funcionamiento del pórtico serial únicamente como una demostración (Demo), rádica únicamente en el hecho de que para transmitir un bit en el Modo de prueba seleccionado (Modo 1), al microcontrolador le toma 32 ciclos de máquina, duración que en las escalas de tiempo ampliadas disponibles, harían que la visualización de este proceso tenga una duración demasiado grande.

- Luego de conocer como el microcontrolador accesa a las diferentes localidades de memoria tanto Interna como Externa, la clasificación hecha de ellas, puede ser considerarada como una formalidad y no como un factor preponderante, razón por la cual no se ha hecho énfasis en mostrar tal información en el programa.

- En cuanto a la habilitación de las interrupciones externas INT0 y INT1, es de conocimiento que se lo puede hacer por nivel o por flanco, la distinción de estas dos formas de habilitarlas no es diferenciada en nuestro programa, puesto que para visualizar el efecto que causa, no es necesario hacerlo.

- Recomendación importante para el uso de esta herramienta es tener presente las consideraciones y limitaciones establecidas, ya que el entendimiento de ellas permitirán explotar al máximo las bondades de esta herramienta que al mismo tiempo toma ventaja de una aplicación desarrollada bajo Windows.

- Se recomienda aquellas personas que crean que se deba añadir alguna funcionalidad a esta herramienta, tomar en cuenta las consideraciones y limitaciones establecidas, ya que son ellas las que estructuran y a la vez limitan la potencialidad del programa.

BIBLIOGRAFIA

- GONZALEZ VAZQUEZ JOSE, " Introducción a los Microcontroladores ", España 1992 McGraw-Hill.

- INTEL, " Microprocessor and Peripheral Handbook " Intel Corporation, 1989.

- JAIME VELARDE, Ing. "Curso de Sistemas Microprocesados", E.P.N. Quito 1994

- MARK STEVEN HEYMAN, " La Esencia del Visual Basic 4 ", México 1996 Prentice-Hall

- TEXAS INSTRUMENT, " TTL Handbook "

- FAIRCHILD Camera and Instrument Corp., " Full Line Condensed Catalog " , USA 1978

- Web : www.jameco.com

- Web : www.intel.com

- Web : www.mot.com

- RONALD J. TOCCI, "Sistema Digitales Principios y Aplicaciones ", México 1993 Prentice-Hall

ANEXOS

ANEXO A

CÓDIGO FUENTE

(VER CUADERNO DE CODIGO FUENTE)

ANEXO B

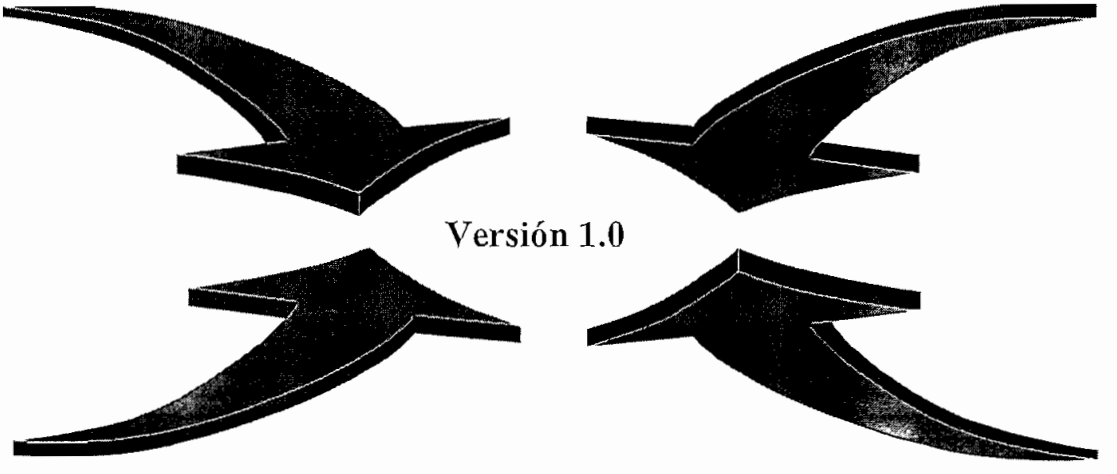
PROGRAMA PARA VISUALIZAR LA COMUNICACION

DE UN MICROCONTROLADOR

CON SUS PERIFÉRICOS

V.I.M.P.

MANUAL DE USUARIO



Quito - Ecuador

B.1 INTRODUCCION

V.I.M.P. es un programa de propósito educativo que ha sido desarrollado bajo el lenguaje de programación Visual Basic y permite visualizar la comunicación del microcontrolador 8031 con distintos periféricos.

V.I.M.P. fue desarrollado con el objetivo de crear una herramienta que facilite la tarea de aprendizaje del microcontrolador, mostrando en forma más objetiva su funcionamiento.

El desarrollo de este programa fue empujado por la necesidad de crear una herramienta que en cierto modo permita salvar la falta de equipo de laboratorio para el dasarrollo de aplicaciones mediante microcontroladores.

A través del siguiente manual se guiará al usuario sobre el manejo del programa para que pueda realizar las acciones deseadas de la forma más óptima y adecuada.

B.2 REQUERIMIENTOS PARA USAR EL PROGRAMA

Es importante describir los aspectos que son requisitos básicos e indispensable para poder utilizar el Programa de la forma más adecuada.

B.2.1 REQUISITOS DE HARDWARE

El hardware básico bajo el cual debe ser instalado el Programa V.I.M.P. debe constar básicamente de lo siguiente :

- Computador 486 Compatible

- ◊ Procesador 486
- ◊ Velocidad del procesador 100Mhz. min.
- ◊ Memoria 16MB en RAM
- ◊ Espacio en disco de 10MB
- ◊ Drive de 3.5"
- ◊ Monitor SVGA color de 14"
- ◊ Teclado
- ◊ Mouse

Para obtener un mejor rendimiento es recomendable un computador con las siguientes características :

- ◊ Procesador Pentium
- ◊ Velocidad del procesador 100Mhz. o superior

- ◊ Memoria 24MB en RAM
- ◊ Espacio en disco de 10MB
- ◊ Drive de 3.5"
- ◊ Monitor SVGA color de 14"
- ◊ Teclado
- ◊ Mouse

B.2.2 REQUISITOS DE SOFTWARE

El software de base mínimo que deberá estar instalado en el computador es el siguiente :

- Windows´95

Debido a que el programa ha sido realizado en el lenguaje de programación Microsoft Visual Basic 4.0, es necesario que en el computador este cargado como sistema operativo Microsoft WINDOWS´95 para poder cargarlo, permitiendonos de este modo hacer uso de la potencialidad del manejo del mouse y el teclado como medio de comunicación entre el usuario y el Programa V.I.M.P.

B.3 ¿COMO INSTALAR V.I.M.P. ?

Una vez que ha cumplido los requerimientos de Hardware y Software, para instalar V.I.M.P. en su computador se debe seguir el siguiente procedimiento:

- Inserte el disco etiquetado como # 1.

- De la barra de **Inicio** de Windows'95, seleccione la opción **Ejecutar**.

- En la pantalla Ejecutar **Examinar** la **Unidad A** , seleccionar el archivo de instalación **setup** , y **Aceptar** , como se muestra en la Fig.1.

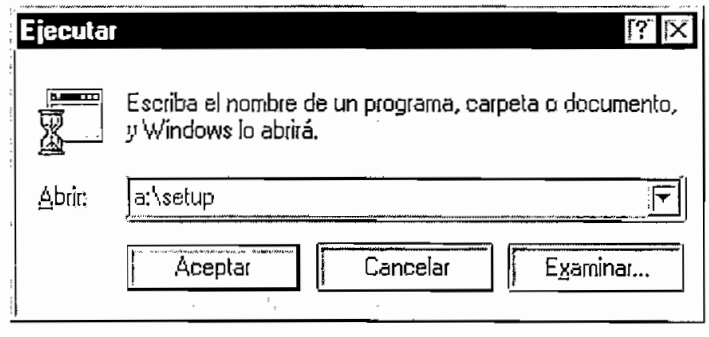


Fig.1 Instalación del Programa V.I.M.P.

- Cotinuar y terminar la instalación siguiendo los pasos que serán mostrados igual que en la instalación de cualquier aplicación bajo Windows.

B.4 COMO NAVEGAR A TRAVEZ DEL PROGRAMA

Para interactuar con el usuario el programa presenta pantallas ya sea con menúes desplegables o pantallas que presentan menúes con botones que muetran información asociada a una acción que en un determinado momento se desea hacer uso.

Una pantalla de menú consiste básicamente de la descripción de las opciones que el programa brinda al usuario.

