

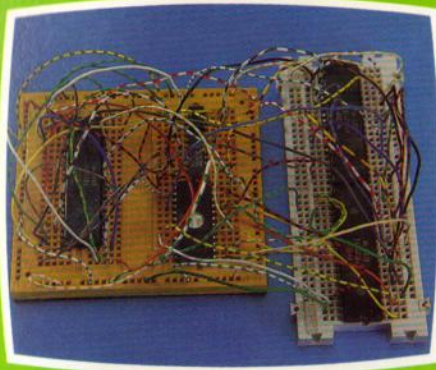
ENCICLOPEDIA PRACTICA DE LA
INFORMATICA
APLICADA

15

**Cómo construir
su propio
ordenador**

AIA

INCLUIE
AMSTRAD
PROGRAMADOR



Cómo construir su propio ordenador



Cuando se trabaja con un ordenador, lo único que puede apreciarse, a simple vista, es una especie de caja negra que, misteriosamente, acepta una serie de instrucciones.

En realidad, un ordenador es una máquina capaz de recibir, transformar, almacenar y suministrar datos. En este libro se construye un pequeño ordenador, capaz de realizar dichas funciones, a partir de un microprocesador. Podrá observar de qué partes se compone el ordenador y cómo conectarlas entre sí. Además, podrá adentrarse en el mundo del «hardware» utilizándolo, por ejemplo, como un temporizador para controlar cualquier electrodoméstico casero.



Por otra parte, se ha pensado en la construcción de un ordenador dinámico. A partir del diseño básico, podrá ampliarlo con el único límite que impone la propia imaginación.

EDICIONES SIGLO CULTURAL

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Directores de la colección:

MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación
y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:

BRAVO-LOFISH.

Dibujos:

JOSE OCHOA Y ANTONIO PERERA.

Tomo XV. **Cómo construir su propio ordenador**

AULA DE INFORMATICA APLICADA

JOAQUIN SALVACHUA, Diplomado de Telecomunicación

JOSE LUIS TODO, Diplomado de Telecomunicación

FERNANDO SUERO, Diplomado de Telecomunicación

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:

Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:

Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-053-7.

ISBN de la obra: 84-7688-018-9.

Fotocomposición:

ARTECOMP, S.A. Albaracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. Pinto (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-1132-1987.

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S.A.

Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid

Octubre, 1986.

P.V.P. Canarias: 365.

I N D I C E

1	Partes del ordenador	5
2	El microprocesador	9
3	La memoria	25
4	La entrada-salida	31
5	Software	57
6	Aplicaciones	61
	Apéndice A: Sistemas de numeración y su representación en el ordenador	65
	Apéndice B: Consejos a la hora de montar el circuito	71
	Apéndice C: Fuente de alimentación	75
	Apéndice D: Listado fuente del programa monitor	77
	Apéndice E: Listado hexadecimial del programa monitor	91
	Apéndice F: Códigos de programación	97
	Apéndice G: Patillaje de circuitos integrados	107
	Apéndice H: Bibliografía	111

V

AMOS a ver, paso a paso, cómo funciona un ordenador; de qué partes se compone y cómo se conectan entre sí. A partir de estas ideas generales veamos cómo llevarlas a la práctica con la construcción de un pequeño, pero versátil, microcomputador.

Todo ordenador tiene dos partes bien diferenciadas: el hardware y el software. El hardware es el conjunto de circuitos que componen físicamente el ordenador. El software son los programas e instrucciones que indican a los circuitos qué funciones deberán realizar.

En este capítulo describiremos brevemente el hardware. Los ordenadores tienen tres partes básicas:

- La unidad central de procesos (CPU) o microprocesador.
- La memoria.
- La unidad de entrada-salida (I/O)*.

La CPU es el «cerebro» del ordenador. Es la que controla a las demás partes y la que realiza las operaciones.

La memoria es un almacén donde están guardados los datos y las instrucciones que la CPU debe utilizar y en ella almacena los resultados.

Con esto parece que tenemos todo lo necesario, pero no es así, ya que para que los resultados de las operaciones puedan ser útiles es necesario comunicarse con el exterior. De esta función se encarga la unidad de I/O. Se encarga de transmitir datos del ordenador al exterior (pantalla del ordenador, impresora, etc.) y del exterior al ordenador (teclado, cinta casette, etc.).

* Las iniciales corresponden a las palabras en inglés. (CPU = Central Processing Unit; I/O = Input/Output). A partir de ahora todas las iniciales se referirán a su escritura en inglés, ya que esta forma es la comúnmente aceptada.

Los programas que aparecen en este libro funcionan en los ordenadores:

- IBM-PC, XT, AT y compatibles.
- AMSTRAD-464, 664, 6128, 1512.
- SINCLAIR-SPECTRUM 48 K, 128 K, PLUS, PLUS 2.
- MSX-Todos los modelos.
- COMMODORE-CBM 64 y CBM 128.

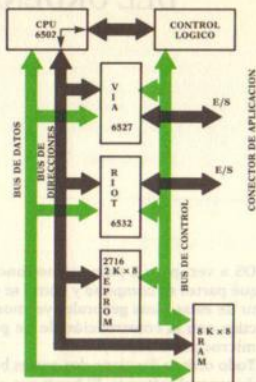


Fig. 1.1. Diagrama de bloques del ordenador.

El ordenador sólo trabaja con datos que le son comprensibles. Estos son bits, o dígitos binarios (para mayor información, ver el apéndice A). Cada bit sólo puede ser 0 u 1 o un 0 u 1. Pero como el ordenador no trata con números, esto se representará como 0 voltios, cuando se quiera representar un 0, y como 5 voltios cuando se quiera representar un 1. A los grupos de 8 bits se les conoce como bytes. Los microprocesadores llamados «de 8 bits» sólo pueden procesar al mismo tiempo un byte.

Como se observa en la figura 1.1, en el ordenador cada bloque está relacionado con los demás mediante una serie de líneas, o cables, conocidos como buses.

Existen tres buses: el de control, el de datos y el de direcciones.

El bus de control es el que se encarga de llevar desde la CPU a los demás bloques las instrucciones que éstos deben realizar.

El bus de datos es el que lleva la información, o datos, que los distintos bloques se comunican. El ancho de este bus (número de bits que se corresponde con el número de cables) es el de ancho de palabra = 1 byte.

El bus de direcciones da la dirección, como su nombre indica, adonde debe llevarse el dato. Está organizado como las direcciones de correos.

Por ejemplo, para que la CPU almacene un dato en la memoria debe enviar por el bus de control la señal de que quiere guardar el dato en la

memoria, el dato por el bus de datos, y en qué parte de la memoria quiere almacenarlo por el bus de direcciones. Del ancho del bus de direcciones dependerá el máximo de memoria que el ordenador puede utilizar.

Esta organización, o arquitectura, se conoce como arquitectura von Neumann, en honor del primero al que se le ocurrió. John von Neumann la aplica en los años cuarenta a la construcción de los primeros ordenadores, y desde entonces no se han realizado realmente diferentes. Actual-

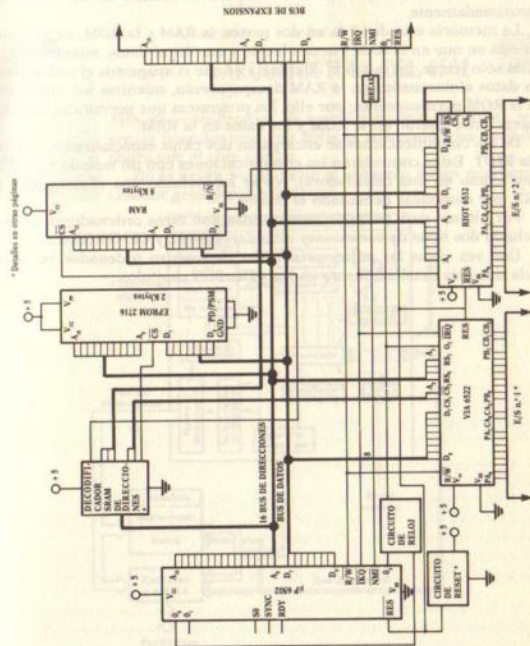


Fig. 1.2. Esquema general del ordenador.

mente se intenta buscar alguna que permita a varias CPUs realizar tareas cooperando entre ellas, pero no se ha encontrado ninguna de utilidad comparable.

Veamos ahora cómo se realizan estas ideas generales en la realización de nuestro ordenador.

Como CPU vamos a usar el microprocesador 6502. Este es un «chip», o circuito integrado, que lleva todo lo necesario para componer una CPU. Hasta hace poco tiempo una CPU ocupaba el tamaño de un televisor, aproximadamente.

La memoria está dividida en dos partes: la RAM y la ROM. La diferencia está en que en la RAM pueden leerse y escribirse datos, mientras en la ROM sólo leerse. La segunda diferencia es que si apagamos el ordenador, los datos almacenados en la RAM desaparecerán, mientras los existentes en la ROM permanecerán; por ello, los programas que permitirán realizar operaciones estarán en la ROM y los datos en la RAM.

De las comunicaciones se encargarán dos chips especializados, la VIA y la RIOT. Estos controlarán las comunicaciones con un teclado y un display (como en una calculadora). No se ha incluido una «Televisión», o CRT, por complicar demasiado el diseño.

Por último, para permitir comunicarse con otros ordenadores se han incluido dos tipos de conexiones estándar: RS-232 y CENTRONICS.

Una vez vistas las partes principales de nuestro ordenador veremos cada una más detalladamente en los siguientes capítulos.

ARQUITECTURA DEL 6502

ADA microprocesador, de marca y modelo diferentes, posee un lenguaje máquina distinto, adecuado a las posibilidades que el chip ofrece.

En la figura puede verse el esquema de un microprocesador genérico.

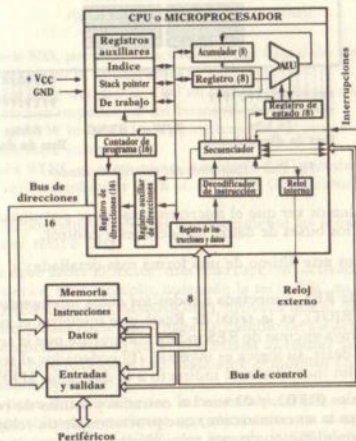


Fig. 2.1. Esquema del ordenador.

En nuestro ordenador utilizaremos el microprocesador 6502, de la marca Rockwell.

Este es un microprocesador de 8 bits, es decir, su bus de datos puede manejar simultáneamente palabras de 8 bits. Puede manejar hasta 64 Kbytes de memoria (su bus de direcciones es de 16 bits: 2^{16} bits = 65536 bytes = 64 Kbytes). Y la frecuencia de reloj es de 1 MHz (en la versión que utilizaremos). En la siguiente gráfica podemos observar la estructura interna del microprocesador 6502 desde el punto de vista del programador (arquitectura software).

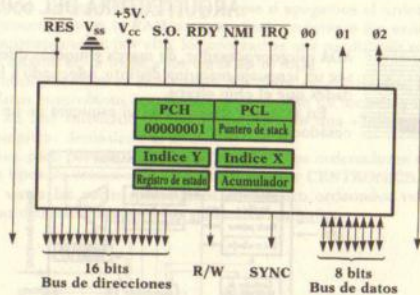


Fig. 2.2. Esquema del microprocesador.

En ella podemos ver que el microprocesador se comunica con el exterior mediante los buses de datos, direcciones y control.

Describamos este último de una forma más detallada:

— La patilla RES, conectada a todos los chips inteligentes (microprocesador, VIA, RIOT), es la señal de Reset que inicializa el sistema al encenderse. (La raya encima de RES indica claramente que la señal es activa a nivel bajo es decir, su lógica es inversa.) (El ordenador al recibir esta señal hace un salto incondicional indirecto a la dirección \$FFFF \$FFFD.)

— Las señales 01, 02, y 03 son las entradas y salidas de reloj. 01 y 02 son salidas (para la sincronización) que provienen de un reloj interno del chip. Para la estabilización de este reloj interno es necesaria una señal de reloj exterior conectada a 00, según uno de los esquemas de la figura.

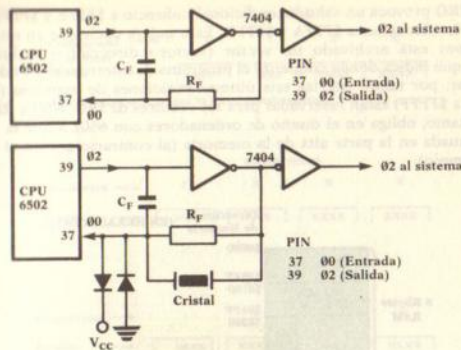


Fig. 2.3. Esquemas posibles de reloj.

- La línea S.O. permite la activación del flag V de sobrepasamiento desde el exterior (no la utilizaremos).
- La línea RDY detiene la CPU en todos los ciclos, excepto en escritura, y deja en alta impedancia los buses (no la usamos).
- La línea R/W indica a los dispositivos si el acceso es de lectura (nivel alto) o escritura (nivel bajo).
- La línea SYNC se pone a uno cuando la CPU accede a un código de operación (ver ensamblador para «Código de Operación»). (Tampoco la usamos.)
- Por último, estudiemos más detalladamente las señales de interrupción; éstas son: NMI e IRQ.

Como ya debe saber el lector, una interrupción, activada por una señal hardware exterior (por ejemplo, pulsando la tecla ESC en el AMSTRAD o en el PC o BREAK en el COMMODORE), provoca que el microprocesador deje la tarea que está haciendo y haga un cambio de contexto, es decir, archive en memoria el estado en el que estaba al recibir la interrupción y pase a ejecutar otro programa (normalmente el monitor en ROM u otra rutina del Sistema Operativo).

Hay dos tipos de interrupción; la mascarable (IRQ), que puede ser inhabilitada por software (mediante una instrucción en máquina que la desactiva) y la no mascarable (NMI), que fuerza a la CPU a abandonarlo todo (suele usarse para abortar situaciones catastróficas = es un BREAK «salvaje»).

La IRQ provoca un salto incondicional indirecto a \$FFFE y \$FFFF y la NMI a las direcciones \$FFFA y \$FFFB. Esto quiere decir que en estas direcciones está archivado un vector (vector = dirección = 16 bits = 2 bytes), que indica dónde comienza el programa de interrupción. Podemos observar, por tanto, que las seis últimas posiciones de memoria (de la \$FFFA a \$FFFF) están reservadas para los vectores de IRQ, NMI y RESET y, por tanto, obliga en el diseño de ordenadores con 6502 a que la ROM vaya situada en la parte alta de la memoria (al contrario que en el Z-80, por ejemplo).

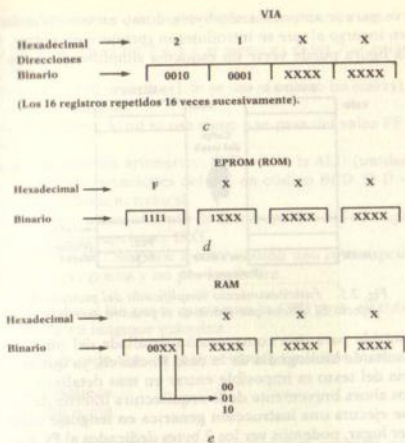
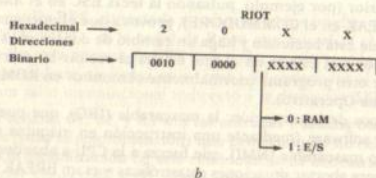
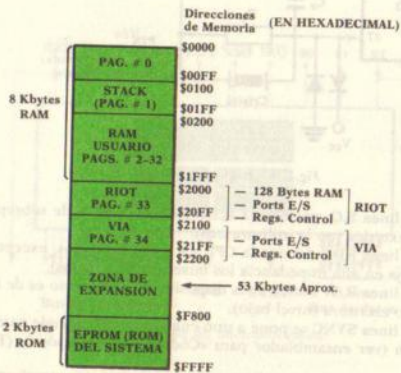


Fig. 2.4. Mapa de memoria.

Veamos ahora más detalladamente la arquitectura software del 6502. Además de las seis posiciones anteriormente citadas, hay dos páginas de memoria (una página = 256 bytes = de \$XX00 a \$XXFF), que son usadas de forma especial por el 6502.

La página cero (\$0000 a \$00FF) es una zona de memoria de acceso rápido (ya que se ahorra un ciclo de máquina cada vez que se usa), por lo que suele utilizarse para almacenar en ella variables o datos de uso frecuente (normalmente las variables internas del sistema). La página 1 (\$0100 a \$01FF) es la zona reservada al STACK o pila. ¿Qué es la pila? Intentaremos explicarlo brevemente.