Si consideramos la forma en que el programa presenta las pantallas, disponemos de tres maneras de navegar a través del programa, la primera por medio de pantallas de menúes desplegables, la segunda por medio de pantallas de menúes con botones y la tercera a través de menúes emergentes.

B.4.1 Navegando a través de los Menúes Desplegables.

La pantalla principal del V.I.M.P. presenta los siguientes menúes deplegables :

Fig.2 Menú principal desplegable

Archivo

Despliega las siguientes opciones :

- Abrir

Presenta una pantalla que permite cargar un archivo de programa ya sea desde la unidad de disco de la máquina o de la unidad de disco flexible.

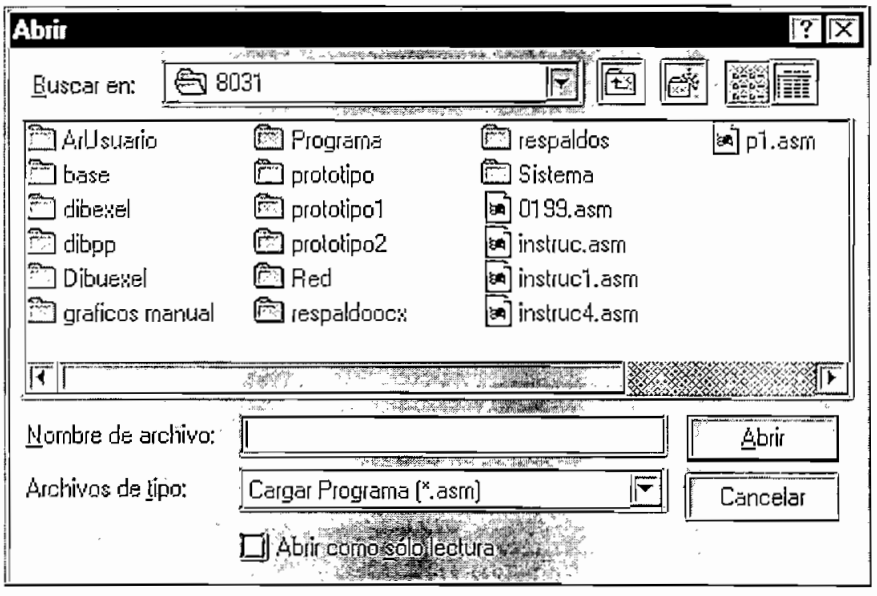


Fig.3 Pantalla para abrir un Archivo

- Salir

Opción que permite finalizar la sesión de trabajo en el V.I.M.P.

Ver

Muestra las siguientes opciones las cuales a la vez despliegan pantallas de acuerdo a la selección:

- Código F7

Permite ver el código del programa que ha sido cargado en el V.I.M.P., asociada a este evento está la tecla F7 (Fig.4).

- Código Detallado

Permite ver el código y toda la información referente a las instrucciones del programa que ha sido cargado en el V.I.M.P. (Fig.5).

- RAM Interna

Permite ver el contenido de los 256 bytes de la la RAM interna (Fig.6)

- Esquemáticos

Despliega un submenú con las siguientes opciones :

- Microcontrolador

Permite ver al detalle la distribución de pines del Microcontrolado (Fig.7).

- Decodificador

Permite ver al detalle la distribución de pines del Decodificador (Fig.8).

- Rom

Permite ver al detalle la distribución de pines de la ROM (Fig.9)

- Ram

Permite ver al detalle la distribución de pines de la Ram (Fig.10).

- Latch 74LS373

Permite ver al detalle la distribución de pines del Latch 74LS373 (Fig.11).

- Latch 74LS377

Permite ver al detalle la distribución de pines del Latch 74LS377 (Fig.12).

- Buffer 74LS541

Permite ver al detalle la distribución de pines del Buffer 74LS541 (Fig.13).

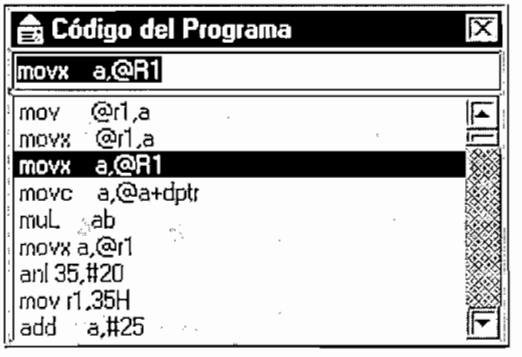
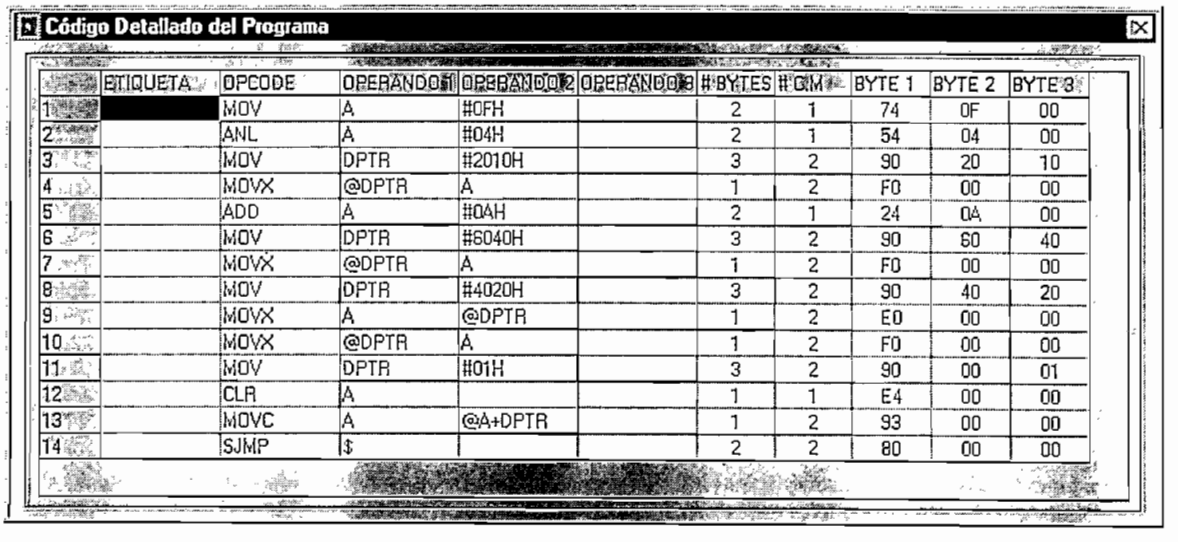


Fig.4 Pantalla para ver el Código del Programa



Código Detallado del Programa

	ETIQUETA	OPCODE	OPERANDO#1	OPERANDO#2	OPERANDO#3	# BYTES	# CIM	BYTE 1	BYTE 2	BYTE 3
1		MOV	A	#0FH		2	1	74	0F	00
2		ANL	A	#04H		2	1	54	04	00
3		MOV	DPTR	#2010H		3	2	90	20	10
4		MOVX	@DPTR	A		1	2	F0	00	00
5		ADD	A	#0AH		2	1	24	0A	00
6		MOV	DPTR	#6040H		3	2	90	60	40
7		MOVX	@DPTR	A		1	2	F0	00	00
8		MOV	DPTR	#4020H		3	2	90	40	20
9		MOVX	A	@DPTR		1	2	E0	00	00
10		MOVX	@DPTR	A		1	2	F0	00	00
11		MOV	DPTR	#01H		3	2	90	00	01
12		CLR	A			1	1	E4	00	00
13		MOVC	A	@A+DPTR		1	2	93	00	00
14		SJMP	$			2	2	80	00	00

Fig.5 Pantalla para ver el Código Detallado

Memoria Ram Interna

Actualización de Ram Interna

addrs	b7	b6	b5	b4	b3	b2	b1	b0	H		
13H	0	0	0	0	0	0	0	0	00	Banco 2	R3
14H	0	0	0	0	0	0	0	0	00	Banco 2	R4
15H	0	0	0	0	0	0	0	0	00	Banco 2	R5
16H	0	0	0	0	0	0	0	0	00	Banco 2	R6
17H	0	0	0	0	0	0	0	0	00	Banco 2	R7
18H	0	0	0	0	0	0	0	0	00	Banco 3	R0
19H	0	0	0	0	0	0	0	0	00	Banco 3	R1
1AH	0	0	0	0	0	0	0	0	00	Banco 3	R2
1BH	0	0	0	0	0	0	0	0	00	Banco 3	R3
1CH	0	0	0	0	0	0	0	0	00	Banco 3	R4
1DH	0	0	0	0	0	0	0	0	00	Banco 3	R5
1EH	0	0	0	0	0	0	0	0	00	Banco 3	R6
1FH	0	0	0	0	0	0	0	0	00	Banco 3	R7
20H	0	0	0	0	0	0	0	0	00	*DATOS	
21H	0	0	0	0	0	0	0	0	00	*DATOS	
22H	0	0	0	0	0	0	0	0	00	*DATOS	

* = Direccionable Bit a Bit

Fig.6 Pantalla para ver el contenido de la Ram Interna

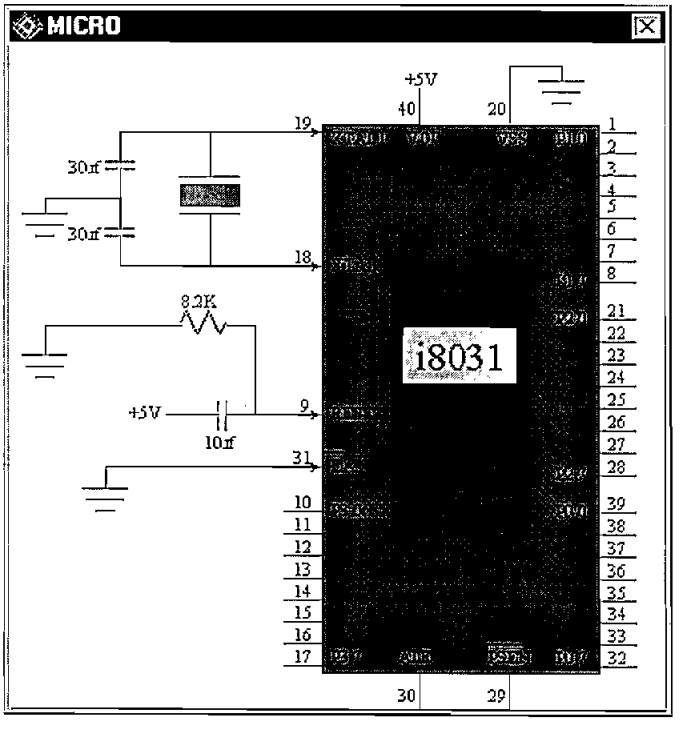


Fig.7 Pantalla con el Microcontrolador

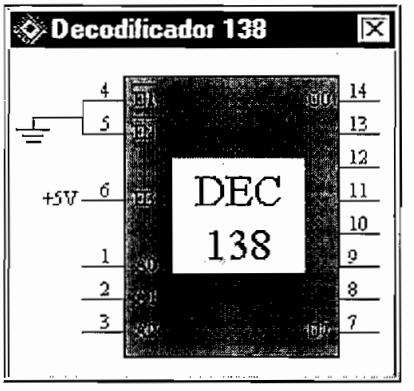


Fig.8 Pantalla con el Decodificador

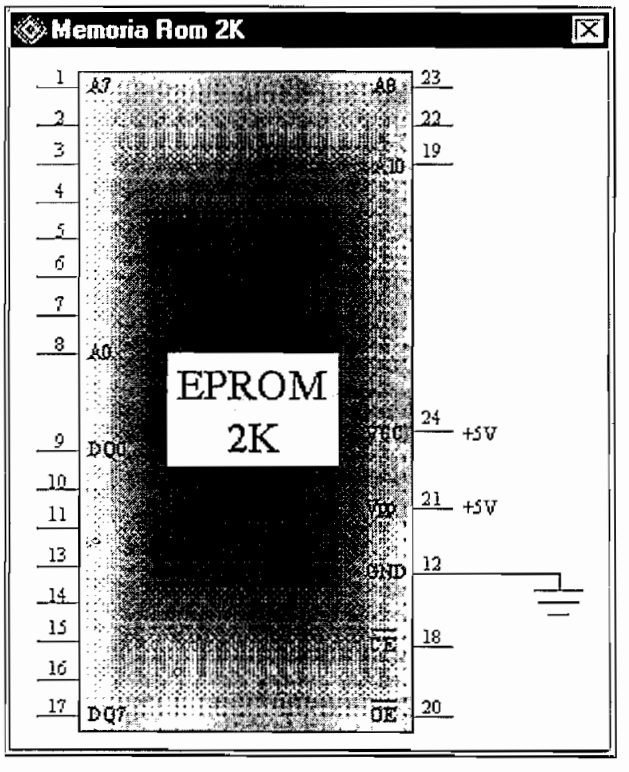


Fig.9 Pantalla con la Rom

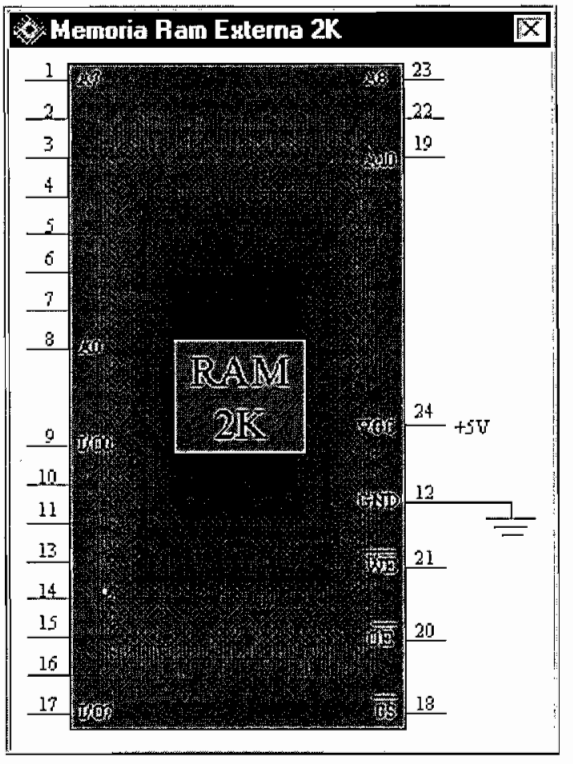


Fig.10 Pantalla con la Ram

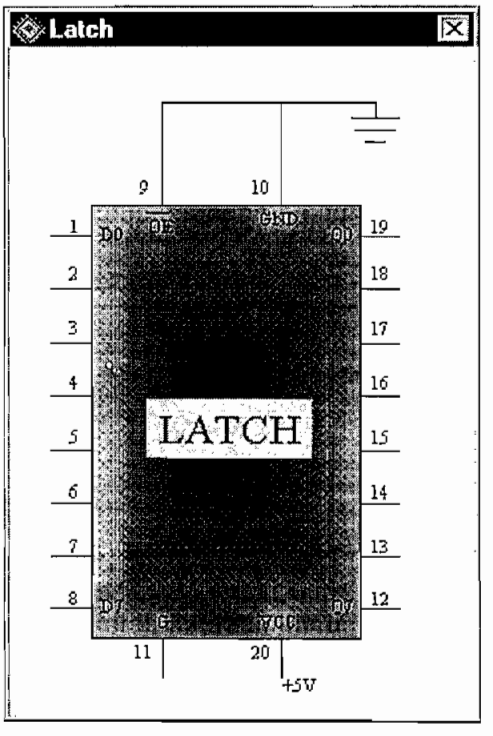


Fig.11 Pantalla con el Latch 74LS373

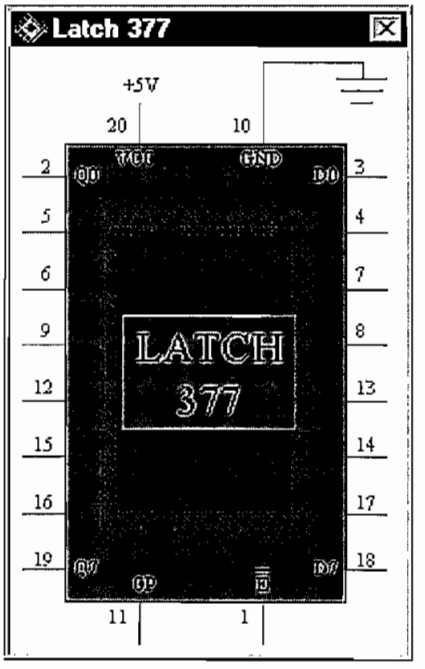


Fig.12 Pantalla con el Latch 74LS377

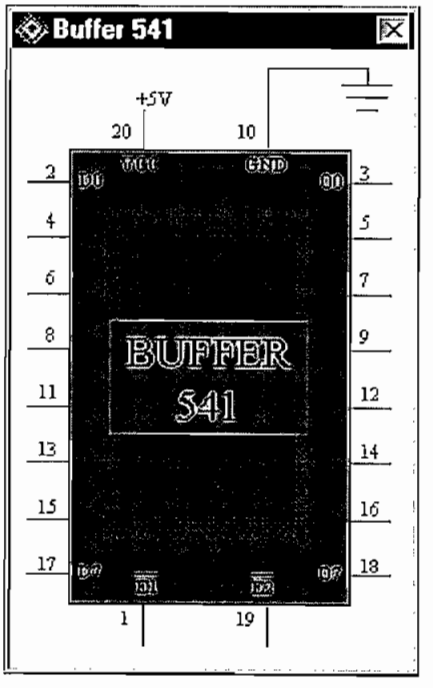


Fig.13 Pantalla con el Buffer 74LS541

Ejecutar

Opción que permite escoger la forma en que el V.I.M.P. ejecutará el programa que se ha cargado.