Cuando en el programa máquina hay que hacer un cambio de contexto (un salto a subrutina o una interrupción), hace falta archivar el estado de la CPU o enviar una serie de parámetros, para lo que se necesita una zona de memoria de acceso rápido en la que almacenar en un orden preestablecido esta información.

La pila ofrece un método muy sencillo mediante el puntero del stack (uno de los registros de la CPU accesible por software en lenguaje máquina).

Esta sirve para ir amontonando bytes, como en un «montón» y sacarlos en orden inverso al que se introdujeron (primero en entrar, último en salir). En la figura puede verse un esquema simplificado de su funcionamiento.

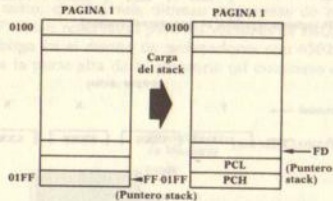


Fig. 2.5. Funcionamiento simplificado del puntero de stack. El último que entra, es el primero que sale.

Es aconsejable para una comprensión detallada del microprocesador 6502 la consulta de bibliografía de la casa Rockwell, ya que por la brevedad necesaria del texto es imposible entrar en más detalles.

Hablemos ahora brevemente de la arquitectura interna de los registros y de cómo se ejecuta una instrucción genérica en lenguaje máquina.

En primer lugar, podemos ver los 2 bytes dedicados al PC (contador de programa - Program Counter); en él está almacenada la posición de memoria en la que está el byte que estamos ejecutando.

El puntero del stack (en realidad, un solo byte, ya que el otro es fijo e indica la página 1) contiene la dirección del último byte almacenado en la pila.

Los registros de índice X e Y se utilizan en las operaciones en las que se necesitan valores de índice o desplazamiento relativo y son de 8 bits (son registros versátiles, pero no de uso general).

El acumulador es el registro primordial de la CPU. En él se realizan todas las operaciones aritméticas y lógicas.

Por último, el registro de estado es un registro de 8 bits, 7 de los cuales son flags (banderas o indicadores) que nos indican cuál es el estado de la CPU en cada momento. En la figura puede verse la colocación de cada indicador.

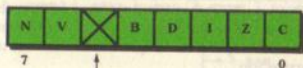


Fig. 2.6. Estructura de los bits del registro de Estado, que actúan como señalizadores o alarmas (flags).

A continuación damos una breve explicación de cada uno:

N: Flag de signo: si el bit 7 = 1 y el signo es el bit 7 \rightarrow valor negativo en el Ac, y $N = 1$.

V: Sobrepasamiento (overflow): Si se usa el octavo bit (carry) como signo e igual a 1 \rightarrow overflow.

C: Señal de acarreo: Si no se usa signo y se pasa del valor FF en el acumulador el $C = 1$.

D: Flag de tratamiento aritmético: Si $D = 1 \rightarrow$ la ALU (unidad aritmético-lógica) realiza las operaciones del AC en código BCD. Si $D = 0 \rightarrow$ operaciones en código binario natural.

I: Bandera de enmascaramiento de interrupción: Si $I = 1$ implica que la CPU no aceptará interrupciones IRQ.

B: Señal de BREAK: Se pone a uno cuando una interrupción IRQ ha sido activada por programa y no por hardware.

Describamos ahora los pasos que sigue la CPU para ejecutar una instrucción genérica en lenguaje máquina.

La ejecución de todo tipo de instrucciones tiene dos fases:

- Fase de búsqueda.
- Fase de ejecución.

La primera, común para todas las instrucciones, se inicia cuando en el PC está la dirección del código de la próxima instrucción a ejecutar. Esta dirección se pone en el bus de direcciones y se activa la patilla R/W (con 0), con lo que la memoria (ROM o RAM) pone el contenido de la dirección en el bus de datos.

Del bus de datos la CPU lee el código y lo interpreta finalizando la primera fase (ver figura 2.7).

MEMORIA DE PROGRAMA	
Dirección	Contenido
n	Código operación AND
n + 1	M_n
n + 2	M_{n+1}
MEMORIA DATOS	
$M_n - M_n$	2.* operando

← 8 bits de menos peso de la dirección de M

← 8 bits de más peso de la dirección de M

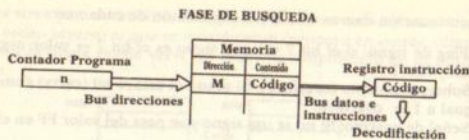


Fig. 2.7. Ejecución de una instrucción.

La fase de ejecución depende de cada instrucción. Si ésta necesita un dato, lo leerá de la memoria de forma similar a como leyó el código, almacenándolo en algún registro o usándolo según el caso. Si este dato es una dirección (2 bytes), es obvio que serán necesarios dos accesos sucesivos a memoria (ver figura).

En el apéndice se pueden ver detalladas tablas que describen todas las instrucciones, su tiempo de ejecución y otros datos de interés. Haremos una descripción breve de ellas en la parte dedicada al ensamblador.

BUSES Y CRONOGRAMAS

En nuestro ordenador todas las operaciones se realizan sincronamente; esto quiere decir que todas las partes implicadas en una operación actúan al mismo tiempo. Es igual que una orquesta en la que todos los músicos deben dar una misma nota al mismo tiempo, no esperan a oír la del compañero, ya que sería fatal. Al igual que los músicos se «sincronizan» al director de orquesta en el microprocesador todos los sistemas se sincronizan con el reloj. Reloj es una señal periódica, conocida como 01.

Existe otra señal 02 que es la inversa de 01; cuando 01 está en estado alto, 02 está en estado bajo, y viceversa.

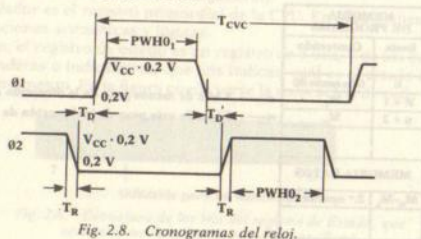


Fig. 2.8. Cronogramas del reloj.

La frecuencia de esta señal, las veces que se repite cada segundo, nos da la rapidez con que el ordenador realiza sus operaciones. En nuestro caso será de 1 MHz (un millón de veces por segundo). Aunque pueda parecer mucho, con la tecnología actual no lo es, pues existen otros microprocesadores que funcionan a 8 MHz e incluso a 16 MHz.

Como todas las señales están referidas al reloj, se suelen dibujar con respecto a 01 o 02. A partir de ahora utilizaremos algunos de estos dibujos para explicar las distintas señales de control.

El bus de direcciones está formado por 16 líneas conocidas como A1, A2, ..., A15 (la A viene de address, dirección en inglés).

El bus de datos estará formado por ocho líneas conocidas como D0, D01, ..., D7 (la D viene de Data, es decir, dato).

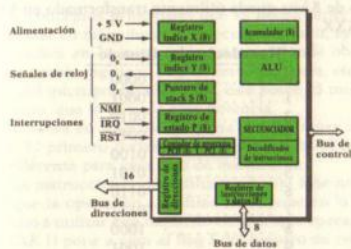


Fig. 2.9. Esquema Riot.

Los cronogramas de estos dos buses se verán en el capítulo siguiente al hablar de la memoria.

Cuando un usuario programa un ordenador, normalmente lo hace en un lenguaje de «alto nivel», es decir, un lenguaje más cercano al nivel humano.

Este tipo de lenguajes (como el BASIC, PASCAL, COBOL, FORTRAN, etc.) resulta incomprensible por el hardware del ordenador (circuitaría), que sólo entiende un tipo de lenguaje: el lenguaje máquina (o código objeto). Este lenguaje máquina es una ristra de unos y ceros (pulsos altos y bajos de tensión), y es un lenguaje eléctrico.

Por eso cuando alguien trabaja en alto nivel, está utilizando un programa que traduce su lenguaje al lenguaje máquina.

El lenguaje máquina permite hacer cosas que desde alto nivel son imposibles, ya que se están controlando los recursos del ordenador a su nivel más detallado.

En contrapartida, el lenguaje máquina es prácticamente inutilizable. (Una cantidad semejante de unos y ceros volvería loco a cualquiera.)

Por ello, ya hace tiempo se creó el lenguaje ensamblador. Este lenguaje es una traducción, palabra a palabra, del lenguaje máquina; por ello, el programa que ensambla lo único que hace (en principio) es traducir una serie de códigos y números (nemetónicos o nemónicos) a una ristra de dígitos binarios.

En el 6502 la palabra de datos tiene un ancho de 8 bits, por lo que en principio sería posible la existencia de $2^8 = 256$ instrucciones, aunque, en realidad, algunos códigos no se utilizan. Un primer paso para la utilización del lenguaje máquina es la comprensión del código hexadecimal, que permite representar números binarios de un número de bits múltiplo de 4 de forma directa y sin operaciones aritméticas intermedias (ver tabla), con lo que un número de 8 bits queda útilmente transformado en $\$XX$ y una dirección en $\$XXXX$.

Hexadecimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Para mayor información sobre el sistema de numeración binario, ver el apéndice.

Aunque la numeración hexadecimal simplifica enormemente la programación en máquina, carece de un sentido «natural» de número para el hombre (por ello, los ensambladores comerciales permiten la utilización de varios sistemas de numeración: hexadecimal, decimal, octal y binario).

Aunque los códigos podrían escribirse en el programa en su forma numérica, la principal ventaja del ensamblador es que traduce una palabra nemotécnica a su número correspondiente. Por eso en lenguaje ensamblador se utilizan como códigos de operación abreviaturas (del inglés) que in-

dican qué es lo que hace la instrucción. Esto (lo del inglés) obliga al programador en máquina a tener una leve idea sobre lo que algunas palabras en inglés significan. Por ejemplo, la instrucción LDA (Load Acumulador) significa «Carga en el acumulador el dato...» LDA es traducido por el ensamblador al número binario 1010.1001 (ver figura)

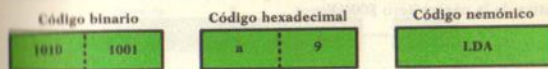


Fig. 2.10. Tres formas de expresar la instrucción de «Carga del Acumulador» mostrando la evolución del lenguaje máquina.

Uno de los conceptos principales que se deben tener en cuenta a la hora de programar en máquina es el direccionamiento, es decir, la forma en la que se indica en la instrucción cómo y dónde obtener los datos para ejecutarla (stack, registros, página cero, memoria, etc.).

En el caso del microprocesador 6502, éste posee 13 modos posibles de direccionamiento, que lo dota de gran potencia.

Las instrucciones en lenguaje máquina pueden tener una longitud de 1, 2 ó 3 bytes. El primero corresponde en todas ellas al código de operación, que es diferente para cada tipo de instrucción.

Por ello, las instrucciones que utilizan un solo byte no poseen mayor información que la operación a realizar (se supone en la mayoría de los casos que el dato a utilizar está implícito en la propia operación). Por ejemplo, CLI (CLEAR I) pone a cero el flag I del registro de estado.

En las instrucciones con 2 bytes, el segundo corresponde a un número de 8 bits (que puede ser un dato o una dirección de página cero $\$00XX$, de Stack $\$01XX$, etc.).

En los de 3 bytes, los dos últimos suelen ser una dirección (16 bits = 2 bytes).

Los 13 modos de direccionamiento son los siguientes:

1) Inmediato (2 bytes): El segundo byte indica el dato a utilizar (el primero es el código de operación).

Ejemplo

Código OP/Operando/Expresión en ensamblador/Expresión

A9 00 LDA # 00 (A) ← 00

2) Absoluto (3 bytes): Los dos últimos bytes indican la dirección del dato a utilizar.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
AD (2136) LDA 2136 (A)--(2136)

3) Por página cero (2 bytes): El segundo byte indica una dirección relativa de la página cero \$00XX.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
A5 (00,61) LDA61 (A)--(00,61)

4) Direccionamiento por página cero, Índice X (Y) (2 bytes): La dirección del operando (que está en la página cero) se halla sumando el segundo byte de la instrucción al contenido del registro X(Y) de la CPU.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
B5 (00,38+X) LDA 38,X (A)--(00,38+X)

6) y 7) Absoluto índice X(Y) (3 bytes): La dirección donde está el dato a utilizar se halla sumando a la dirección de los dos últimos bytes de la instrucción el contenido del registro X(Y).

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
BD (22,33+X) LDA 2233,X (A)--(22,33+X)

8) Direccionamiento por acumulador (1 byte): En el acumulador se encuentra el dato a utilizar.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
6A ROR Acumulador Ver fig. 2.11

9) Implicada (1 byte): El operando va implicado en la propia instrucción.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
18 Carry CLC C←0

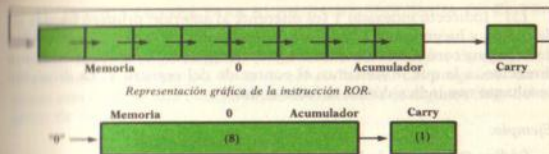


Fig. 2.11. Representación gráfica de la instrucción LSR.

10) Relativo (2 bytes): El segundo byte de la instrucción es un valor de offset que se añade al contador de programa para provocar una bifurcación. Esta es condicional si es necesaria alguna condición (del registro de estado) para que se realice.

Ejemplo:

Código OP/Operando/Expresión en ensamblador/Expresión
F0 13 BEQ13 (PC)--(PC)+13

11) Indirecto (3 bytes): Los dos últimos bytes proporcionan una dirección. De ella y de la siguiente obtenemos otra, que es la que indica dónde está el dato (esta última dirección se suele llamar vector).

Ejemplo:

Código OP/Operando/Ensamblador/Expresión
6C 3128 JMP(3128) (PC)--((3128) Y (3129))

12) Indexado X indirecto (primero indexa y luego indirecta) (2 bytes): El segundo byte se añade al registro X y el resultado es una dirección de la página cero. Con su contenido y el de la siguiente obtenemos (el vector) dirección del dato.

Ejemplo:

Código OP/Operando/Ensamblador/Expresión
A1 30 LDA(30,X) (A)--((0030)+X)

NOTA: Cuando se escribe (---) indica el contenido de lo que hay dentro del paréntesis. Por ejemplo (\$30F1), es el número \$XX contenido en la dirección \$30F1.

13) Indirecto indexado Y (es diferente al anterior; primero hace la dirección y luego indexa) (2 bytes): El segundo byte indica una dirección de la página cero. Con su contenido y el del siguiente byte obtenemos otra dirección, a la que le sumamos el contenido del registro Y. La dirección resultante nos indica dónde se halla el dato.

Ejemplo:

Código OP/Operando/Ensamblador/Expresión

B1 32 LDA (32),Y (A)---(((0032)(0033))+Y)

Veamos ahora un pequeño resumen en el que se indica brevemente la función de las principales instrucciones del 6502:

1) GRUPO DE INSTRUCCIONES ARITMETICAS Y LOGICAS

Realizan operaciones aritméticas o lógicas:

ADC	Suma aritmética
SBC	Substracción
AND	Operación lógica AND
EOR	OR exclusiva
ORA	Operación OR

2) INSTRUCCIONES DE LECTURA Y ESCRITURA EN MEMORIA

Transferen información entre la memoria y los registros:

LDA	Cargar acumulador
LDX	Cargar X
LDY	Cargar Y
STA	Almacenar el acumulador en memoria
STX	Almacenar el registro X
STY	Almacenar Y

3) INSTRUCCIONES DE TRANSFERENCIA ENTRE REGISTROS

Intercambian la información entre registros:

TAX	Transfiere el acumulador a X
TAY	Transfiere el acumulador a Y
TXA	Transfiere X al acumulador
TYA	Transfiere Y al acumulador
TSX	Transfiere el puntero del stack a X
TXS	Transfiere el registro X al puntero del stack

4) INSTRUCCIONES PARA RUPTURA DE SECUENCIA EN EL PROGRAMA

Estas instrucciones cambian el contenido del contador de programa, lo que implica una ruptura de la secuencia normal del programa. En algunos esta ruptura está condicionada al contenido de algunos flags del registro de estado.

JMP	Salto incondicional
BCC	Bifurcación si C = 0
BCS	Bifurcación si C = 1
BEQ	Bifurcación si Z = 1
BNE	Bifurcación si Z = 0
BMI	Bifurcación si N = 1
BPL	Bifurcación si N = 0
BVC	Bifurcación si V = 0
BVS	Bifurcación si V = 1

5) INSTRUCCIONES DE USO DEL STACK

Manejan el puntero de stack:

PHA	Mete al acumulador del stack
PHP	Mete al registro de estado del stack
PLA	Saca al acumulador del stack
PLP	Saca al registro de estado del stack

6) INSTRUCCIONES DE TRATAMIENTOS DE SUBROUTINA

Para el uso de subrutinas e interrupciones:

JSR	Salto a subrutina
RTS	Retorno de subrutina
BRX	Provoca IRQ (interrupción mascarable) por software
RTI	Retorno de rutina de interrupción

7) INSTRUCCIONES DE CAMBIO DE LOS «FLAGS» DEL REGISTRO DE ESTADO

Ponen dichos flags a cero o unos, según el caso:

CLC	C = 0
CLD	D = 0
CLI	I = 0
CLV	V = 0
SEC	C = 1
SED	D = 1
SEI	I = 1

8) INSTRUCCIONES DE INCREMENTO Y DECREMENTO

Afectan a memoria o a registros:

DEC	Decrementa en 1 el contenido de memoria
DEX	Decrementa en 1 el registro X
DEY	Decrementa en 1 el registro Y
INC	Incrementa memoria
INX	Incrementa el registro X
INY	Incrementa el registro Y

9) INSTRUCCIONES DE TRASLADO Y ROTACION DE BITS

Permiten desplazar o rotar el acumulador (interviniendo el carry) o la memoria:

ASL	Desplaza hacia la izquierda el acumulador o memoria
LSR	Desplaza hacia la derecha el acumulador o memoria

(ver figuras)

ROR	Desplazamiento cíclico a la derecha del acumulador o memoria
ROL	Desplazamiento cíclico a la izquierda del acumulador o memoria

10) INSTRUCCIONES DE COMPARACION LOGICA O ARITMETICA

AND	Operación lógica AND
CMP	Compara (resta a A el contenido de M) sin afectar al acumulador. Sólo afecta al registro de estado (N Z y C)
CPX	X-M
CPY	Y-M
BIT	A AND M sólo afecta a los flags

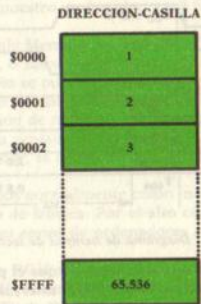
11) INSTRUCCIONES ESPECIALES

NOP	No hace nada (pierde dos ciclos de máquina)
-----	---



COMO vimos en el capítulo 1, la memoria es una de las partes principales del ordenador, ya que permite tener los programas a ejecutar y los datos a usar.

Desde el punto de vista de la CPU la memoria es una serie de «casillas», cada una con una dirección, en las que se pueden leer o guardar datos. Estas direcciones son un número, siendo estas casillas consecutivas y refiriendo cada número, o dirección, a una y sólo una de las casillas.



Esquema de la memoria.

Fig. 3.1. Esquema de la memoria.

La dirección se indica por un número que la CPU coloca en el bus de direcciones. Como se indicó, este número estará en notación binaria. Por ser este bus de 16 hilos, o líneas, podrá diferenciar, o direccionar, $2^{16} = 65.536$ posiciones diferentes de memoria, o 64 K (por K se entiende un kilobyte, que está formado por 1024 casillas).

Los datos a leer o escribir serán colocados, o leídos, del bus de datos. Por ser este bus de datos de 8 líneas, podrán tomar los datos $2^8 = 256$ valores diferentes; pudiendo representar estos datos instrucciones o datos propiamente dichos.

Por último, en el bus de control existe una señal llamada R/W (del inglés Read/Write = lectura/escritura), que indica si la CPU quiere leer o escribir un dato.

Si la señal R/W tiene un nivel lógico alto, un «1», indica que el procesador quiere leer el dato; por el contrario, si tiene un nivel lógico bajo, un «0», el procesador querrá escribir el dato.

Como se dijo en el capítulo anterior, el ordenador funciona de modo síncrono dirigido por un reloj; luego las distintas operaciones descritas anteriormente seguirán un orden en el tiempo.

Para realizar una operación de escritura la CPU coloca la dirección de la posición de memoria a leer en el bus de direcciones y un 1 en la línea de R/W durante la parte final del ciclo alto de $\phi 1$.

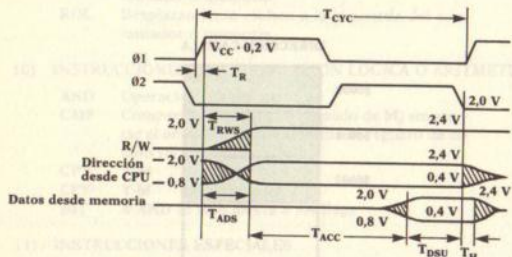


Fig. 3.2. Diagrama de tiempos de lectura.

Desde que la memoria recibe la señal de que el procesador quiere leer un dato hasta que lo coloca pasará un cierto intervalo de tiempo. En nuestro caso éste será desde que $\phi 2$ pasa de estado bajo a estado alto, pues al final del ciclo alto de $\phi 2$ el procesador tomará como dato el valor que la memoria coloque en el bus de datos.

El proceso de escritura es prácticamente idéntico.

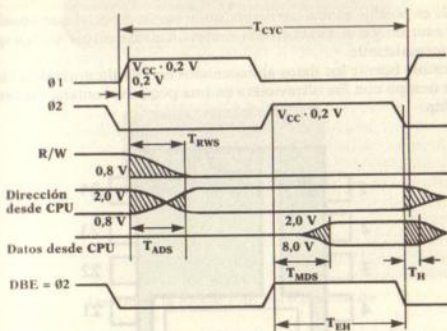


Fig. 3.3. Diagrama de tiempos de escritura.

La diferencia está en que mientras $\phi 2$ está alto R/W está a nivel bajo y que durante la parte alta de $\phi 2$ el microprocesador coloca el dato a escribir, es leído por la memoria y escrito.

La memoria de nuestro ordenador está formada por dos bloques principales.

La ROM (Read Only Memory = Memoria de sólo lectura) es la que almacena el programa, o software, que controla nuestro ordenador. Como su nombre indica, sólo se pueden realizar en ella operaciones de lectura. Si intentamos escribir un dato en ella, sencillamente no lo admitirá, permaneciendo la posición de memoria direccionada con el valor que tuviese en un principio. La ventaja de este tipo de memoria es que no se borra si se apaga el ordenador, es decir, los datos almacenados estarán siempre listos para la lectura.

Los circuitos usados normalmente como memoria ROM ya vienen con el programa grabado de fábrica. Por el alto coste que esto conlleva sólo se utiliza para grandes series de ordenadores comerciales, ya que así resulta más barato.

Otros circuitos de ROM son las PROM. Estas se pueden grabar, pero sólo una vez, ya que el proceso se realiza fundiendo pequeños fusibles que representan los bits. Tiene el problema que si cometemos un error éste no tendrá posible arreglo.

El chip, o circuito integrado, que utilizaremos es la EPROM (Erasable Programmable Read Only Memory = Memoria programable de sólo lectu-

ra). En él es posible grabar mediante un proceso especial que consiste en intentar escribir en ella con circuitos especiales de mayor voltaje que los usados normalmente.

Es posible borrar los datos almacenados en ella iluminándola durante un largo tiempo con luz ultravioleta en una pequeña ventana existente sobre el chip.

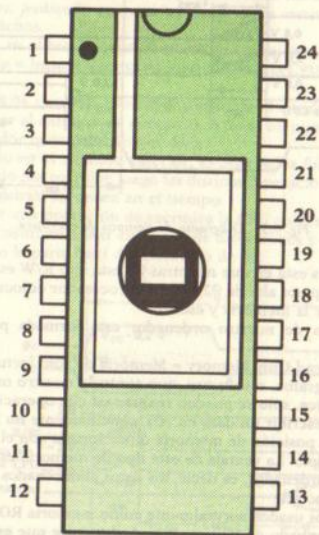


Fig. 3.4. Patillaje de una Eprom.

Por ello, esta ventana deberá estar tapada por algún tipo de etiqueta, o pegatina, ya que una excesiva exposición a los rayos solares podrían borrar nuestro programa.

El otro tipo de memoria es la RAM (Random Access Memory = Memoria de acceso aleatorio, es decir, que puede accederse a cualquiera de sus



Aspecto de la Eprom y del microprocesador.

posiciones). En ésta se puede leer y escribir, pero tiene el defecto que todo lo almacenado se pierde si se apaga el ordenador.

En el comercio existen dos tipos de memoria RAM: estáticas y dinámicas.

Las estáticas son más antiguas, pero son más sencillas de manejar por necesitar menos circuitos adicionales. Además, para sistemas pequeños como el nuestro son las más indicadas.

Las dinámicas tienen mayor capacidad, pero tienen un defecto, que olvidan los datos que tienen almacenados cada milésima de segundo, por lo que hay que estar «refrescándolas» constantemente, con la circuitería que esto lleva asociado.

Si los datos que tenemos en esta memoria quieren ser conservados deberán almacenarse en otra parte para no perderlos, tales como guardarlos en una cinta o cassette.



EL DECODIFICADOR DE DIRECCIONES

El decodificador de direcciones es una parte esencial del circuito de un ordenador. Su función es la de activar una de las patillas del circuito dependiendo de los bits de mayor peso del bus de direcciones. Es decir, de la zona de memoria en la que pretende trabajar el microprocesador y que corresponde a la posición que ocupa en ella el dispositivo y que hemos prefijado previamente mediante el diseño del mapa de memoria.

El decodificador de nuestro ordenador ha sido simplificado expresamente para que podamos realizarlo con unos pocos chips TTL-LS.

Con ellos implementamos las funciones lógicas que tomarán el valor

adecuado para activar las pastillas cuando en el bus de direcciones halla las direcciones correspondientes. Para calcular estas funciones no sólo es necesario conocimientos de electrónica digital, sino también del contenido de los chips existente en el mercado para utilizar el menor número de ellos. Existe una amplia bibliografía al respecto (ver apéndice), aunque en este tipo de diseños es importante la habilidad y técnicas «artesanas».

También se puede aprovechar el hecho de que algunas pastillas tengan varias pastillas o terminales de activación del chip a la hora de simplificar el circuito decodificador.

En nuestro caso el decodificador activa la pastilla de 8K de RAM (RAM estática 5565) cuando se direccionan los 8 primeros Kbytes de la memoria. Activa el RIOT cuando utilizamos las direcciones \$20XX y la VIA con las direcciones \$21XX.

Cuando utilizamos los dos últimos Kbytes, activa la EPROM, donde están almacenados los programas de utilidades que se proporcionan.



A función de las etapas de entrada-salida es comunicar el ordenador con el exterior. En la mayoría de los casos debe adaptar el tipo de señal que le llega del exterior a una señal que sea comprensible por el ordenador.

El lugar de llegada de los datos del exterior se conoce normalmente como port, o puerto, siendo en nuestro caso de igual longitud que el bus de datos, 8 bits.

Por ser este tipo de circuitos, usualmente, complicados y voluminosos usaremos dos circuitos integrados especializados en estas tareas: la VIA 6522 y el RIOT 6532, de la misma familia que el microprocesador que utilizamos.

La VIA (Versatil Interface Adapte = Adaptador versátil de periféricos) dispone de los siguientes elementos:

- Dos puertos, conocidos como A y B, de ocho líneas. Cada línea es programable como entrada o salida independientemente.
- Cuatro líneas de control, dos de cada puerto. Son conocidas como CA1, CA2, CB1 y CB2.
- Un registro de desplazamiento de 8 bits para convertir la información serie en paralelo, y viceversa.
- Dos temporizadores de 16 bits usados para contar o generar impulsos.
- Lógica para interrumpir a la CPU mediante la señal IRQ.
- Registros programables para el uso de los diferentes recursos.

Los registros de la VIA se sitúan en la memoria principal usando la lógica de decodificación que se vio en el capítulo anterior. Existen 16 posiciones de memoria que se direccionan por 4RS0, RS1, RS2, RS3

Nota: El único problema que se puede encontrar en el decodificador es el de los «glitches» (ver bibliografía sobre este tema). En nuestro caso, es tan sencillo y su retardo tan pequeño (20nS x n como máximo) comparado con el periodo de reloj (1 uS = 10⁻⁶S).

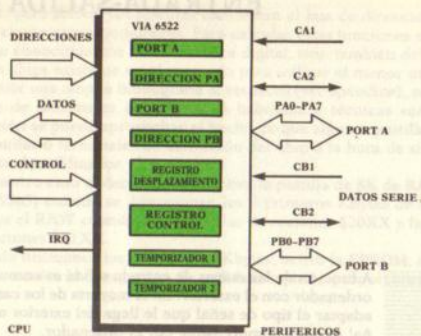


Fig. 4.1. Esquema de la vía.

(4 bits $\Rightarrow 2^4 = 16$ posiciones). El decodificador activa las patillas CS1 y CS2 cuando nos referimos a las direcciones \$21XX.

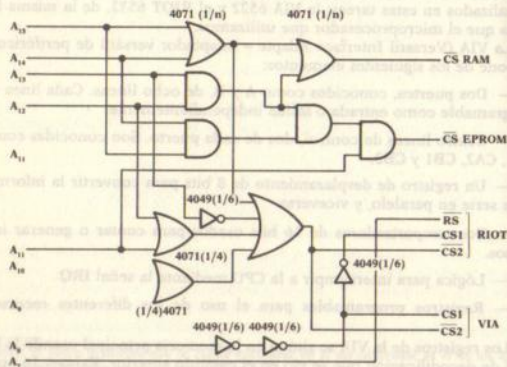
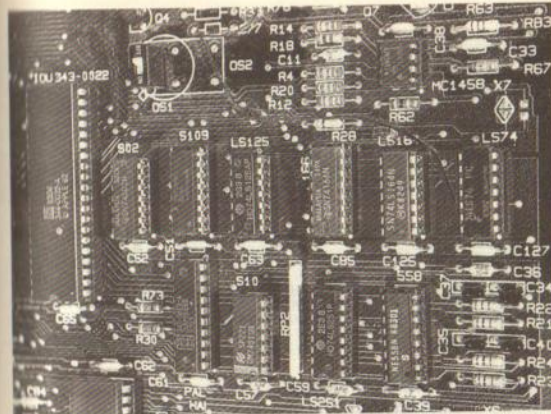


Fig. 4.2. Decodificador de direcciones.



Ejemplo práctico del decodificador de un ordenador.

Las direcciones de los port son \$2100 para el B y \$2101 para el A. Las direcciones de los registros de control son \$2102 para el B y \$2103 para el A.

Para programarlos cada bit se colocará un 0 si se quiere programar como entrada y 1 si se quiere programar como salida en los registros de direcciones.

Por ejemplo, para programar todo el port B como salida pondríamos en su registro de direcciones el número:

11111111

después de esto todo lo que colocáramos en los registros de los ports saldría al exterior; por el contrario, si lo que queremos es leer en el port B, en el registro de control se colocaría:

00000000

y después el valor que leeríamos del registro del port B sería el que le estuviera llegando del exterior.

Veamos dos ejemplos:

```

;L
1      LDA  #%11111111 ;PROGRAMA SALIDAS
2      STA  DRBU
3      LDA  PBV          ; LEER DATOS

```

```

;L
1      LDA  #%00000000 ;PROGRAMA SALIDAS
2      STA  DRBU
3      STA  PBV          ; ESCRIBE DATOS

```

El registro de control de las líneas CA1, CA2, CB1 y CB2 está en la posición \$200C. Su configuración se observa en la figura 4.3.

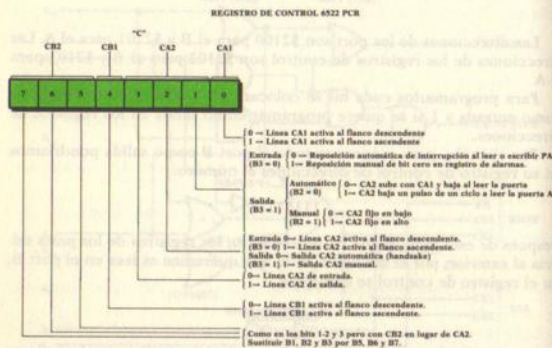


Fig. 4.3. Función de cada bit del registro de control PCR.

Estas líneas, dos para cada port, sirven para establecer un protocolo, o handshaking. Esta es la forma que tienen dos ordenadores para comunicarse. Antes de transmitirse datos deben mandar la señal equivalente de «que van los datos» y si los recibe bien, decir «ya los tengo». El equivalente de las personas son las frases hechas que se utilizan al comenzar una conversación; por ejemplo, dos amigos se encuentran, y antes de contarse nada de interés dicen:

Hola, Fulano, ¿la familia que tal?

a lo que el otro contesta:

La familia bien, ¿y tú que tal?