- Con Animación

Despliega el siguiente submenú :

- Todo F5

Desencadena el proceso mediante el cual el V.I.M.P. muestra gráfica y dinámicamente la forma en que el microcontrolador interactúa con los periféricos ejecutando todas las instrucciones del programa cargado de una sola vez. Asociada a este evento está la tecla F5

- Por Instrucción F8

Desencadena el proceso mediante el cual el V.I.M.P. muestra gráfica y dinámicamente la forma en que el microcontrolador interactúa con los periféricos ejecutando una sola instrucción del programa cargado. Asociada a este evento está la tecla F8

- Escala de Tiempo

Muestra un submenú que permite determinar una escala de tiempo asociada a la fase de un ciclo de máquina ; donde :

-1 Equivale a multiplicar cada fase por 2segundos.

-2 Equivale a multiplicar cada fase por 4segundos.

-3 Equivale a multiplicar cada fase por 6segundos.

-4 Equivale a multiplicar cada fase por 8segundos.

Esto permite que la ejecución de una instrucción en el modo con animación pueda ser vista más detenidamente o más rápidamente.

- Sin Animación

Despliega el siguiente submenú :

- Todo Ctrl F5

Desencadena el proceso mediante el cual el V.I.M.P. ejecuta todas las instrucciones del programa cargado de una sola vez pero sin mostrar gráficamente la forma en que el microcontrolador interactúa con los periféricos, y nos servirá para determinar de una manera rápida el estado de los registros y memorias al término de la ejecución del programa. Asociada a este evento está la convinación de las teclas Ctrl + F5.

- Por Instrucción Ctrl F8

Desencadena el proceso mediante el cual el V.I.M.P. ejecuta de una en una las instrucciones del programa cargado, pero sin mostrar gráficamente la

forma en que el microcontrolador interactúa con los periféricos, y nos servirá para determinar de una manera rápida el estado de los registros y memorias al término de la ejecución de cada una de las instrucciones. Asociada a este evento está la combinación de las teclas Ctrl + F8.

Memorias Externas

Muestra las siguientes opciones :

- Rom

Despliega un submenú de selección e información de la memoria Rom:

- 1 Kbytes
- 2 Kbytes
- 4 Kbytes
- Ver Contenido

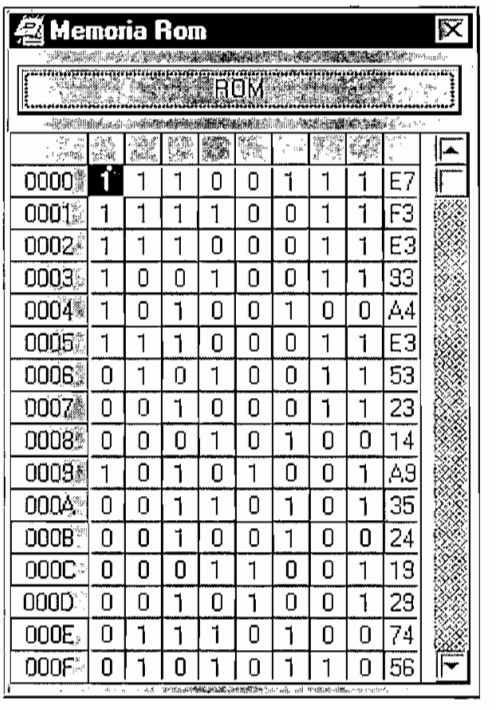
Despliega una pantalla con el contenido de la memoria (Fig.14).

- Ver Esquemático

Permite ver el detalle de la distribución de pines de esta memoria (Fig.9).

- Ram

Despliega un submenú de selección e información igual al de la ROM, y el contenido es mostrado en la Fig.15

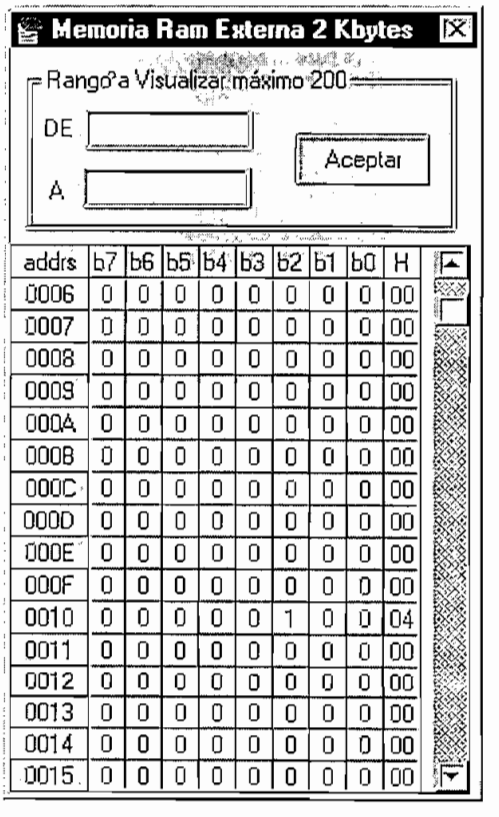


Memoria Rom

ROM

0000	1	1	1	0	0	1	1	1	E7
0001	1	1	1	1	0	0	1	1	F3
0002	1	1	1	0	0	0	1	1	E3
0003	1	0	0	1	0	0	1	1	93
0004	1	0	1	0	0	1	0	0	A4
0005	1	1	1	0	0	0	1	1	E3
0006	0	1	0	1	0	0	1	1	53
0007	0	0	1	0	0	0	1	1	23
0008	0	0	0	1	0	1	0	0	14
0009	1	0	1	0	1	0	0	1	A9
000A	0	0	1	1	0	1	0	1	35
000B	0	0	1	0	0	1	0	0	24
000C	0	0	0	1	1	0	0	1	19
000D	0	0	1	0	1	0	0	1	29
000E	0	1	1	1	0	1	0	0	74
000F	0	1	0	1	0	1	1	0	56

Fig.14 Pantalla con el contenido de la Rom



Memoria Ram Externa 2 Kbytes

Rango a Visualizar máximo 200

DE

A

Aceptar

addrs	b7	b6	b5	b4	b3	b2	b1	b0	H
0006	0	0	0	0	0	0	0	0	00
0007	0	0	0	0	0	0	0	0	00
0008	0	0	0	0	0	0	0	0	00
0009	0	0	0	0	0	0	0	0	00
000A	0	0	0	0	0	0	0	0	00
000B	0	0	0	0	0	0	0	0	00
000C	0	0	0	0	0	0	0	0	00
000D	0	0	0	0	0	0	0	0	00
000E	0	0	0	0	0	0	0	0	00
000F	0	0	0	0	0	0	0	0	00
0010	0	0	0	0	0	1	0	0	04
0011	0	0	0	0	0	0	0	0	00
0012	0	0	0	0	0	0	0	0	00
0013	0	0	0	0	0	0	0	0	00
0014	0	0	0	0	0	0	0	0	00
0015	0	0	0	0	0	0	0	0	00

Fig.15 Contenido de la Ram

Herramientas

Despliega las siguientes opciones :

- Editor de Instrucciones Ctrl-E

Muestra la pantalla de un editor de instrucciones que permite crear archivos de programas que pueden ser utilizados por el V.I.M.P.