Bien

le responde, y a partir de aquí comienzan a hablar de los temas que les interesan.

En el siguiente programa observamos un sencillo protocolo que usa sólo una señal de ocupado (Busy). Suele ser el usado para comunicarse con una impresora.

```

;L
1      STA  PAV          ;DEJA EN PORT A
2      SIGUE LDA  PCRV          ;COMPROBAR SI LEIDO
3      AND  #%00001000
4      BEQ  SIGUE       ;SI NO REPETIR

```

La posibilidad de realizar entrada-salida por interrupciones es muy útil. Para ello existe el registro de control situado en la posición \$200D el de alarma y en la \$200E el de activación.

En la figura 4.4 vemos las misiones de los distintos registros.

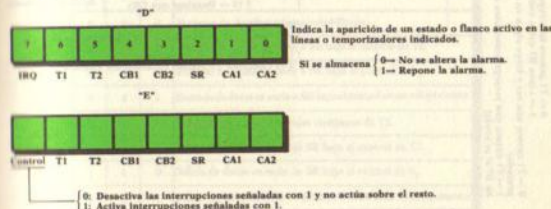


Fig. 4.4. Registro de alarmas de interrupciones IFR.

La utilidad de la interrupción es clara en el siguiente ejemplo. Supongamos que tenemos que realizar unas operaciones en la CPU, y leer, por ejemplo, un teclado. Por ser la velocidad de las personas lenta comparada con el ordenador. Si realizáramos un bucle de espera que comprobase si se ha pulsado alguna tecla se perderían algunos segundos, durante los cuales la CPU no ha realizado ninguna de las operaciones pendientes. Por tanto, si hace las operaciones, éstas podrían durar demasiado tiempo y no leer la tecla cuando fuese pulsada. La solución es programar la VIA de forma que cuando llegue un dato genere una señal de interrupción, se lea la tecla pulsada y se continúe con la ejecución del programa.

La VIA también dispone de dos temporizadores, o Timers, que tienen diversas utilidades: generar cada cierto número de reloj una interrupción, cortar los impulsos que llegan por las patillas PB6 o generar dichos impulsos por la línea PB7.

En la figura 4.5 se ve qué registros utiliza y su forma de programación.

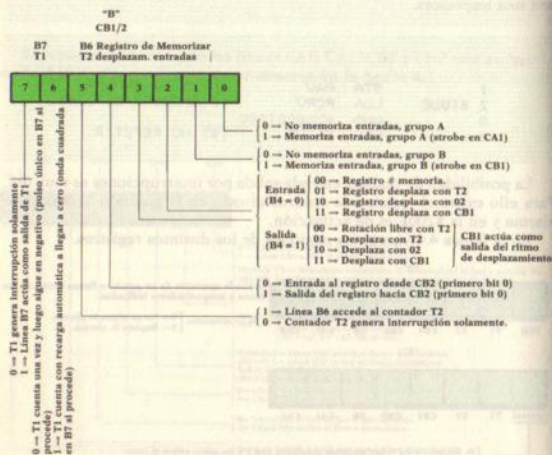


Fig. 4.5. Registro auxiliar de control ACR.

Por último, nos queda por ver el registro de desplazamiento. Este sirve para convertir los datos serie en paralelo, y viceversa.

Los datos están en paralelo, por ejemplo, en el bus de datos. Al mandarse un dato se mandan los ocho bits simultáneamente por las ocho líneas. Esto requiere menos tiempo, pero ocupa más sitio.

El enviar los datos en serie se enviarlos en «fila india». Esto ocupa más tiempo, pero sólo una línea. Para convertir de una forma a otra se utiliza el registro que pasa los bits que le llegan desplazando los que ya le han llegado, de ahí el nombre.

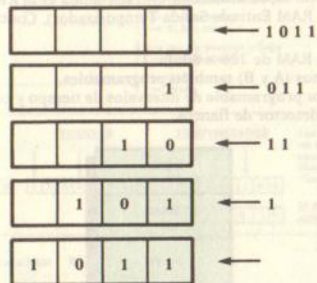


Fig. 4.6.

El registro de desplazamiento está situado en la posición de memoria \$200A y el control en \$200B (compartido con el temporizador).

Bits de ACR			MODO DE FUNCIONAMIENTO DE SR
2	3	4	
0	0	0	Registro de desplazamiento inhabilitado.
0	0	1	Entrada de datos en serie a SR bajo el control de T2.
0	1	0	Entrada de datos en serie a SR bajo el control de θ_p .
0	1	1	Entrada de datos en serie a SR bajo el control de un reloj externo.
1	0	0	Salida de datos en intervalos continuos de T2.
1	0	1	Salida de datos en serie de SR bajo el control de T2.
1	1	0	Salida de datos en serie de SR bajo el control de θ_p .
1	1	1	Salida de datos en serie de SR bajo el control de un reloj externo.

Fig. 4.7. Funcionamiento del registro de desplazamiento.

La frecuencia con que se realiza el desplazamiento puede determinar se por:

- La activación de la bandera T2.
- Con O2.
- Los flancos descendentes de CB1 (por flanco descendente se entiende el paso de un nivel alto a un nivel bajo).

Un ejemplo de uso de este registro se ve mas adelante en el interfaz RS-232.

El otro circuito especializado de entrada-salida es la RIOT (RAM Input-Output Timer = RAM Entrada-Salida Temporizador). Contiene los siguientes elementos:

- Memoria RAM de 128 x 8 bits.
- Dos puertos (A y B) también programables.
- Generador programable de intervalos de tiempo y con interrupción.
- Circuito detector de flancos.

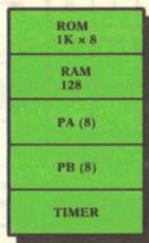


Fig. 4.8. RIOT.

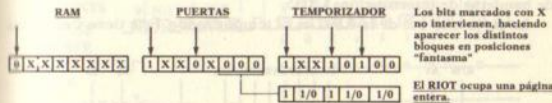
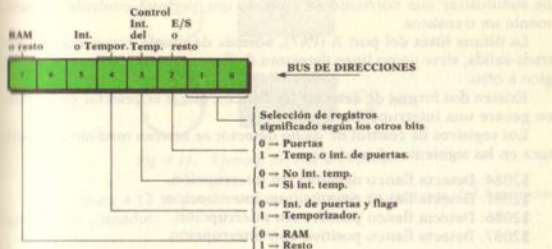
La memoria RAM es idéntica a la utilizada por el ordenador.

Sus direcciones en nuestro mapa de memoria son desde \$200 hasta \$207F. El decodificador descrito en el capítulo anterior la selecciona utilizando las líneas CS1 y CS2.

La función de esta memoria es para usar el ordenador como un control industrial. Se puede realizar una configuración mínima con el microprocesador y la RIOT (versión de este mismo chip que incorpora una memoria ROM).

Los ports de la RIOT son idénticos a los vistos para la VIA. La dirección de port A es \$2080 y la de su registro de control es \$2081. La dirección del port B es \$2082 y la de su registro de control \$2083.

RIOT (ocupa 256 bytes)



00	Memoria RAM	RS = 0		
01				
02	PA, Datos	RS = 1		
03	PA, Programación	A2 = 0		Puertas (E/S)
04	PB, Datos			
05	PB, Programación			
06	PA7, Flanco negativo, no int.	RS = 1		Programación
07	PA7, Flanco positivo, no int.	A2 = 1		Interrupciones
08	PA7, Flanco negativo, si int.	A4 = 0		de puertas
09	PA7, Flanco positivo, si int.	R/W = 0		(Solo E.)
0A	Lectura y reposición de flags de int.	PA7 TEMP.	7 6 ----- 0	Lectura: Flags de interrupción (Solo L)
0B	E: Temp. en + 1 no int.	(Escripción)		Temporizador
0C	E: Temp. en + 8 no int.			Programación
0D	E: Temp. en + 64, no int.			contaje e int.
0E	E: Temp. en + 1.024, no int.			no legible
0F				contaje actual
10				repite interr.
11	E: Temp. en + 1, si int.			Temporizador
12	E: Temp. en + 8, si int.			Programación
13	E: Temp. en + 64, si int.			contaje e int.
14	E: Temp. en + 1.024, si int.			si legible
15				contaje actual
16				repite interr.

Fig. 4.9.

En este chip los ports no son exactamente iguales, en port B es capaz de suministrar una corriente de 3mA, lo que permite controlar directamente un transistor.

La última línea del port A (PA7), además de usarse como línea de entrada-salida, sirve como línea detectora de flancos, cambios de un nivel lógico a otro.

Existen dos formas de detectar los flancos: que si al detectar dicho flanco genere una interrupción, o no.

Los registros de control de dicho detector se activan mediante la escritura en las siguientes direcciones:

§2084: Detecta flanco negativo sin interrupción.

§2085: Detecta flanco negativo con interrupción.

§2086: Detecta flanco positivo sin interrupción.

§2087: Detecta flanco positivo con interrupción.

Para desactivar la detección de flancos será suficiente leer el registro de banderas de interrupción: §2085.

La parte más útil de la RIOT es el temporizador. Este tiene tres partes.

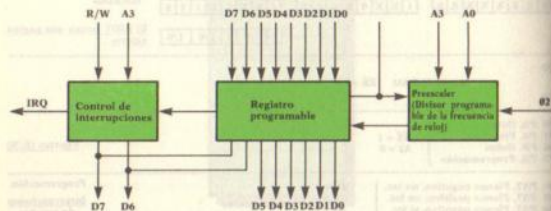


Fig. 4.10. Esquema de la RiOT.

El reloj $\emptyset 2$ pasa al divisor programable de la frecuencia de reloj. Este puede dividir la frecuencia por 1, 8, 64 y 1.024.

La señal dividida pasa al contador propiamente dicho. Este puede programarse para contar 255 intervalos de tiempo. Al llegar a 255 generará, si se lo indicamos, una señal de interrupción. También leen el contador, con lo que podemos tener ideas sobre el tiempo transcurrido desde que los activamos.

Un ejemplo de utilización de este contador, para realizar un contador de tiempo real de veinticuatro horas, se puede ver en el temporizador descrito en el capítulo siguiente.

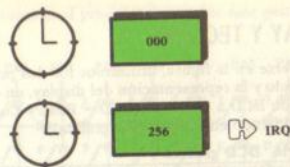


Fig. 4.11. Ejemplo del uso del contador.

En la figura 4.12 se puede ver un diagrama de tiempos del funcionamiento del contador.

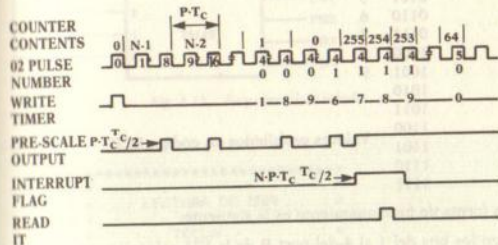


Fig. 4.12. Diagrama de tiempos del contador.

APLICACIONES

Vamos a ver cómo aplicar lo que acabamos de ver a la práctica. Como comunicaciones con las personas incluiremos un pequeño teclado de 21 teclas y un display de siete segmentos y seis dígitos. Para comunicaciones con otros ordenadores realizaremos dos interfaces: RS-232 y CENTRONICS, serie y paralelo, respectivamente. Estos son los más comúnmente utilizados en los ordenadores personales.

DISPLAY Y TECLADO

Como puede verse en la figura, utilizamos los dos ports de la vía para la lectura del teclado y la representación del display, en combinación con un decodificador de BCD a decimal. (BCD → Binary Coded Decimal = es decir, decimal codificado en binario (ver gráfica)).

Binario BCD gráfica

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	
1011	
1100	
1101	Valores prohibidos en código BCD
1110	
1111	

La forma de funcionamiento es la siguiente:

Con los bits del 1 al 4 del port B de la VIA seleccionamos una de las tres filas del teclado (para leer) o uno de los seis displays de siete segmentos (para escribir en ellos). (Con lo que sobra una patilla, la 3.) Al mismo tiempo utilizamos los bits del 0 al 6 del port A para leer o escribir, dependiendo del caso, las siete líneas correspondientes a los siete segmentos, o a las siete columnas del teclado. Veamos ahora por separado cada una de las dos partes: En primer lugar, veamos el proceso de lectura del teclado de una forma algo más detallada.

Para leer el teclado programamos el PORT B como salida y activamos secuencialmente las patas, de la 1 a la 4 inclusive, escribiendo los valores BCD del 0 al 2. Con esto activamos sucesivamente una y sólo una de las filas del teclado (que son tres).

Al mismo tiempo exploramos con el Port A (programado como lectura), las siete columnas con los bits del 0 al 6.

Así, cada vez que activamos una fila, leemos las siete columnas secuencialmente. Si el valor leído es 1 en algún caso, la tecla correspondiente de fila y columna estará pulsada.

Véase a continuación el programa máquina que gestiona el teclado siguiendo esta idea.

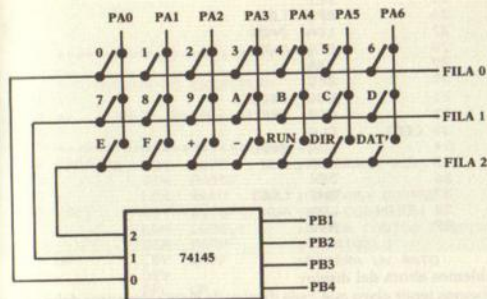


Fig. 4.13. Esquema del teclado.

```

1 *****
2 *                                     *
3 *      LECTURA DE UNA              *
4 *                                     *
5 *      TECLA                         *
6 *                                     *
7 *****
8 LEEKB  LDA  #00          ;PROGRAMA PORT
9        STA  PADDR
10       LDX  #06
11       STX  PDBR
12 LEE1  LDA  PADR
13       BNE  SI
14       INC  #1
15       CPX  #09
16       BNE  LEE1
17       JSR  ESCRIBE
18       JMP  LEEKB
19 SI    STX  AUX1
20       PHA
21       JSR  ESCRIBE
22       PLA
23       LDX  AUX1
    
```

```

24 LEE2    LDY  ##7F
25        DEY
26        BPL  LEE2
27        LDA  PADR
28        STA  AUX
29        TXA
30        SEC
31        SBC  ##06
32        BEQ  LEF3
33 LEE3    CLC
34        LDA  ##06
35        ADC  AUX
36        DEX
37        BNE  LEE3
38 LEF3    LDA  AUX
39        RTS

```

Hablemos ahora del display.

Debemos tener claro que cada display de siete segmentos debe ser reescrito cada cierto tiempo para que se mantenga constantemente encendido. Por ello, lo refrescaremos siempre que la CPU no tenga otra tarea más importante que hacer.

Para escribir en un determinado dígito del display programamos como salida el Port B y activamos las patas de 1 a 4 con el número en BCD correspondiente al dígito a escribir (del 4 al 9). En el port A (programado como salida) escribimos en los bits del 0 al 7 el carácter a escribir, con lo que cada bit encenderá el segmento correspondiente.

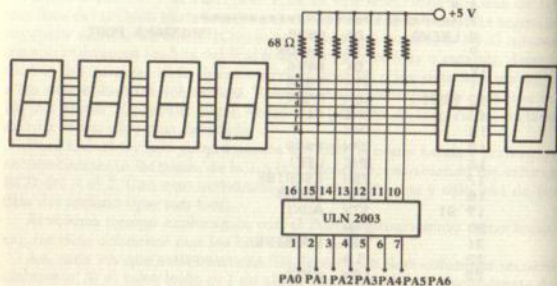


Fig. 4.14. Esquema del «Display».

Veamos el programa de control.

```

IL
1 *****
2 *
3 *   RUTINA DE ESCRITURA   *
4 *
5 *   EN EL DISPLAY         *
6 *
7 *****
8 ESCRIBE LDA  ##7F          ;PROGRAMA PORT A
9         STA  PADDR
10        LDX  ##05          ;PREPARA BUFFER
11 ESCI   LDY  OUTBF,X       ;LEE CODIGO
12        LDA  CODE,Y       ;CARGA CODIGO DISPLAY
13        STA  PADR         ;ESCRIBELO
14 UP     LDY  ##7F          ;ESPERA UN RATO
15        DEY
16        BPL  UP
17        STY  PADR         ;REPROGRAMA PORT A
18        LDA  ##06
19        STA  PBDR         ;REPROGRAMA PORT B
20        DEX
21        BPL  ESCI         ;FIN ?
22        RTS

```

Utilizamos la tabla para encender los segmentos de cada dígito.

0	7	E	E
1	1	8	F
2	2	9	A
3	3	A	X
4	4	B	Y
5	5	C	P
6	6	D	S

Fig. 4.15. Códigos para «Display».