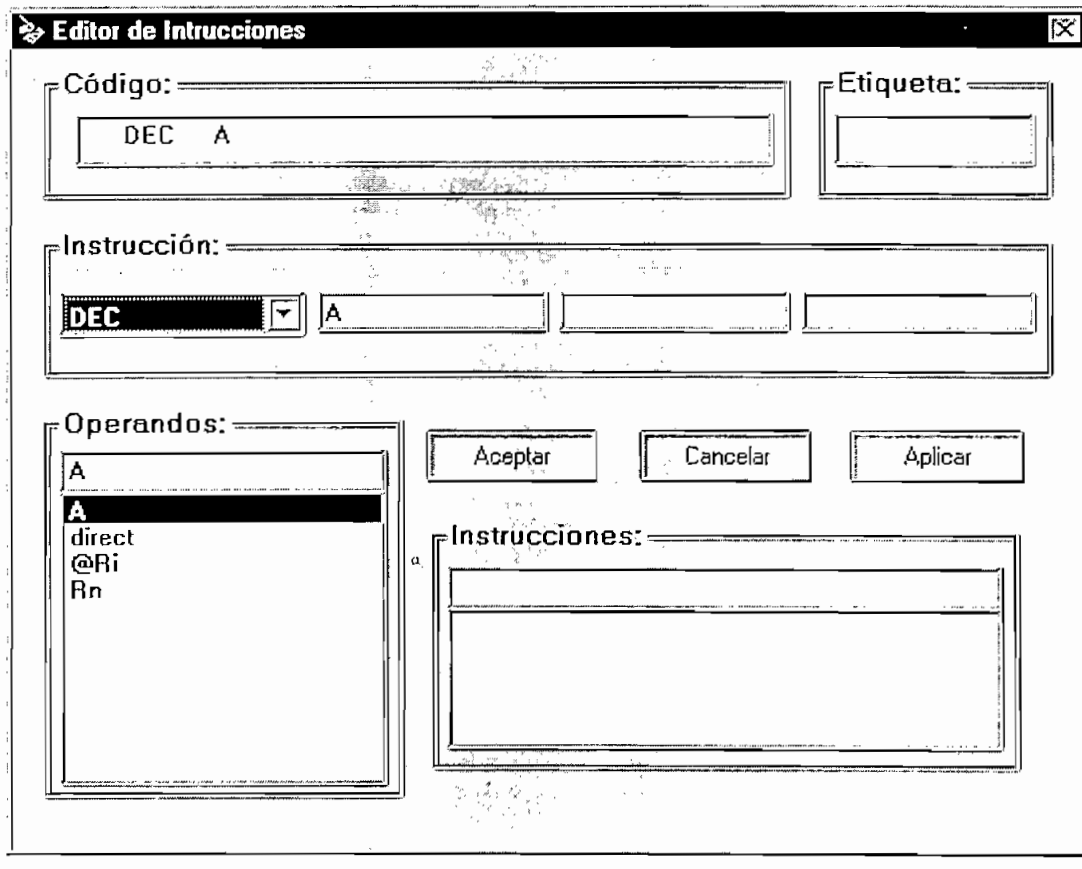


Fig.16 Editor de instrucciones

- Decodificación de RAM

Muestra una pantalla que pemite seleccionar las direcciones para los dispositivos considerados como Ram Externa.

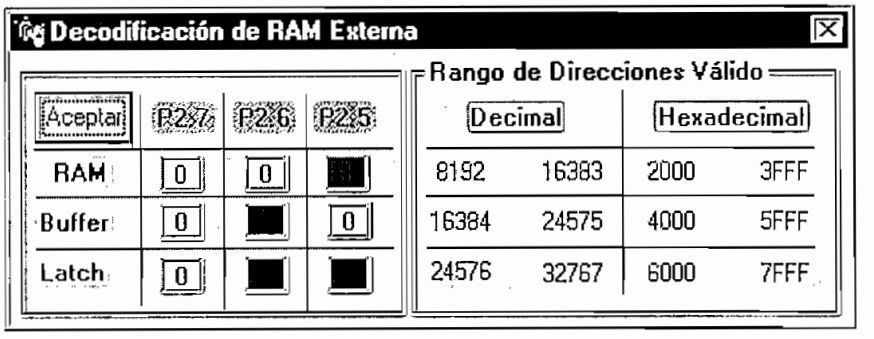


Fig.17 Pantalla para decodificación de RAM

Ayuda

Despliega las siguientes opciones :

- Manual de Usuario, permite accesar a un archivo donde se encuentra este manual.
- Pórtico Serial

  Presenta dos Opciones:

  - Iniciar

  Muestra la pantalla que permite ver la demostración del funcionamiento del Pórtico Serial en Modo 1(Fig.18).

  - Finalizar

  Abandona la pantalla de demostración del Pórtico Serial.
- Intel 8X51, muestra el manual en inglés de la Familia Intel 8051
- Acerca de V.I.M.P.

Pantalla que da información de la elaboración del Programa

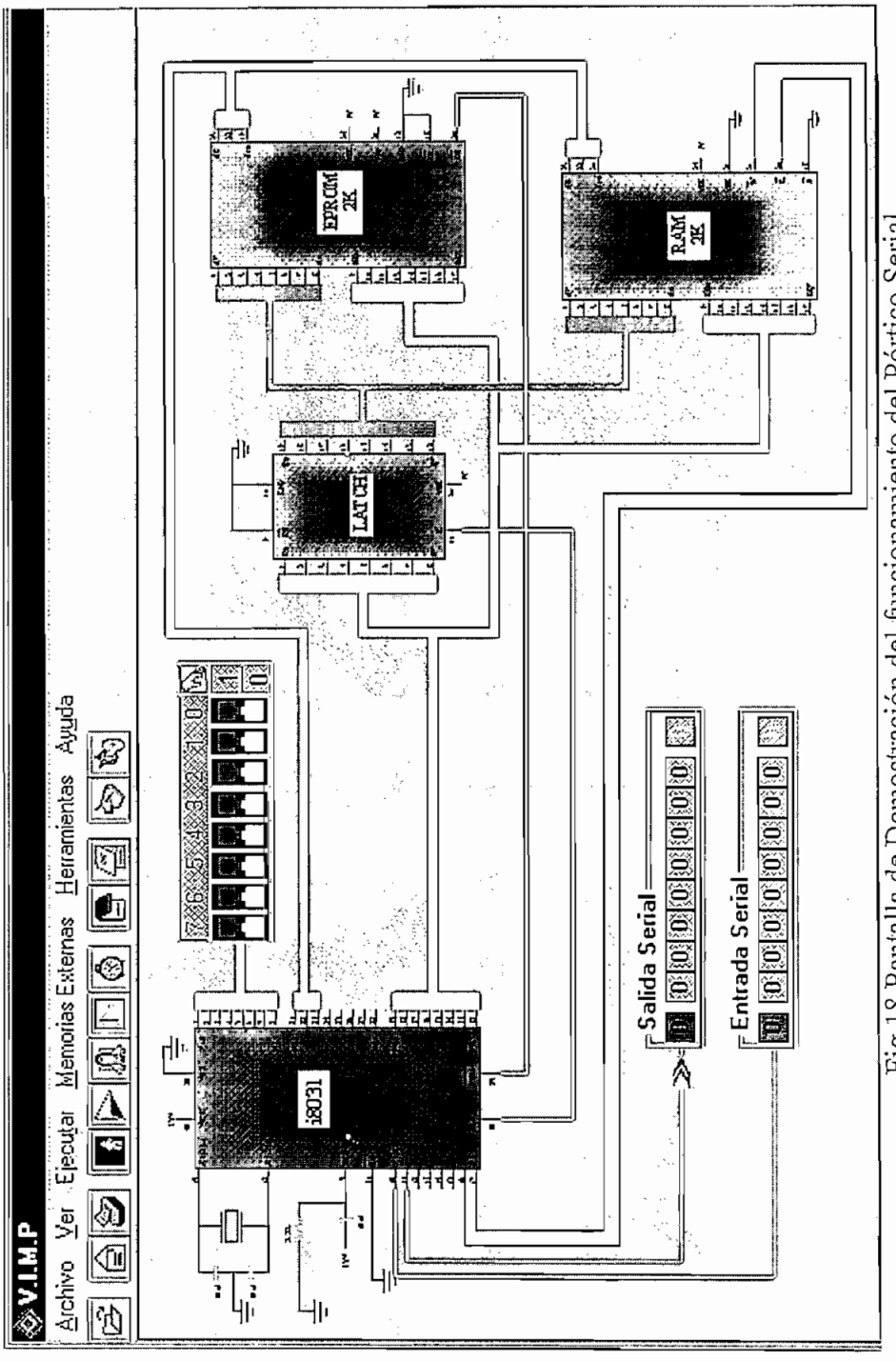


Fig.18 Pantalla de Demostración del funcionamiento del Pórtico Serial

B.4.2 Navegando a través de los Menúes de Botones.

La pantalla principal del V.I.M.P. presenta al usuario un menú de botones que permiten accesar a diferentes formularios simplemente haciendo un click en el ícono deseado. Además cada uno de ellos da información acerca de lo que hace solamente con posicionarse sobre él, como se muestra en la siguiente Figura.