Pero si quisiera usar otra, aqui damos todos los posibles códigos.

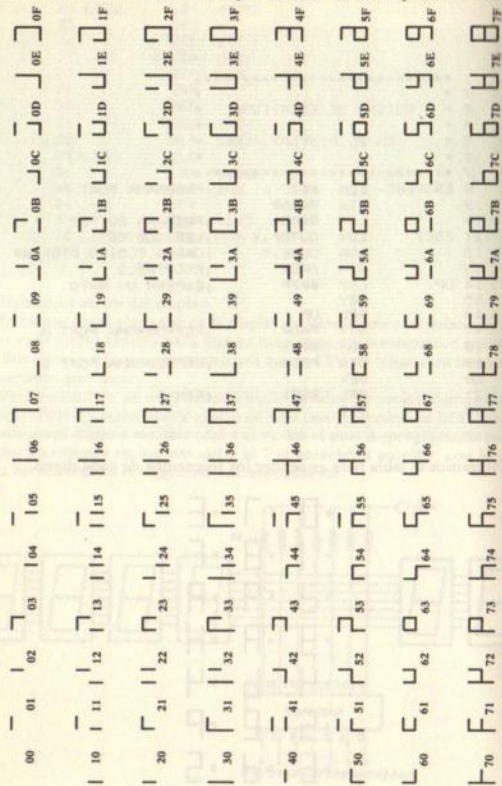


Fig. 4.16. Figuras posibles del «Display».

Y una propuesta de caracteres posibles, para que el sufrido lector pueda utilizar caracteres alfanuméricos:



Fig. 4.17. Posible código ASCII en el «Display».



CENTRONICS

El interface paralelo más usado es el CENTRONICS. En él se comunican los ocho bits de golpe por ocho líneas diferentes.

Uno de los datos principales de un interface es su velocidad de transmisión. Esta suele darse en baudios, que son los bits de información que transmite por segundo. Parece lógico que ésta sea lo más alta posible, pero hay que tener en cuenta que a mayor velocidad hay una mayor probabilidad que se cometan errores de transmisión (que los datos recibidos no sean los mismos que los emitidos).

Para su realización práctica usaremos el port A de la VIA para los datos y las líneas CA1 y CA2 para el protocolo.

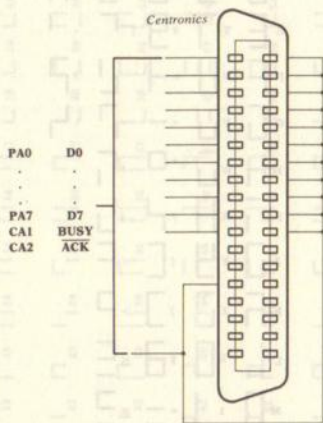
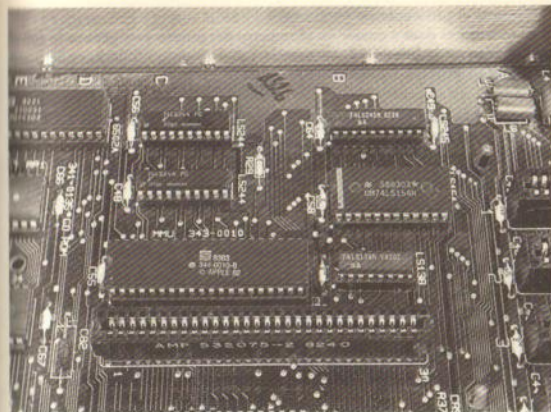


Fig. 4.18. Conexión interfaz Centronics.

Vamos a ver cómo funciona este protocolo. Este es realizado automáticamente por la VIA, una vez programada para ello.

Vamos a ver el protocolo de lectura.

El periférico del que queremos leer debe mandar una señal por la li-



Realización práctica de entrada/salida.

nea CA1 de que está listo para recibir los datos (si nuestro periférico no dispone de ella, deberemos tener esta señal en nivel alto). Después el puerto leerá los datos, y mandará por la línea CA2 una señal de que ha recibido los datos.

En la siguiente figura se observa un cronograma de esta operación.

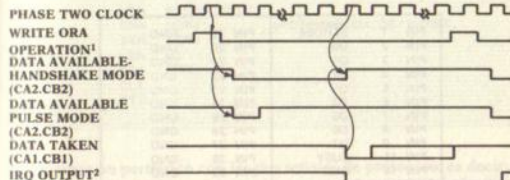


Fig. 4.19. Diagrama de tiempos del «Handshake» de escritura.

Veamos el protocolo de escritura:

Al escribir el dato a enviar en el registro del puerto éste genera la señal de dato listo en la línea CA2 y espera la señal de dato recibido en la línea CA1. Esto se observa en la siguiente figura:

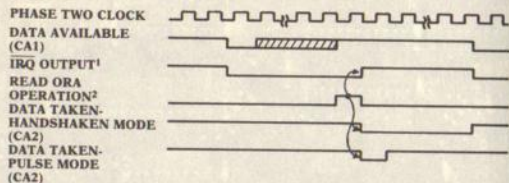
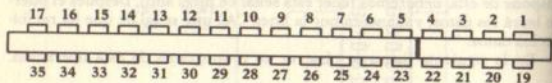
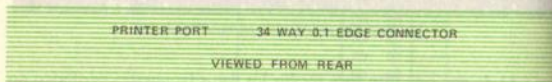


Fig. 4.20. Diagrama de tiempos del «Handshake» de lectura.

En nuestro caso el protocolo del AMSTRAD sólo dispone de la línea de «Busy» (equivalente a la señal de dato recibido vista anteriormente).



PIN 1	STROBE	PIN 19	GND
PIN 2	D0	PIN 20	GND
PIN 3	D1	PIN 21	GND
PIN 4	D2	PIN 22	GND
PIN 5	D3	PIN 23	GND
PIN 6	D4	PIN 24	GND
PIN 7	D5	PIN 25	GND
PIN 8	D6	PIN 26	GND
PIN 9	D7	PIN 27	GND
PIN 11	BUSY	PIN 28	GND
PIN 14	GND	PIN 33	GND
PIN 16	GND	All other pins	NC

Fig. 4.21. Conector paralelo Centronics.

El programa que gestiona este interfaz es el siguiente:

```

1 *****
2 *
3 *   PROG. CENTRONICS   *
4 *
5 *****
6 *
7 * PROGRAMACION DEL PORT A
8 * COMO ENTRADA/SALIDA
9 * INTERFACE CENTRONICS
10 *
11 * PARA LEER EL PUNTO DE ENTRADA
12 *   ES INCENT
13 *
14 * PARA ESCRIBIR ES OUTCENT
15 *
16 INCENT LDA #0           ;RUTINA DE ENTRADA
17       STA DRWU         ;PROGRAMA REG. DIRECCION
18       STA CENFLAG
19       JMP CONT
20 OUTCENT LDA #FF        ;RUTINA SALIDA
21        STA DRWU         ;PROGRAMA REG. DIRECCION
22        LDA #001
23        STA CENFLAG
24 CONT   LDA #0           ;PARTE COMUN
25        STA PAV          ;REGISTRO FLAGS INT.
26        LDA PCRV         ;REGISTRO DE CONTROL
27        AND #11110000
28        ORA #00001000
29        STA PCRV
30 *
31        LDA ACRV          ;REGISTRO AUXILIAR DE CONTROL
32        AND #11111110
33        ORA #00000001
34        STA ACRV
35 *
36        LDA #10000011
37        ORA IERV         ;ACTIVACION DE INTER.
38        STA IERV
39        LDA #0
40        STA IFRV
41        CLI
42        RTS

```

Si se usase un periférico con las dos señales de protocolo, es decir, con la señal «ACK» (equivalente del papel representado por CA2 en lo visto anteriormente), el esquema de conexión sería idéntico, pero conectando CA2 a ACK.

EXPLICACION DEL INTERFACE RS-232

El interface RS-232 (o V24) es uno de los interfaces serie más comúnmente utilizados para la transmisión de todo tipo de datos.

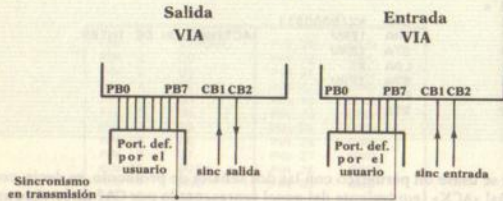
Muchos periféricos de ordenador disponen del mismo, como estándar, por lo que es obvia su gran utilidad y la conveniencia de disponer de él para conseguir una mayor versatilidad en un equipo informático. Sin embargo, y tal vez por la dificultad de obtener sus tensiones, muchos de los ordenadores personales carecen de esta interface (como, por ejemplo, la gama AMSTRAD).

Nosotros hemos adaptado nuestra fuente para que proporcione las tensiones necesarias. Más adelante explicaremos cómo utilizar nuestra placa microcomputadora como adaptador de interfaces RS232-CENTRONICS.

Como la transmisión es serie necesitaremos programar el Port B de la VIA adecuadamente, ya que el Port A no permite la adaptación de forma automática de paralelo (del bus de datos: ocho bits) al canal serie que necesitamos.

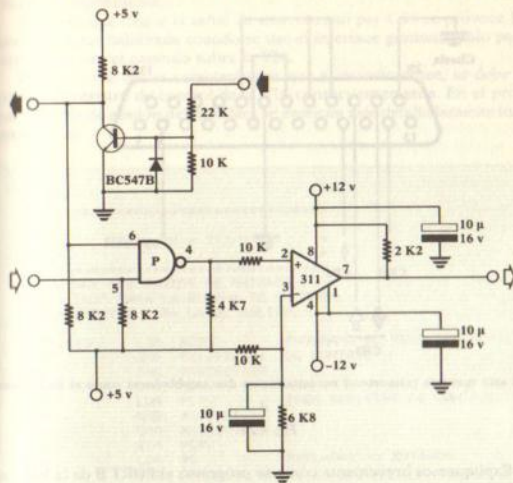
El RS-232 funciona con lógica negativa. Esto quiere decir que cuando queramos transmitir un «0» debemos transmitir un valor de tensión positivo (+12, aunque permite en teoría cualquiera en el rango +5+15) y cuando deseamos transmitir un «1» negativo -12 (-5-15). Por estas razones, necesitamos hacer un cambio en el nivel de tensiones a la salida serie de la VIA, esto se puede hacer con el circuito de la figura 4.22. En ella se puede ver también unas indicaciones acerca del conector, el patillaje, etc.

Interface RS-232



a

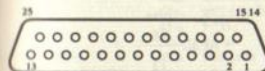
Por cada canal de I/O necesitaremos un adaptador de nivel como el de la figura:



La puerta P forma parte del IC. 74LS27 (que tiene cuatro puertas NAND).

b

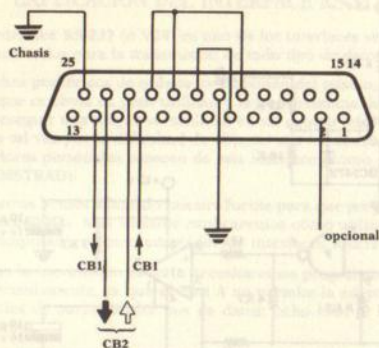
El conector RS 232/V24



1. Masa del chasis *
2. Emisión de datos \leftarrow a CB2
3. Recepción de datos \rightarrow
4. Solicitud de emisión *
5. Listo para emitir *
6. Bloqueo de datos listo *
7. Masa de señal \perp
8. Detección de portadora
- 9-14. No utilizados
15. Bit de sincronización interno para emisión. a CB1
16. No utilizado
17. Bit de sincronización de recepción, procede del emisor. a CB1
- 18-19. No utilizados
20. Terminal de datos listo
21. No utilizado
22. Indicador de llamada entrante
23. Selector de velocidad de transmisión
24. Bit de sincronización de emisión hacia el dispositivo de transmisión (a CB1)
25. No utilizado

c

Una forma sencilla de conexión sería la siguiente:



Con este montaje (asíncrono) necesitaríamos dos cambiadores como el de la figura.

d
Fig. 4.22.

Explicaremos brevemente cómo se programa el PORT B de la VIA para la gestión serie (→ RS-232/V24).

Como entrada

	bits ACR		bits
	234		76
Asíncrono	011	→ ACR:	76
Síncrono	001		11

Como salida

	bits ACR		bits
	234		76
Asíncrono	111	→ ACR:	76
	100		11

En el caso síncrono habrá que escribir en T1 (registro de la VIA) el período correspondiente.

Cada vez que llegue la señal de sincronismo por CB1 se provoca IRQ, que debe estar habilitada cuando se use el interface gestionándolo por interrupción: ver el capítulo sobre la VIA.

Dependiendo de las características que el usuario desee, se debe programar el registro de control de la VIA consecuentemente. En el programa máquina de gestión del Interface se explican más detalladamente los pasos a seguir.

```

tL
1 *****
2 *
3 *   PROG. RS = 232/V-24   *
4 *
5 *****
6 * HAY DOS PUNTOS DE ENTRADA
7 * INRS PARA LA RUTINA DE ENTRADA
8 * Y OUTRS PARA LA DE SALIDA
9 *
10 INRS   LDA  ACRV      ;PROGRAMA EL REGISTRO AUXILIAR
11        AND  W%11100001 ;DE CONTROL
12        ORA  W%00011110
13        STA  PCRV      ;IDEM REGISTRO DE CONTROL
14        LDA  ACRV      ;IDEM REGISTRO DE CONTROL
15        AND  W%00001111
16        ORA  W%10000000
17        STA  PCRV
18        LDA  #0        ;RSFLAG=0 -> ENTRADA
19        STA  RSFLAG
20        JMP  CONTI
21 OUTRS  LDA  ACRV      ;IDEM REGISTRO AUXILIAR DE
22        AND  W%11100001 ;CONTROL
23        ORA  W%00011110
24        STA  ACRV
25        LDA  PCRV      ;IDEM REGISTRO DE CONTROL
26        AND  W%00001111
27        ORA  W%00000000
28        STA  PCRV
29        LDA  #1        ;RSFLAG=1 -> SALIDA
30        STA  RSFLAG
31 CONTI  LDA  #0        ;ACTIVAMOS LA INTERRUPTIOIN
32        STA  IFRV      ;CORRESPONDIENTE
33        LDA  IERV
34        ORA  W%10000100
35        STA  IERV
36        CLI
37        RTS
;PR#0
    
```

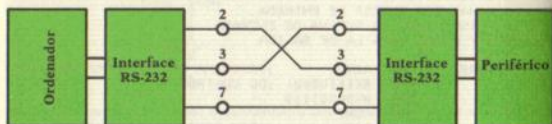
Como se observa en el programa, la velocidad de transmisión es programable por software, por lo que podrá variar según la aplicación.

AMPLIACIONES

Con lo visto hasta ahora ya tenemos la versión básica de nuestro ordenador. Por sobrar líneas de los puertos de entrada salida que se pueden usar para posibles ampliaciones.

Para añadirle más memoria a nuestro ordenador será necesario colocar más decodificadores para generar la señal de selección de chips, según la cantidad de memoria que queramos añadir. Para ello, debemos observar en el mapa de memoria qué zonas están libres.

Como conexión mínima:



Una vez vista la zona en la que queremos colocar la memoria debemos ver qué direcciones, en binario, le corresponden; con ello diseñaremos un circuito combinatorial que detecte dichas direcciones.

E

N este capítulo trataremos de la «inteligencia» del ordenador. Con un mismo hardware se pueden realizar muchas funciones distintas, que sólo dependerán de la imaginación de quien realice los programas.

Para guardar el programa seleccionamos una EPROM capaz de almacenar 2 Kbytes (2048 bytes), por ser las más fáciles de localizar en el mercado y aunque pueda parecer extraño, son más baratas que las EPROM de 1 Kbyte (1024 bytes) (porque al ser más antiguas escasean en el

mercado).

De todas formas, siempre podremos utilizar lo que sobra de RAM para introducir programas.

Por ello, realizamos dos tipos de programas: un pequeño editor que nos permita introducir nuestros programas en código máquina, depurarlos y ejecutarlos, y un programa de reloj de veinticuatro horas, que permita ejecutar una rutina definida por el usuario al llegar una hora predeterminada.

Junto a estos programas existen diversas rutinas, que éstos utilizan, que gestionan los recursos del sistema, tales como el teclado, el display, los interfaces serie y paralelo.

Siempre podremos poner en la RAM libre nuestras propias aplicaciones; además, la mitad de la EPROM está vacía. En ella podremos almacenar nuestras utilidades.