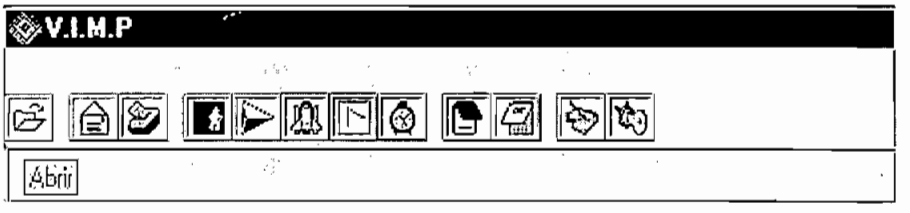


Fig.19 Menú principal de botones

- Botones INT0 , INT1 , T0 , T1.

INT0 y INT1 permiten generar interrupciones externas, mientras que T0 y T1 permiten introducir una señal de reloj externa de frecuencia no fija cuando el Timer trabaje como contador.

B.4.3 Navegando a través de los Menúes Emergentes.

Este tipo de menúes emergentes serán presentados al usuario cuando al posicionarse sobre los elementos donde el puntero del ratón cambia, presiona el botón derecho. Por ejemplo el siguiente es el menú emergente que obtenemos cuando nos posicionamos sobre la Rom y presionamos el botón derecho :

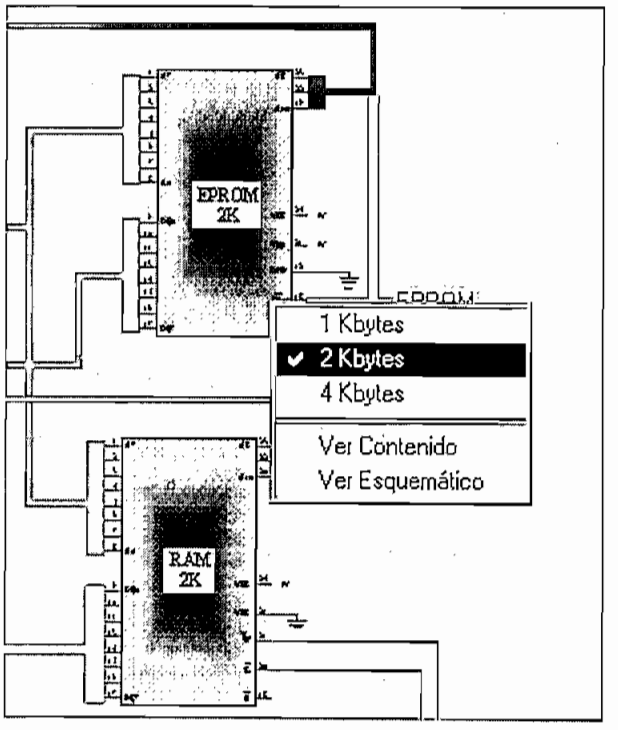


Fig.20 Menú emergente sobre la Rom

Los menúes emergentes disponibles en el V.I.M.P. son los siguientes :

- Menú emergente sobre el microcontrolador

Despliega un menú que permitirá ver tanto el esquemático del microcontrolador como el contenido de la RAM Interna.

- Menú emergente sobre el Decodificador

Despliega un menú que permitirá ver tanto el esquemático del Decodificador y la pantalla de Decodificación de la RAM Externa.

- Menú emergente sobre el Latch

Muestra la opción de ver el esquemático del Latch.

- Menú emergente sobre la Rom

Muestra un menú que permite :

Seleccionar la capacidad del dispositivo de memoria

Ver el contenido de las localidades de esta memoria

Ver el esquemático de la memoria seleccionada

- Menú emergente sobre la Ram

Muestra un menú con las mismas opciones que en la Rom.

- Menú emergente sobre el Latch377

Muestra la opción de ver el esquemático del Latch377 y además da información acerca del valor que se encuetra en la salida del mismo, tanto en decimal como en hexadecimal.

- Menú emergente sobre el Buffer

Muestra la opción de ver el esquemático del Buffer.

- Menú emergente sobre el ícono de los interruptores al Puerto P1

Despliega una pantalla que permite configura los interruptores conectados al puerto P1

- Menú emergente sobre el ícono de los interruptores al Buffer

Despliega una pantalla que permite configura los interruptores conectados al Buffer.

Las dos últimas opciones despliegan una pantalla como la siguiente :

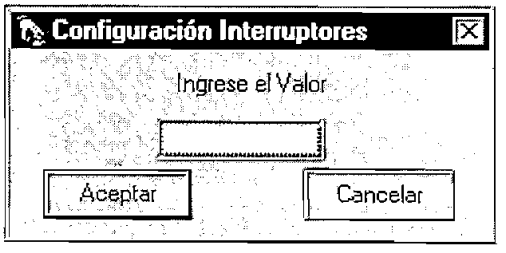


Fig.21 Pantalla de configuración de interruptores

B.5 Cómo Utilizar V.I.M.P.

Después de haber descrito la función de todos los tipo de menúes existentes en el programa de visualización, detallar la forma en que el usuario debe proceder para hacer uso del mismo es parte importante en este manual.

A continuación se detalla como el usuario debe proceder para hacer uso del V.I.M.P.

Paso 1 :

Para ingresar al programa V.I.M.P. se lo hace a través del botón Inicio de Windows, seleccionando la opción VIMP.

Paso 2:

Definir los Elementos de Memoria de los cuales quiere hacer uso, tanto de Ram como de Rom.

Paso 3:

Asignar las direcciones para cada dispositivo de RAM externa a través de la pantalla de Decodificación de RAM Externa.

Paso 4:

Dentro del V.I.M.P. a través del menú deplegable o de la botonera, seleccionar y abrir un archivo. Si el usuario quiere generar un nuevo archivo puede hacer uso del Editor de Instruciones, este archivo puede ser guardado y a continuación abierto.

Paso 5:

Verificar si el archivo abierto ha sido cargado, para esto, el usuario puede hacer uso de la opción Ver Código Detallado.

Paso 6:

Si en el programa cargado existen intrucciones que involucran a los interruptores ya sean al puerto P1 o al Buffer, el usuario puede configurar dichos interruptores en este momento, si va hacer uso de la opción de ejecutar todo el programa ; o puede hacerlo luego de la ejecución de cada instrucción si la opción es paso a paso.

Paso 7:

Luego, para ejecutar el V.I.M.P. sobre el archivo cargado, el usuario puede decidir si hacerlo con animación (se debe definir la escala de tiempo a la que el usuario pueda adaptarse) o sin animación, esta última opción es con el objetivo de obtener de una manera rápida el estado de los diferentes registros y memorias, después de cada instrucción o después de la ejecución de todo el programa.

ANEXO C

INFORMACION TECNICA

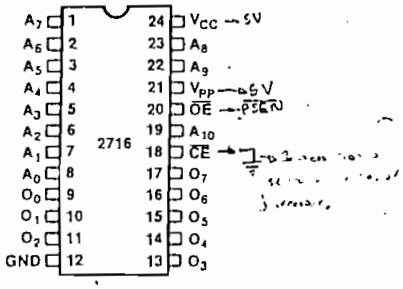
2716
16K (2K x 8) UV ERASABLE PROM

- Fast Access Time
  - — 2716-1: 350 ns Max.
  - — 2716-2: 390 ns Max.
  - — 2716: 450 ns Max.
  - — 2716-5: 490 ns Max.
  - — 2716-6: 650 ns Max.
- Single +5V Power Supply
- Low Power Dissipation
  - — Active Power: 525 mW Max.
  - — Standby Power: 132 mW Max.
- Pin Compatible to Intel 2732A EPROM
- Simple Programming Requirements
  - — Single Location Programming
  - — Programs with One 50 ms Pulse
- Inputs and Outputs TTL Compatible During Read and Program
- Completely Static

The Intel 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single-address programming. It makes designing with EPROMs fast, easy and economical.

The 2716, with its single 5-volt supply and with an access time up to 350 ns, is ideal for use with high-performance +5V microprocessors such as Intel's 8085 and 8086. Selected 2716-5s and 2716-6s are also available for slower speed applications. The 2716 also has a static standby mode which reduces power consumption without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 uses a simple and fast method for programming—a single TTL-level pulse. There is no need for high voltage pulsing because all programming controls are handled by TTL signals. Programming of any location at any time—either individually, sequentially or at random is possible with the 2716's single-address programming. Total programming time for all 16,384 bits is only 100 seconds.



PIN NAMES

A_0–A_{10}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS

Figure 1. Pin Configuration

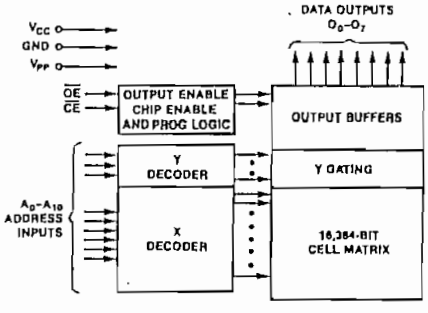


Figure 2. Block Diagram

NOVEMBER 1982
ORDER NUMBER: 210310-002

intel®

→ 39909

2708
8K (1K × 8) UV ERASABLE PRO

	Max. Power	Max. Access
2708	800mW	450ns
2708L	425mW	450ns
2708-1	800mW	350ns
2708-8	800mW	550ns

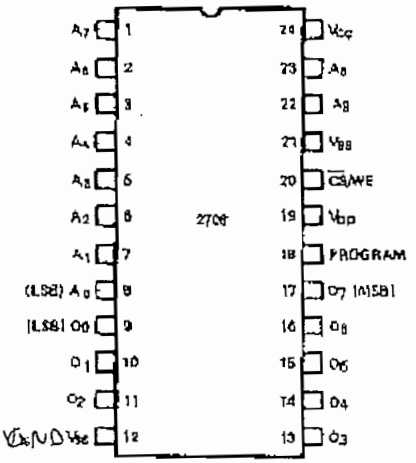
- Low Power Dissipation — 425 mW Max. (2708L)
- Fast Access Time — 350 ns Max. (2708-1)
- Static — No Clocks Required
- Data Inputs and Compatible duri Program Modes
- Three-State Outp Capability

The Intel® 2708 is an 8192-bit ultraviolet light erasable and electrically reprogrammable fast turnaround and pattern experimentation are important requirements. All data inpu patible during both the read and program modes. The outputs are three-state, allowing system bus structures.

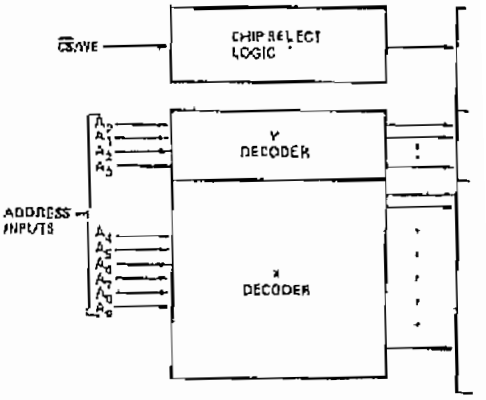
The 2708L at 425mW is available for systems requiring lower power dissipation than from savings of over 50% without any sacrifice in speed is obtained with the 2708L. The 2708L and is specified at 10% power supply tolerance. A high-speed 2708-1 is also available requiring fast access times.

The 2708 family is fabricated with the N-channel silicon gate FAMOS technology and is av package.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0-A_9	ADDRESS INPUTS

PIN CONNECTION DURING REA

	PIN NUMBER
ADDRESS	

intel

2732A
32K (4K x 8) UV ERASABLE PROM

- 200 ns (2732A-2) Maximum Access Time . . . HMOS*-E Technology
- Compatible with High-Speed 8mHz iAPX 186...Zero WAIT State
- Two Line Control
- Compatible with 12 MHz 8051 Family
- Industry Standard Pinout . . . JEDEC Approved
- Low Standby Current...30 mA Maximum
- ±10% V_{CC} Tolerance Available
- intelligent Identifier™ Mode

The Intel 2732A is a 5V only, 32,768 bit ultraviolet erasable and electrically programmable read-only-memory (EPROM). The standard 2732A access time is 250 ns with speed selection (2732A-2) available at 200 ns. The access time is compatible with high performance microprocessors such as the 8 MHz iAPX 186. In these systems, the 2732A allows the microprocessor to operate without the addition of WAIT states.

An important 2732A feature is the separate output control, Output Enable ($\overline{OE}$), from the Chip Enable control ($\overline{CE}$). The $\overline{OE}$ control eliminates bus contention in microprocessor systems. Intel's Application Note AP-72 describes the microprocessor system implementation of the $\overline{OE}$ and $\overline{CE}$ controls on Intel's EPROMs. AP-72 is available from Intel's Literature Department.

The 2732A has a standby mode which reduces power consumption without increasing access time. The maximum active current is 125 mA, while the maximum standby current is only 35 mA, a 70% saving. The standby mode is selected by applying the TTL-high signal to the $\overline{CE}$ input.

The 2732A is fabricated with HMOS*-E technology, Intel's high-speed N-channel MOS Silicon Gate Technology.

*HMOS is a patented process of Intel Corporation.

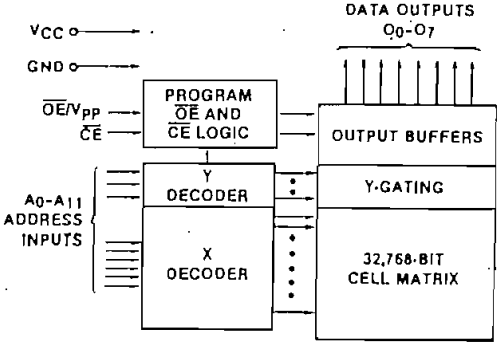


Figure 1. Block Diagram

PIN NAMES

A_0-A_{11}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}/V_{PP}$	OUTPUT ENABLE/ V_{PP}
O_0-O_7	OUTPUTS

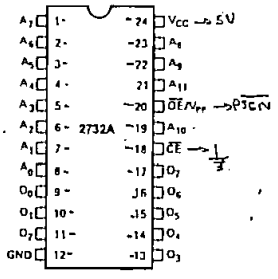


Figure 2. Pin Configuration

RAM 2K

Se usara en el programa

MOTOROLA

MCM65116

Advance Information

16K BIT STATIC RANDOM ACCESS MEMORY

The MCM65116 is a 16,384-bit Static Random Access Memory organized as 2048 words by 8-bits, fabricated using Motorola's High-performance silicon-gate CMOS (HCMOS) technology. It uses a design approach which provides the simple timing features associated with fully static memories and the reduced power associated with CMOS memories. This means low standby power without the need for clocks, nor reduced data rates due to cycle times that exceed access time.

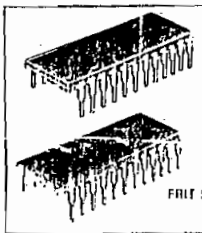
Chip Enable ($\overline{E}$) controls the power-down feature. It is not a clock but rather a chip control that affects power consumption. In less than a cycle time after chip enable ($\overline{E}$) goes high, the part automatically reduces its power requirements and remains in this low-power standby as long as the chip enable ($\overline{E}$) remains high. The automatic power-down feature causes no performance degradation.

The MCM65116 is in a 24-pin dual-in-line package with the industry standard JEDEC approved pinout and is pinout compatible with the industry standard 16K EPROM/ROM.

- Single +5 V Supply
- 2048 Words by 8-Bit Organization
- HCMOS Technology
- Fully Static: No Clock or Timing Strobe Required
- Maximum Access Time: MCM65116-12 — 120 ns
  MCM65116-15 — 150 ns
  MCM65116-20 — 200 ns
- Power Dissipation: 55 mA Maximum (Active)
  10 mA Maximum (Standby-TTL Levels)
  2 mA Maximum (Standby)
  100 μA Maximum (Standby-MCM65L116)
- Low Voltage Data Retention (MCM65L116 only) 100 μW Maximum

HCMOS
(COMPLEMENTARY MOS)

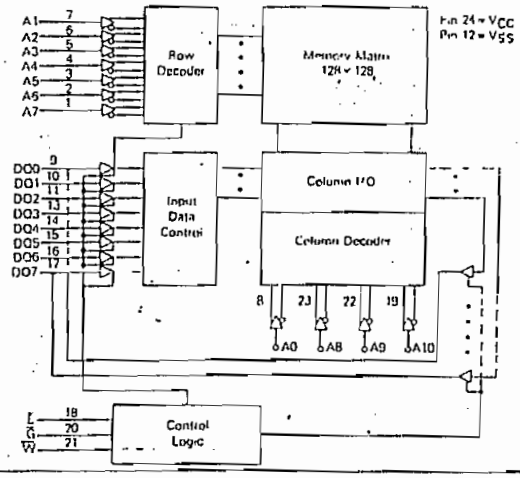
2,048 × 8 BIT
STATIC RANDOM
ACCESS MEMORY



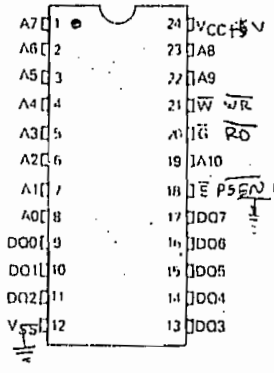
P SUFFIX
PLASTIC PACKAGE
CASE 709

C SUFFIX
FRIT SEAL CERAMIC PACKAGE
CASE 623

BLOCK DIAGRAM



PIN ASSIGNMENTS



PIN NAMES	
A0-A10	Address Input
DO0-DO7	Data Input/Output
$\overline{W}$	Write Enable
$\overline{G}$	Output Enable
$\overline{E}$	Chip Enable
V_{CC}	Power (+5 V)
V_{SS}	Ground

This document contains information on a new product. Specifications and information herein are subject to change without notice.