EDITOR

La parte principal de este programa está realizado por un bucle que espera que se pulse alguna tecla.

Si la tecla pulsada es de alguna de las funciones, lo que detecta por ser su código mayor que \$F, ejecuta la rutina.

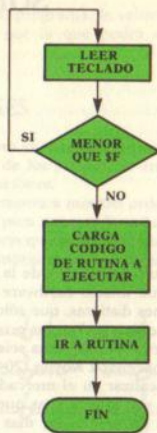


Fig. 5.1. Organigrama del bucle principal.

Para la ejecución de dicha función toma la dirección de una tabla, indexándola con el código de dicha función.

Almacena dicho dato en FUNC y ejecuta un salto indirecto a dicha posición, con lo que la dirección efectiva es la que almacenamos anteriormente.

```

1 *****
2 *                                     *
3 *      BUCLE PRINCIPAL                *
4 *                                     *
5 *****
6 ENTRADA JSR LDAT
7          JSR LEEKB          ;LEE TECLA
8 ENT1    CMP  #10           ;SI ES UN NUMERO
9          BMI ENTRADA       ;IGNORALO
10         CMP  #14           ;SI ERROR
11         BPL ENTRADA       ;IGNORALO
12         SEC
  
```

```

13          SBC  #10          ;TRANSFORMAR TECLA
14          ASL              ;EN VALOR
15          TAX              ;PARA CALCULAR DIRECCION
16          LDA TABLA,X      ;DE LA FUNCION
17          STA FUNC
18          INX
19          LDA TABLA,X
20          STA FUNC+1
21          JMP (FUNC)       ;EJECUTA LA FUNCION
  
```

Veamos ahora las distintas funciones del editor.

Para observar el contenido de una posición de memoria pulsaremos la tecla de función DIR, seguida de cuatro cifras hexadecimales, que serán la dirección a observar.

En el display aparecerá la dirección pulsada y el dato contenido en ella, en hexadecimal.

Si queremos observar el contenido de la dirección de memoria siguiente bastará con pulsar la tecla +.

Para observar el contenido de la anterior posición de memoria pulsaremos la tecla -.

Si ahora queremos modificar la posición de memoria cuyo contenido estamos observando bastará con pulsar la tecla de función DATO y dos cifras hexadecimales que representarán el nuevo contenido de dicha posición de memoria.

Con esto podemos observar y modificar cualquier posición de memoria de nuestro ordenador, pero ¿cómo ejecutar un programa?

Para ejecutar un programa pulsaremos DIR y la dirección en que comienza el programa. Después pulsaremos RUN y dicho programa comenzará a ejecutarse.

Además de observar las posiciones de memoria podríamos querer observar y modificar los registros internos de la CPU.

Para observar el contenido de cualquier registro pulsaremos, sucesivamente, las teclas DIR, DATO y una tercera que representará el código del registro a visualizar. Los códigos son:

- A para el acumulador
- 0 para el registro X
- 1 para el registro Y
- 2 para el registro de status
- 3 para el puntero del stack

Una vez realizada la anterior función si queremos modificar alguno de estos registros bastará con pulsar la tecla DATO y las dos cifras hexadecimales que representen el nuevo contenido.

Con lo visto hasta ahora si queremos escribir nuestros propios programas deberemos realizar las siguientes funciones:

- Escribir el programa en lenguaje ensamblador.
- Traducir, mediante la tabla que se da en uno de los apéndices, las instrucciones por su correspondiente código hexadecimal.

Pulsaremos la tecla DIR seguida de la dirección en que comienza nuestro programa.

Pulsaremos DATO y el código de la primera instrucción.

Pulsaremos +, DAFO y el código siguiente. Esta última operación la repetiremos hasta completar el programa.

Pulsaremos la tecla DIR y la dirección de comienzo de nuestro programa y con la tecla RUN lo ejecutaremos.

Por último, para detener la ejecución de un programa y regresar al monitor, pulsaremos RESET.

TEMPORIZADOR

Es otra rutina del monitor. Programa un reloj de veinticuatro horas. Para ejecutarlo debemos pulsar DIR, la dirección \$FF2F y la tecla RUN. Tras esta operación introduciremos la hora actual con ocho dígitos (dos para la hora, dos para los minutos y dos para los segundos). Y otros ocho para la hora en que queramos que se ejecute una rutina, cuya dirección colocaremos previamente en las posiciones de memoria \$28 y \$29. Tras esto, el reloj entrará en funcionamiento.

E

N este capítulo veremos algunos de los posibles usos del ordenador anteriormente desarrollado. Estos son algunos de los muchos posibles, ya que las posibilidades son prácticamente ilimitadas.

Los casos aquí estudiados son: un adaptador del interface RS-232 a CENTRONICS, y viceversa, y un temporizador para el control de algún electrodoméstico.

ADAPTADOR RS-232-CENTRONICS (Y VICEVERSA)

Esta aplicación pretende proporcionar al usuario de un ordenador personal que posea uno de los dos interfaces, pero carezca del otro, la posibilidad de utilizar ambos, lo que le permitirá conectar su ordenador a otros periféricos sin un desembolso excesivo.

Habrà, por tanto, dos maneras de utilizar la placa microcomputadora como interface:

1. Entrada RS-232 → Salida CENTRONICS
2. Entrada CENTRONICS → Salida RS-232

NECESIDADES HARDWARE

Hace falta para ambas el adaptador (o adaptadores) de nivel explicados en el apartado del interface RS-232/V24, así como ambos conectores estándar RS-232 y CENTRONICS.

NECESIDADES SOFTWARE

Proponemos el siguiente programa máquina para la gestión de los interfaces en esta aplicación. Es importante seguir paso a paso las explicaciones al margen para comprender con detalle los pasos a seguir.

El usuario podrá siempre cambiar el programa para adaptarlo a sus propias necesidades.

Para la primera aplicación:

```

;L
1 *PROGRAMA MISMOS BUFFER
2 *PARA LA ENTRADA Y SALIDA
3
4 LDA ##00
5 STA CENBUFF
6 STA RSBUFF
7 LDA ##08
8 STA CENBUFF+1
9 STA RSBUFF+1
10 JSR CENOUT ;PROGRAMA CENTRONIS SALIDA
11 JSR RSIN ;PROGRAMA RS-232 ENTRADA
RTS ; LO HACE
;
    
```

Para la segunda:

```

0
;L
1 *PROGRAMA MISMOS BUFFER
2 *PARA LA ENTRADA Y SALIDA
3
4 LDA ##00
5 STA CENBUFF
6 STA RSBUFF
7 LDA ##08
8 STA CENBUFF+1
9 STA RSBUFF+1
10 JSR CENIN ;PROGRAMA CENTRONIS ENTRADA
11 JSR RSOUT ;PROGRAMA RS-232 SALIDA
RTS ; LO HACE
;
    
```

En cualquier ordenador se mandarán los datos igual que se escribe por la impresora.

TEMPORIZADOR

Como ejemplo más típico damos un controlador programable de un electrodoméstico.

Programando el temporizador de la RIOT para realizar un reloj de veinticuatro horas, podemos saber qué hora es.

Cuando llegue la hora asignada a través de un relé, interruptor controlado por tensión, podemos enchufar y desenchufar el aparato.

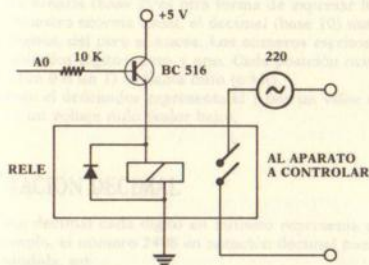


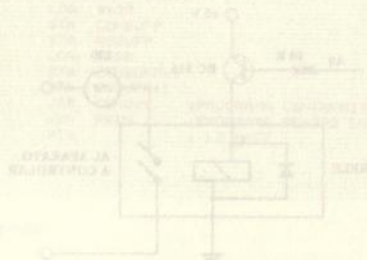
Fig. 6.1. Controlador utilizado por el temporizador.

Aquí se observa el programa encargado de ello:

```

0
;L
1 LDA DRPAR ;CARGA CONTENIDO REG. CONTROL P.A
2 ORA #10000000 ;COLOCA PA7 COMO SALIDA
3 STA DRPAR ;PROGRAMALO
4 LDA #10000000 ;COLOCA VALOR EN SALIDA
5 STA PAR ;FINALIZA
6 RTS
;
    
```


NECESIDADES SOFTWARE



APENDICE A SISTEMAS DE NUMERACION Y SU REPRESENTACION EN EL ORDENADOR

La notación binaria (base 2) es otra forma de expresar los valores de los números. Nuestro sistema usual, el decimal (base 10) usa la combinación de diez dígitos, del cero al nueve. Los números escritos en notación binaria usan sólo dos dígitos, cero y uno. Cada posición ocupada por un dígito binario (un 0 o un 1) se llama bitio (o bit).

Internamente el ordenador representa el 1 por un valor de 5 V (valor alto) y el 0 por un voltaje nulo (valor bajo).



NOTACION DECIMAL

En notación decimal cada dígito en número representa una potencia de 10. Por ejemplo, el número 2408 en notación decimal puede escribirse en forma expandida, así:

$$(2 \times 10^3) + (4 \times 10^2) + (0 \times 10^1) + (8 \times 10^0)$$

Lo que es igual a 2408, como puede verse a continuación:

$$\begin{array}{r} 2 \times 10^3 = 2 \times 1000 = 2000 \\ 4 \times 10^2 = 4 \times 100 = 400 \\ 0 \times 10^1 = 0 \times 10 = 0 \\ 8 \times 10^0 = 8 \times 1 = 8 \end{array} +$$

2408

(Obvio.)

NOTACION BINARIA

En notación binaria la que usa el ordenador internamente, cada dígito representa una potencia de 2. Por ejemplo, el número binario 101101 puede escribirse como:

$$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$$

A continuación están las sucesivas potencias de 2 y su valor decimal:

2^{14}	2^{13}	2^{12}	...	2^3	2^2	2^1	2^0
16384	8192	4096	...	8	4	2	1

El equivalente decimal de 101101 puede calcularse así:

$$1 \times 2^5 = 1 \times 32 = 32$$

$$1 \times 2^4 = 0 \times 16 = 0$$

$$1 \times 2^3 = 1 \times 8 = 8$$

$$1 \times 2^2 = 1 \times 4 = 4 +$$

$$1 \times 2^1 = 1 \times 2 = 2$$

$$1 \times 2^0 = 1 \times 1 = 1$$

45

OPERACIONES LOGICAS

Las operaciones lógicas se hacen bit a bit:

Las reglas para las cuatro operaciones lógicas se dan a continuación:

Operador Regla

AND	Si los dos bits son 1, el resultado es 1. Si algún bit es 0, el resultado es 0.
OR	Si algún bit es 1, el resultado es 1. Si los dos bits son 0, el resultado es 0.
XOR	Si algún bit, pero no los dos, es 1, el resultado es 1. Si los dos bits son 0, el resultado es 0.
NOT	Si el bit es 1, el resultado es 0. Si el bit es 0, el resultado es 1.

Ejemplo:

Cuando se realizan operaciones binarias con los números 77 y 67, éstos son primero convertidos a notación binaria. El número 77 se represen-

ta en 16 bits, como 000000001001101 y el número 67 se representa en 16 bits, como 000000001000001. El resultado de hacer AND, OR y XOR con los dos valores es el siguiente:

	0000.0000.0100.1101
	0000.0000.0100.0001
AND:	0000.0000.0100.0001
OR:	0000.0000.0100.1101
XOR:	0000.0000.0000.1100

La resta de dos números binarios se hace sumando el primero al complementario del segundo.

Para obtener el complemento a 2 de un número binario, se cambia cada 1 por un 0 y cada 0 por 1. Entonces se suma 1 al número obtenido. Por ejemplo, el complemento a 2 de 77 se obtiene como se indica a continuación:

77 en binario	000000001001101
Cambiando bits	111111110110010
Sumando 1	1
-77 en binario	111111110110011

El bit más a la izquierda = 1 significa número negativo, negativo.

Para una descripción más detallada de la aritmética binaria búscuese en un libro sobre la materia.

Ejemplo:

Para convertir un número de notación decimal a binaria se reduce progresivamente el número decimal por la potencia de 2 mayor que no sobrepase al número hasta que éste sea nulo.

El número decimal 77 puede convertirse a notación binaria usando la técnica siguiente.

La potencia de 2 mayor que contiene el número 77 es 64 (2^6). Ponemos un 1 en esa posición del número binario, como se muestra a continuación:

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	0

Restamos a 77 el 64, con lo que tenemos 13. La potencia mayor de 2 que contiene 13 es 8 (2^3), con lo que colocamos un 1 en esa posición. Restamos a 13 el 8 y queda 5. La potencia de 2 mayor que contiene a 5 es 4 (2^2) y colocamos un 1 en su lugar, restamos 5 a 4 y queda 1, con lo que colocamos un 1 en la posición de 2^0 .

El número 77 en notación binaria resulta:

128	64	32	16	8	4	2	1
0	1	0	0	1	1	0	1

Se puede comprobar la exactitud de la conversión así:

$$0.2^7 + 1.2^6 + 0.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 77$$

NOTACION HEXADECIMAL

En notación hexadecimal existen 16 posibles números. Como sólo existen 10 números (0-9), los siguientes números se representan mediante letras. Casualmente un número hexadecimal se representa por cuatro dígitos binarios, por lo que esta notación se utiliza para acortar la representación de cantidades binarias.

Binario	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Por ello, un número binario de 8 bits se representará como dos dígitos hexadecimales:

$$11110000 = F0$$

Para realizar la conversión de decimal a hexadecimal la cosa se complica un poco, ya que debemos ir dividiendo por 16 y tomando los restos; pero como esto es un poco complicado, recomendamos pasar previamente a binario.

La forma de sumar dos números hexadecimales es igual a la utilizada en decimal. Veamos un ejemplo:

$$\begin{array}{r} 3A \\ + BC \\ \hline EC \end{array} \quad \begin{array}{l} A + 2 = B + 1 = C \\ B + 3 = C + 2 = D + 1 = E \end{array}$$

La resta también sería igual.

Queda por indicar que para indicar la base en que se representa cada número existe una notación usada por todos los programas ensambladores:

- Decimal: Se escribe tal cual: 255.
- Binario: Se antecede de %: %1111111.
- Hexadecimal: Se antecede de \$: \$FF.

APENDICE B CONSEJOS A LA HORA DE MONTAR EL CIRCUITO

No se debe olvidar, en primer lugar, que todo circuito integrado debe tener sus patillas de alimentación conectadas a la fuente, aunque no conste en alguna de las figuras de los circuitos. Para ello debe estudiarse detenidamente el patillaje de los mismos del apéndice. Si realizamos ampliaciones por nuestra cuenta, debemos sumar el consumo en mA de todos los chips para comprobar que nuestra fuente es suficiente. En caso contrario, deberíamos sobredimensionarla.

También es aconsejable realizar primeramente un prototipo en esas placas de pinchar componentes (pinchómetro, según los entendidos), fáciles de encontrar en cualquier tienda del ramo y que evitan la engorrosa tarea de desoldar para hacer modificaciones.

Si el lector tiene algo más de práctica en el cacharreo electrónico, puede montar su prototipo, que, si funciona, probablemente haga definitivo en alguna de las placas con pistas pregrabadas y agujereadas que se venden en el mercado. El número de cablecillos será grande, pero si el montaje se hace con orden y ajustando la longitud de los mismos, el resultado puede ser aceptable. (Aunque siempre habrá el riesgo de las capacidades entre cablecillos, que provocarán efectos indeseados.)

Si la cosa se hace más en serio, y se pretende hacer una placa de circuito impreso «elegante», es aconsejable recurrir al tablero de dibujo o utilizar un programa de diseño de los que ya existen para ordenadores personales (Smartwork, AutoCad, etc., para PC y compatibles, por ejemplo). De todas formas, para más información sobre las técnicas al uso es conveniente la consulta de bibliografía sobre el tema, imposible de tratar en un libro de estas dimensiones.

Si se sueldan circuitos integrados directamente, aunque es totalmente aconsejable el uso de zócalos (que, además, facilita la sustitución en caso de fallos), no deben calentarse más que lo que el tacto de cada uno pueda soportar. (Es una buena regla.)

NOTACION HEXADECIMAL

Decimal	Hexadecimal	Binario
0000	0	0
0001	1	1
0010	2	10
0011	3	11
0100	4	100
0101	5	101
0110	6	110
0111	7	111
1000	8	1000
1001	9	1001
1010	A	1010
1011	B	1011
1100	C	1100
1101	D	1101
1110	E	1110
1111	F	1111

$$11110000 = F0$$

Otro tema importante es el de los conectores. Cuanto mejores, más caros, pero más fiables. En nuestro caso no es necesario demasiado desembolso, pero, eso sí, deben de soldarse bien (algo difícil). Es evidente que un conector estándar, por ejemplo el RS-232/V24, no tiene más remedio que cumplir la norma, por lo que será caro. (Relativamente.)

Se aconseja el uso de conectores entre placas si se desea hacer el ordenador de forma modular. Esto permite que con una misma arquitectura básica podamos usar nuestro ordenador para muchas cosas. Por ejemplo, si ponemos un conector de placa a la salida de la VIA y es RIOT, podremos cambiar el tipo de periféricos con sólo cambiar de placa adaptadora y de programa de gestión de los controladores de periféricos, según el caso.

Los dos principales problemas a la hora de la realización práctica de un circuito digital, con los que suele toparse todo el mundo, son el acople y los gliches. Estos dos fantasmas que siempre acechan se pueden evitar. El primero, utilizando pequeños condensadores de desacoplo entre las salidas de los chips digitales de alimentación y masa (se aconseja el valor de unos 100 nF). El segundo, realizando un diseño adecuado. En nuestro caso, la velocidad de reloj es lo suficientemente lenta (1 MHz) para que carezca de importancia.

Cuando vayamos a soldar, debemos utilizar un soldador eléctrico de baja potencia (aproximadamente 15 W), para no dañar el circuito, y estaño de uso electrónico. (Se vende en rollos y lleva núcleo de resina para facilitar su fundición.)

Si se utilizan cables, deben utilizarse siempre lo más cortos posible, para evitar capacidades parásitas.

En el mercado electrónico de componentes los precios son enormemente variables, pudiendo cambiar en casi un 100%, dependiendo del establecimiento. Por ello, es aconsejable ir a varias tiendas antes de hacernos con el material que necesitemos.

Hablemos de la EPROM. Como ya habíamos dicho anteriormente, la EPROM es una memoria de sólo lectura (ROM) que permite ser reprogramada. Esto quiere decir que existen en el mercado unos aparatos especiales para grabarlas. Está claro que sólo es aconsejable comprar (o hacerlos, para los hábiles) un grabador de EPROM, si se graban muchas.

En nuestro caso, que sólo vamos a grabar una, esto sería superfluo. Por ello, debemos buscar algún establecimiento del ramo en el que nos la graben a partir del listado hexadecimal. (Ver apéndice E.)

En algunas tiendas, si compramos en ellas la EPROM, la programan gratis (en Madrid).

Si el lector es de provincias, y está muy desesperado, puede buscar en alguna revista electrónica algún artículo que describa cómo montar el grabador. (Se aconseja experiencia para intentarlo.)

Los chips de la familia CMOS son muy sensibles a las corrientes estáticas (al contrario que los TTL). Esto quiere decir que si tocamos las pat-

llas (terminales) con los dedos, pueden deteriorarse. Por eso suelen venderse en tubos de plástico antiestáticos o envueltos en papel de aluminio. Cuando los vayamos a montar en su zócalo debemos agarrarlos por el cuerpo.

Aconsejamos que antes de alimentar el circuito se realice un repaso concienzudo del montaje para localizar posibles fallos:

— Soldaduras frías: Parece soldado, pero no hace buen contacto eléctrico.

— Cortocircuitos: Entre pistas por gotas de soldadura, etc.

— Circuitos abiertos: Pistas cortadas, etc.

— Mala orientación al enchufar los C.I. (pueden estropearse).

— Otras múltiples razones por las cuales no puede funcionar:

• Aparatos eléctricos cerca (interferencias)

• Falta de alimentación de algún integrado.

• Etc.

Por último, para los que vayan a diseñarse su propio circuito impreso a partir del circuito teórico que proporcionamos, les aconsejamos utilizar métodos fotográficos para realizar la placa (resina fotosensible, copia de papel vegetal, etc.).

Deseamos suerte y aconsejamos paciencia a todos.

APENDICE C FUENTE DE ALIMENTACION

Aún no nos habíamos ocupado de alimentar a nuestro ordenador. Para que los circuitos integrados funcionen deben «enchufarse». La mayoría de ellos van conectados a 5 voltios, aunque los relacionados con el RS-232 irán conectados a ± 12 voltios. Para ello diseñaremos una fuente de alimentación que de los 220 V alterna de la red nos dé 5 y ± 12 voltios de corriente continua.

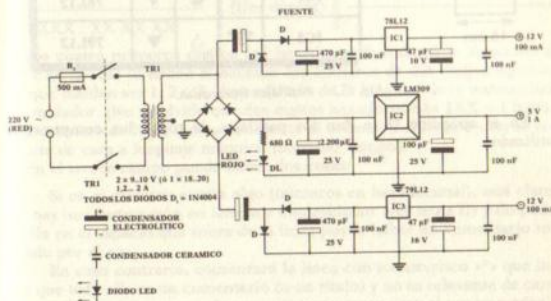


Fig. C.1.

La fuente de 5 voltios proporciona una corriente máxima de 1 amperio, la de ± 12 sólo 100 mA.

Se ha incluido un led (diodo emisor de luz) que estará encendido si la fuente funciona adecuadamente.

No debe olvidarse el colocar un disipador de potencia al circuito integrado LM 309 (CI2), pues de lo contrario éste se sobrecalentaría demasiado, pudiendo llegar a destruirse.

PATILLAJE

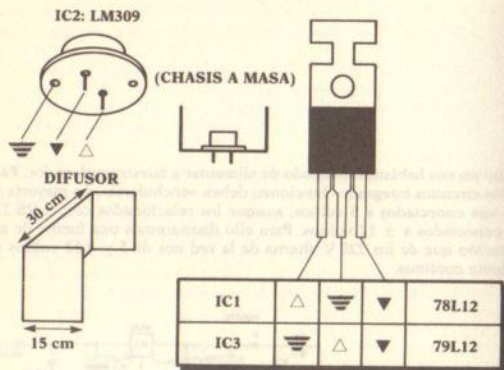


Fig. C.2. Patillajes integrados.

En el apéndice D se dan los patillajes de todos los componentes utilizados.

APENDICE D LISTADO FUENTE DEL PROGRAMA MONITOR

Aquí presentamos el listado fuente del programa monitor. Para los no habituados a este tipo de listados en lenguaje ensamblador diremos lo siguiente:

Las líneas tienen el siguiente formato:

Listado hexadecimal		N.º línea (tres dígitos)		Listado en ensamblador o comentarios
XXXX : XX XX XX				

Los cuatro primeros dígitos (en hexadecimal) indican la dirección a partir de la cual se deben almacenar los números de dos dígitos siguientes, que pueden ser 1, 2 ó 3, dependiendo de la longitud de la instrucción ensamblador. (No se olvide que dos dígitos hexadecimales \$XX = 1 byte).

El número de línea indica solamente el orden, en el listado y no es relevante de cara a lenguaje máquina (segundo campo).

En el tercer campo puede haber dos cosas:

— Si en el primero existía algo (números en hexadecimal), está claro que hay una instrucción en lenguaje ensamblador (ver tema II) y después de ella en el espacio que sobra de la línea puede haber un comentario separado por el «;».

— En caso contrario, comenzará la línea con un asterisco «*» que indica que toda ella es un comentario (o un título) y no es relevante de cara al ensamblado (aunque puede ser muy importante para el programador).

En la zona dedicada a la instrucción en ensamblador hay tres campos:

— El primero (vacío o no) es el campo para la etiqueta (nombre de variable de dirección = 2 bytes).

— El segundo es el dedicado al nemónico (Código de Operación).

— El tercero es el dedicado al operando, que puede ser:

- Una etiqueta (16 bits = 2 bytes = dirección).
- Una variable (8 bits = 1 byte = dirección o dato).
- Un número de 8 a 16 bits en cualquier sistema de numeración que representará dirección o dato según el caso:

n → decimal
 \$n → hexadecimal
 %n → binario
 8n → octal

Al final del listado van dos tablas con los nombres de las variables y etiquetas utilizadas y su valor (? indica que son sólo orientativas)

iASM

```

1 *****
2 *
3 * MONITOR MICRO *
4 * ***** *
5 * *
6 * *
7 *****
8 *
9 * VARIABLES DEL SISTEMA
10 * INCLUIDA EN PAG. 0
11 *
12 XREG EQU $0001
13 YREG EQU XREG+1
14 STATUS EQU YREG+1
15 STKP EQU STATUS+1
16 ACC EQU STKP+1
17 DIRL EQU ACC+2
18 DIRH EQU DIRL+1
19 MAXDP EQU DIRH+2
20 MINBF EQU MAXDP+1
21 OUTBF EQU MINBF+1
22 AUX EQU OUTBF+6
23 AUX1 EQU AUX+1
24 USRG EQU AUX1+1
25 CENFLAG EQU USRG+1
26 CENBUF EQU CENFLAG+1
27 RSFLAG EQU CENBUF+2
28 RSBUF EQU RSFLAG+1
29 FUNC EQU RSBUF+2
30 IRROR EQU FUNC+2
31 HRC EQU IRROR+2
32 HRS10 EQU HRS+1
33 MIN EQU HRS10+1
34 MIN10 EQU MIN+1
35 SECS EQU MIN10+1
36 SECS10 EQU SECS+1
37 HRSF EQU SECS10+1
38 MINF EQU HRSF+1
39 SECF EQU MINF+1
40 USREL EQU SECF+1
41 INTCON EQU USREL+2
42 *
43 * DEFINICION DE LOS VALORES
44 * DE LAS TECLAS DE FUNCION
45 *
46 MAS EQU $10
47 MENOC EQU $11

```

```

48 RUN EQU $12
49 DIR EQU $13
50 DAT EQU $14
51 *
52 * DIRECCION DE LOS REGISTROS
53 * DEL RIOT
54 *
55 PADR EQU $2080 ;DIRECCION PORT A RIOT
56 PADOR EQU $2081 ;DIRECCION PROGRAMAR PORT A RIOT
57 PBDR EQU $2082 ;DIRECCION PORT B RIOT
58 PBDOR EQU $2083 ;DIRECCION PROGRAMAR PORT C RIOT
59 PPNCSI EQU $2084 ;FLANCO NEGATIVO SIN INTER.
60 PFPPI EQU $2085 ;FLANCO POSITIVO SIN INTER.
61 PPNCCI EQU $2086 ;FLANCO NEGATIVO CON INTER.
62 PFPCCI EQU $2087 ;FLANCO POSITIVO CON INTER.
63 T1S1 EQU $2094 ;REG CONTADOR 1 SIN INTER.
64 T8S1 EQU $2095 ;REG CONTADOR 8 SIN INTER.
65 T44S1 EQU $2096 ;REG CONTADOR 44 SIN INTER.
66 T1024S1 EQU $2097 ;REG CONTADOR 1024 SIN INTER.
67 T1C1 EQU $209C ;REG CONTADOR 1 CON INTER.
68 T8C1 EQU $209D ;REG CONTADOR 8 CON INTER.
69 T44C1 EQU $209E ;REG CONTADOR 44 CON INTER.
70 T1024C1 EQU $209F ;REG CONTADOR 1024 CON INTER.
71 *
72 * DIRECCION DE LOS REGISTROS
73 * DE LA VIA
74 *
75 PBU EQU $2100 ;PORT B DE VIA
76 PAW EQU $2101 ;PORT A DE VIA CON HANDSHAKE
77 DBU EQU $2102 ;PROGRAMACION PORT B DE VIA
78 DBW EQU $2103 ;PROGRAMACION PORT A DE VIA
79 CT1BU EQU $2104 ;PROGRAMACION BAJA T1
80 CT1AW EQU $2105 ;PROGRAMACION ALTA T1
81 RT1B EQU $2106 ;REGISTRO T1 BAJA
82 RT1A EQU $2107 ;REGISTRO T1 ALTO
83 RT2B EQU $2108 ;REGISTRO T2 BAJA
84 RT2A EQU $2109 ;REGISTRO T2 ALTA
85 SRC EQU $210A ;REGISTRO DE PLAZAMIENTO
86 ACRV EQU $210B ;REGISTRO ALC. DE CONTROL
87 PCRV EQU $210C ;REGISTRO DE CONTROL
88 IFRU EQU $210D ;REGISTRO ALARMA INTER.
89 IERR EQU $210E ;REGISTRO ACTIVACION INTER.
90 PASV EQU $210F ;PORT A SIN HANDSHAKE
91 *
92 * COMIENZO DEL CONTENIDO
93 * DE LA EPROM
94 *
95 * ORG $F800
96 *
97 * RESERVA DE ESPACIO PARA
98 * LAS RUTINAS DEL
99 * USUARIO
100 *
101 * LAS RUTINAS DEL SISTEMA NO OCUPAN MAS DE LA
102 * MITAD DE LA CAPACIDAD DE LA EPROM POR LO QUE
103 * LA PARTE INFERIOR DE LA MISMA ESTA LIBRE
104 * PARA LAS APLICACIONES DEL USUARIO
105 *
106 * DS $406
107 *
108 * TABLA DE LAS DIRECCIONES
109 * DE LAS FUNCIONES DEL
110 * EDITOR
111 *
112 TABLA DFB $HMSI/256
113 OFD $HMSI
114 OFB $HNSI/256
115 OFD $HNSI
116 OFB $MHI/256

```

```

FCB0: DE 117 DFB #RUI)
FCB1: FD 118 DFB #DIRI/256
FCB2: FD 119 DFB #DATO/256
FCB3: 0A 120 DFB #DIRI)
FCB4: 0D 121 DFB #DATO)
122 *
123 * DEFINICION DE REPRESENTACION
124 * DE CARACTERES EN EL
125 * DISPLAY
126 *

FC90: 40 79 24 127 CODE DFB #40,879,824,800 1'0','1','2','3'
FC94: 19 12 02 128 DFB #19,812,802,878 1'4','5','6','7'
FC97: 28 03 00 129 DFB #00,810,808,803 1'8','9','A','B'
FC9C: 46 24 00 130 DFB #46,824,806,80E 1'C','D','E','F'
FC9F: 0E 09 00 131 DFB #0E,805,800,80C 1'A','X','Y','P'
FDA3: 0C 132 DFB #12,87F 1'8','9'
FC94: 12 7F 133 *****
134 *
135 * Rutina de RESEY *
136 *
137 *****
FC96: 08 138 RES RHP
FDA7: 05 05 139 STA ACC
FC9V: 86 01 140 STX XREG ; SALVA LOS REGISTROS DEL USUARIO
FC9B: 04 02 141 STY YREG
FC9D: 68 142 PLA ;SACA DIRECCION DEL PROGRAMA
FC9E: 03 03 143 STA STATUS ;YA EL STATUS DEL STACK
FCB0: 68 144 PLA
FCB1: 05 07 145 STA DIRL
FCB3: 68 146 PLA
FCB4: 05 08 147 STA DIRH
FCB5: A7 0F 148 LDA #00F ;PROGRAMA PORT D RIOT
FCB8: 00 09 20 149 STA #B008
FCB1: 08 150 CLD ;CALCULA EN BINARIO
FCB3: 08 151 SET ;INHIBE INTERRUPCIONES
152 *****
153 *
154 * BUCLE PRINCIPAL *
155 *
156 *****
FCB0: 20 15 FE 157 ENTRADA JSR L0AT
FC01: 20 07 FD 158 JMP LEEKB ;LEE TECLA
FCC3: C9 10 159 ENTI ORP #910 ;SI ES UN NUMERO
FCC5: 30 F4 160 STA ENTRADA ;IGNORALO
FCC7: C9 14 161 ORP #614 ;SI ERROR
FCC9: 10 F2 162 BPL ENTRADA ;IGNORALO
FCB1: 38 163 SEC
FCB3: C9 19 164 SBC #610 ;TRANSFORMAR TECLA
FCC2: 0A 165 ASL ;EN VALOR
FCC3: 0A 166 TAX ;PARA CALCULAR DIRECCION
FC01: 80 86 FC 167 LDA TABLA,X ;DE LA FUNCION
FCC3: 0A 168 STA FUNC
FCD5: 18 169 INCI
FCD6: 80 86 FC 170 LDA TABLA,X
FCD9: 95 1C 171 STA FUNC+1
FCD8: 4C 18 00 172 RFP -(FUNC) ;EJECUTA LA FUNCION
173 *****
174 *
175 * Rutina de ATENCION *
176 *
177 * A RUN *
178 *
179 *****
FCDE: A6 04 180 RUI) LDY STKP ; RESTAURA LOS REGISTROS