ADI 911/5 F2

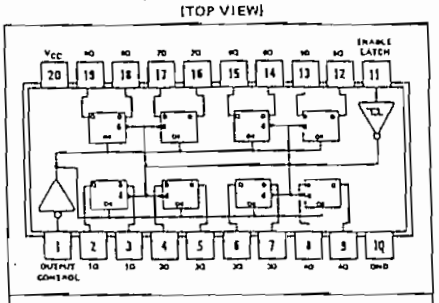
TTL
MSI

TYPES SN54LS373, SN54LS374, SN54S373, SN54S374, SN74LS373, SN74LS374, SN74S373, SN74S374 OCTAL D-TYPE TRANSPARENT LATCHES AND EDGE-TRIGGERED FLIP-FLOPS

BULLETIN NO. DL-S 12350, OCTOBER 1975 – REVISED JUNE 1979

- Choice of 8 Latches or 8 D-Type Flip-Flops In a Single Package
- 3-State Bus-Driving Outputs
- Full Parallel-Access for Loading
- Buffered Control Inputs
- Clock/Enable Input Has Hysteresis to Improve Noise Rejection
- P-N-P Inputs Reduce D-C Loading on Data Lines ('S373 and 'S374)
- SN54LS363 and SN74LS364 Are Similar But Have Higher V_{OH} For MOS Interface

SN54LS373, SN54S373 . . . J PACKAGE
SN74LS373, SN74S373 . . . J OR N PACKAGE
(TOP VIEW)

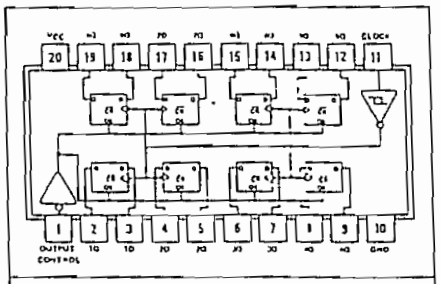


logic: see function table

'LS373, 'S373
FUNCTION TABLE

OUTPUT ENABLE	ENABLE LATCH	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

SN54LS374, SN54S374 . . . J PACKAGE
SN74LS374, SN74S374 . . . J OR N PACKAGE
(TOP VIEW)



logic: see function table

'LS374, 'S374
FUNCTION TABLE

OUTPUT ENABLE	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

See explanation of function tables on page 1-13.

description

These 8-bit registers feature totem-pole three-state outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance third state and increased high-logic-level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the 'LS373 and 'S373 are transparent D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was set up.

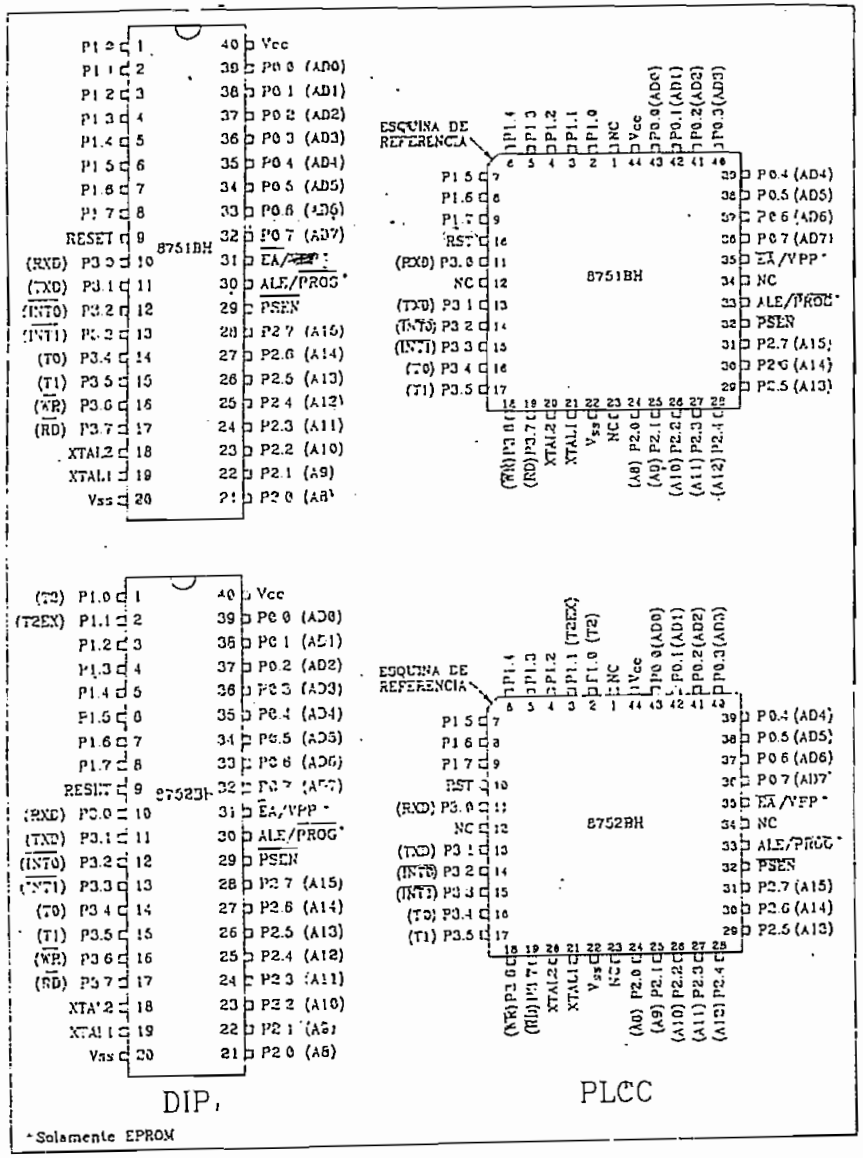


Figura 1.3.

DIP = Dual in Line Package
PLCC = Plastic Leaded Chip Carrier

PORTs o PUERTOS: Los Microcontroladores 8052/8051 tienen 4 puertos de 8 bits bidireccionales (P0. P1, P2 y P3). Esto quiere decir que pueden programarse como entrada o salida.

Como salida los *drivers* del *Puerto 0* pueden soportar una cargabilidad, es decir, un número de entradas aplicadas a sus *pines* de salida, de 8 cargas TTL-LS, el resto de los puertos sólo admiten 4 cargas TTL-LS.

Para que los puertos actúen como entradas de alta impedancia, es preciso escribir en los *latch* de cada *pin* un nivel de tensión alto (+5v), es decir un «1». Este punto se estudiará más adelante.

Además de este comportamiento como puertos de E/S para enviar o recibir información de los periféricos, éstos están capacitados para realizar las siguientes funciones:

***Puerto 0* (P0)**: Multiplexa en el tiempo por sus 8 líneas la parte baja del *bus de direcciones* durante el acceso a la memoria externa de programas y datos, y el *bus de datos*.

El *Puerto 0* también recibe los bytes de código durante la programación de la memoria EPROM integrada y salen a través de él los bytes de código durante la verificación de la memoria EPROM o ROM (Figura 1.4 A).

***Puerto 1* (P1)**: El *Puerto 1* también recibe la parte baja de direcciones, durante la programación y verificación de la memoria EPROM (Figuras 1.7 y 1.9).

Los bits P1.0 y P1.1 tienen otra función especial, sólo para el Microcontrolador 8052 (Tabla 1.2).

Tabla 1.2.*

Pines	Función alternativa
P1.0	T2 (Timer/Contador 2. Entrada externa)
P1.1	T2EX (Timer/Contador 2. Captura e impulso de recarga)

* Véase Figura 1.4 C).

***Puerto 2* (P2)**: El *Puerto 2* emite la parte alta del bus de direcciones en los accesos de memoria externa (memoria de programa) cuando utilizan 16 bits de dirección y en los accesos a la memoria de datos que usa también 16 bits de dirección (MOVX @DPTR). Durante el acceso a la memoria de datos externa con direccionamiento de 8 bits (MOVX @Ri), los pines del *Puerto 2* emiten el contenido del registro P2 del SFR (Special Function Register) (véase Figura 1.4 B).

El *Puerto 2* recibe la parte alta de la dirección, durante la programación y verificación de la memoria EPROM (Figuras 1.7 y 1.9).