```

```

FC02: 0A 181 TSX
FC01: A5 08 182 LDA DIRH
FC03: 40 183 PHA
FC04: A5 07 184 LDA DIRL
FC06: 48 185 PHA
FC07: A5 03 186 LDA STATUS
FC08: 48 187 PHA
FC0A: A4 01 188 LDY XREG
FC0C: A4 02 189 LDY YREG
FC0E: A5 05 190 LDA ACC
FC0F: 40 191 RTI ; EJECUTA LA RUTINA DEL USUARIO
192 *****
193 *
194 * Rutina de ATENCION *
195 *
196 * A *
197 *
198 *****
FCF1: 10 199 MMS1 CLC ;INCREMENTA DIRECCION USUARIO
FCF2: A5 07 200 LDA DIRL
FCF4: 3F 01 201 ADC #501
FCF5: 85 07 202 STA DIRL
FCF6: 90 02 203 BCC MMS2
FCFA: E6 08 204 ITC DIRH
FCFC: 4C 80 FC 205 MMS2 JMP ENTRADA ;IR BUCLE PRINCIPAL
206 *****
207 *
208 * Rutina de ATENCION *
209 *
210 * A - *
211 *
212 *****
FCFF: 38 213 MENOS1 SEC ; DECREMENTA DIRECCION USUARIO
FD00: A5 07 214 LDA DIRL
FD02: E9 01 215 SBC #601
FD04: 85 07 216 STA DIRL
FD06: 90 02 217 BCC MENOS2
FD08: C4 08 218 DEC DIRH
FD0A: 4C 80 FC 219 MENOS2 JMP ENTRADA ;VOLVER BUCLE PRINCIPAL
220 *****
221 *
222 * Rutina de ATENCION *
223 *
224 * A DAT *
225 *****
FD00: A0 13 226 DATO LDY #613 ;PREPARA BUFFER
FD0F: A2 04 227 DAT1 LDY #604
FD11: 86 08 228 STX MINIBF
FD13: A9 06 229 LDA #605
FD15: 05 0A 230 STA MAIBF
FD17: 20 9A FD 231 JSR LEDAT ;LEE LOS DATOS
FD1A: A2 04 232 LDY #604
FD1C: 20 90 FD 233 JSR ALKI ;LOS CONVIERTE EN UN BYTE
FD1F: C0 15 234 CPY #613
FD21: F0 06 235 BEQ DAT2
FD23: 9F 01 00 236 STA XREG,Y
FD26: 18 237 CLC
FD27: 90 04 238 BCC DATFIN
FD29: A2 00 239 DAT2 LDY #0 ;ALMACENA DATO
FD2B: B1 07 240 STA (DIRL,X)
FD2D: 4C 80 FC 241 DATFIN JMP ENTRADA
242 *
243 * Rutina AUXILIAR de DATO *
244 *
FD30: 05 0C 245 AUX1 LDA OUTBF,X ;CONVIERTE DOS NUMEROS
FD32: 0A 246 ASL ;DE CUATRO BITZ EN
FD33: 0A 247 ASL ;UNO DE OCHO.
FD34: 0A 248 ASL
FD35: 0A 249 ASL

```



```

FD361 E8 250 I/OX
FD371 15 0C 251 I/OX OUTBF,X
FD391 40 252
253 *****
254 *
255 * RUTINA DE ATENCION *
256 *
257 * A DIR *
258 *
259 *****
FD3A1 20 D7 FD 240 DIR1 JSR LEEKB ILEE TECLA
FD301 C9 10 241 CHP #B10
FD3F1 30 06 242 BHI DIREC IVER SI ES DAT
FD411 C5 14 243 CHP DAT IVA A DIRDAT
FD401 00 95 244 BNE D1R1 IPROGRAMA EL BUFFER DE SALIDA
FD451 F0 1B 245 BEQ DIRDAT
FD471 A2 00 246 DIREC LDX #B00
FD491 96 0B 247 STX HINBF
FD481 AF 04 248 LDA #B04
FD4D1 85 0A 249 STA MAXBF
FD4F1 20 9A FD 270 JSR LEDAT ILEE DIRECCION ALMACENADA
FD521 AF 00 271 LDA #B00
FD041 20 12 00 272 JSR AUX
FD571 85 0B 273 STA DIRH
FD591 E8 274 INX
FD5A1 20 12 00 275 JSR AUX
FD5D1 85 07 276 STA DIRL
FD5F1 4C 8D FC 277 JMP ENTRADA IVA AL BUCLE PRINCIPAL
278 *****
279 *
280 * RUTINA DE ATENCION *
281 *
282 * A DIR DAT *
283 *
284 *****
FD621 20 D7 FD 285 DIRDAT JSR LEEKB ILEE TECLA
FD651 09 0A 286 CHP #B0A
FD671 03 03 287 BNE D0AT1 IVER SI ES EL ACUMULADOR
FD691 A9 04 288 LDA #B04
FD6B1 A8 289 TAY
FD6C1 AF 7F 290 D0AT1 LDA #B7F ICOLOCA ESPACIOS VACIOS
FD6E1 85 12 291 STA AUX IEN EL BUFFER DE SALIDA
FD701 85 0C 292 STA OUTBF
FD721 85 0D 293 STA OUTBF+1
FD741 85 0E 294 STA OUTBF+2
FD761 85 0F FC 295 LDA CODE,Y
FD791 85 0F 296 STA OUTBF+3
FD7B1 A2 04 297 LDX #B04
FD7D1 20 2C FE 298 JSR GUARDA ILO ESCRIBE
FD801 20 87 FD 299 JSR ESCRIBE
FD831 20 D7 FD 300 JSR LEEKB ILEE TECLA
FD841 C5 14 301 CHP DAT IVE SI ES DAT
FD8B1 F0 0B 302 BEQ D0AT2
FD8A1 A8 303 TAY
FD8B1 20 15 FE 304 JSR LOAT
FD8E1 48 305 PLA ICONVIERTE DATOS PARA DISPLAY
FD8F1 4C C3 FC 306 JMP ENTI
FD921 A4 12 307 D0AT2 LDX AUX IIR A C EJECUTAR DIRDAT
FD941 20 0F FD 308 JSR DAT1
FD971 4C 8D FC 309 JMP ENTRADA
310 *****
311 *
312 * LEE UNA DIRECCION *
313 *
314 * O UN DATO DEL *
315 *
316 * TECLADO *
317 *
318 *****

```

```

FD9A1 A4 0B 319 LEDAT LDX HINBF
FD9C1 20 D7 FD 320 LEI JSR LEEKB ILEE TECLA
FD9F1 C5 11 321 CHP MENOS ICOMPRUEBA SI ES MENOS
FDA11 F0 09 322 BEQ BK
FDA31 95 0C 323 STA OUTBF,X
FDA51 20 87 FD 324 JSR ESCRIBE
FDA81 E8 325 INX
FDA91 E4 0A 326 CPX MAXBF
FDAB1 40 327 RTS
FDAC1 AF 7F 328 BK LDA #B7F ISI ES MENOS
DAE1 95 0C 329 STA OUTBF,X IRETROCEDE UNO MAS
FD801 CA 330 DEX
FD811 E4 0B 331 CPX HINBF
FD831 F0 E5 332 BEQ LEDAT
FD851 0C E5 333 BNE LEI
334 *****
335 *
336 * RUTINA DE ESCRITURA *
337 *
338 * EN EL DISPLAY *
339 *
340 *****
FD871 AF 7F 341 ESCRIBE LDA #B7F IPROGRAMA PORT A
FD891 8D 01 20 342 STA PADR
FD8C1 A2 05 343 LDX #B03 IPREPARA BUFFER
FD8E1 84 0C 344 ESCI LDX OUTBF,X ILEE CODIGO
FD001 8F 90 FC 345 LDA CODE,Y ICARGA CODIGO DISPLAY
FD031 8D 02 20 346 STA PADR IESCRIBELO
FD041 A8 7F 347 UP LDY #B7F IESPERA UN RATO
FD061 8B 348 DEY
FD0F1 10 FB 349 SPL UP
FD0B1 8C 00 20 350 STY PADR IREPROGRAMA PORT A
FD0E1 AF 04 351 LDA #B06
FD081 8D 02 20 352 STA PBDR IREPROGRAMA PORT B
FD031 CA 353 DEX
FD041 10 E8 354 SPL ESCI IFIN ?
FD041 40 355 RTS
356 *****
357 *
358 * LECTURA DE UNA *
359 *
360 * TECLA *
361 *
362 *****
FD071 AF 00 363 LCCIB LDA #B00 IPROGRAMA PORT
FD091 8D 01 20 364 STA PADR
FD0C1 A2 06 365 LDX #B06
FD0E1 BE 82 20 366 STX PBDR
FD0F1 AD 80 20 367 LEI1 LDA PADR
FD041 D0 0B 368 BNE SI
FD061 E8 369 INX
FD071 E0 09 370 CPX #D09
FD091 D0 F4 371 BNE LCE1
FD0B1 20 87 FD 372 JSR ESCRIBE
FD0E1 4C 07 FD 373 JMP LEEKB
FD0F1 84 13 374 B1 STX AUX1
FD0F1 48 375 TAY
FD0F1 20 87 FD 376 JSR ESCRIBE
FD0F1 48 377 PLA
FD0F1 A4 13 378 LDX AUX1
FD0A1 A8 7F 379 JSR LEEKB ILEE TECLA
FD0C1 98 380 DEY
FD0D1 10 FB 381 SPL LEE2
FD0F1 AD 80 20 382 LDA PADR
FD0F1 95 12 383 STA AUX2
FD041 8A 384 TAY
FE01 38 385 SEC
FE01 E9 04 386 SBC #B04
FE001 F0 0B 387 BEQ LEFIN

```

```

FE0A: 18 388 LEE3 CLC
FE0B: A5 04 389 LDR #004
FE0D: A5 12 390 ACC #004
FE0F: CA 391 DEX
FE10: 06 F8 392
FE12: A5 12 393 LEP/HL LDR #004
FE14: 40 394 RTS
FE15: 40 395 *****
FE16: * 396 * *
FE17: * 397 * GARGA DATOS EN *
FE18: * 398 * *
FE19: * 399 * BUFFER DEL DISPLAY *
FE20: * 400 * *
FE21: * 401 *****
FE25: A5 08 402 LDAT LDR DIRH (ALMACENA PARTE ALTA Y
FE27: A2 00 403 LDX #000 (BAJA DE DIRECCION EN EL
FE29: 20 2C FE 404 JSR GUARDA (BUFFER DE SALIDA
FE2C: A5 07 405 CLR DIRL
FE2E: 20 2C FE 406 JSR GUARDA
FE31: A0 00 407 LDV #0
FE32: B1 97 408 LDR (DIRL),Y (GARGA CONTENIDO DIRECCION
FE33: 20 2C FE 409 JSR GUARDA
FE36: 20 B7 FD 410 JSR ESCRIBE
FE38: 40 411 RTS
FE39: * 412 *
FE3A: * 413 * Rutina Auxiliar de LDAT
FE3B: * 414 *
FE3C: 40 415 GUARDA PHA (COMPARTE NUMERO
FE3D: 2F F0 416 AND #0F0 (DE OCHO BITS
FE3F: 0A 417 ASL (EN DOS DE CUATRO BITS
FE39: 0A 418 ASL
FE31: 0A 419 ASL
FE32: 0A 420 ASL
FE33: 95 0C 421 STA OUTBF,X
FE35: E8 422 INX
FE36: 08 423 PLA
FE37: 27 0F 424 AND #0F
FE39: 95 0C 425 STA OUTBF,X
FE38: E8 426 INX
FE3C: 40 427 RTS
FE3C: 40 *****
FE39: * 428 * *
FE39: * 430 * PROG. CENTRONICS *
FE39: * 431 * *
FE3C: 40 *****
FE39: * 433 * *
FE39: * 434 * PROGRAMACION DEL PORT A
FE39: * 435 * COMO ENTRADA/SALIDA
FE39: * 436 * INTERFACE CENTRONICS
FE39: * 437 *
FE39: * 438 * PARA LEER EL PUNTO DE ENTRADA
FE39: * 439 * ES INCENT
FE39: * 440 *
FE39: * 441 * PARA ESCRIBIR ES OUTCENT
FE39: * 442 *
FE3D: A9 00 443 INCENT LDR #0 (ROUTINA DE ENTRADA
FE3F: 00 03 21 444 STA DRAY (PROGRAMA REG. DIRECCION
FE42: 85 15 445 STA CENFLAG
FE44: AC 50 FE 446 JHP CONT
FE47: A9 FF 447 OUTCENT LDR #0FF (ROUTINA SALIDA
FE49: 8C 03 21 448 STA DRAY (PROGRAMA REG. DIRECCION
FE4C: A9 01 449 LDR #01
FE4E: 95 15 450 STA CENFLAG
FE50: A9 00 451 CONT LDR #0 (PARTE COMUN
FE52: 8C 01 21 452 STA DRAY (REGISTRO FLAGS INT.
FE55: A0 0C 21 453 LDR PCRV (REGISTRO DE CONTROL
FE58: 2F F0 454 AND #C1110000
FE5A: 0F 08 455 ORA #C0000100
FE5C: 80 0C 21 456 STA PCRV

```

```

FE5F: A0 0B 21 457 *
FE62: 2F FE 458 LDR ACRV (REGISTRO AUXILIAR DE CONTROL
FE64: 0F 01 459 AND #C1111111
FE66: 80 0B 21 460 ORA #C0000001
FE67: * 461 * STA ACRV
FE69: A9 83 462 *
FE6B: 80 0E 21 463 LDR #C1000011
FE6C: 80 0E 21 464 ORA IERV (ACTIVACION DE INTER.
FE6E: 80 0E 21 465 STA IERV
FE71: A9 00 466 LDR #0
FE73: 80 0D 21 467 STA IFRV
FE76: 58 468 CLI
FE77: 40 469 RTS
FE78: * 470 *****
FE78: * 471 * *
FE78: * 472 * PROG. RS = 232/A 24 *
FE78: * 473 * *
FE78: * 474 *****
FE78: * 475 * INV. DOS PUNTOS DE ENTRADA
FE78: * 476 * INRS PARA LA ROUTINA DE ENTRADA
FE78: * 477 * Y OUTRS PARA LA DE SALIDA
FE78: * 478 *
FE78: A0 0B 21 479 INRS LDR ACRV (PROGRAMA EL REGISTRO AUXILIAR
FE7B: 2F E1 480 AND #C1110001 (DE CONTROL
FE7D: 69 0E 481 ORA #C0000110
FE7F: 80 0B 21 482 STA ACRV
FE82: A0 0C 21 483 LDR PCRV (IDEM REGISTRO DE CONTROL
FE85: 2F 0F 484 AND #C0000111
FE87: 69 0E 485 ORA #C1000000
FE89: 80 0C 21 486 STA PCRV
FE8C: A9 00 487 LDR #0 (RFLAG#0 => ENTRADA
FE8E: 85 18 488 STA RFLAG
FE90: 4C AB FE 489 JHP CONT
FE93: A0 0B 21 490 OUTRS LDR ACRV (IDEM REGISTRO AUXILIAR DE
FE96: 2F E1 491 AND #C1110001 (CONTROL
FE98: 69 1E 492 ORA #C0001110
FE9A: 80 0B 21 493 STA ACRV
FE9D: A0 0C 21 494 LDR PCRV (IDEM REGISTRO DE CONTROL
FEA0: 2F 0F 495 AND #C0000111
FEA2: 69 0E 496 ORA #C0000000
FEA4: 80 0C 21 497 STA PCRV
FEA7: A9 01 498 LDR #1 (RFLAG#1 => SALIDA
FEA9: 85 18 499 STA RFLAG
FEAB: A9 09 500 LDR #9 (CONTI
FEAC: 80 0D 21 501 STA IFRV (ACTIVANDO LA INTERRUPCION
FEAD: A0 0E 21 502 LDR IERV (CORRESPONDIENTE
FEAE: 09 0A 503 ORA #C1000010
FEB0: 80 0E 21 504 STA IERV
FEB2: 58 505 CLI
FEB3: 40 506 RTS
FEB4: * 507 *****
FEB4: * 508 * *
FEB4: * 509 * Rutina de Atencion a *
FEB4: * 510 * *
FEB4: * 511 * IRQ *
FEB4: * 512 * *
FEB4: * 513 *****
FEBA: A0 0D 21 514 IRQD LDR IFRV (IRQD ES LA ROUTINA QUE
FEBC: 2F 80 515 AND #C1000000 (ATENDE A LA INTERRUPCION
FEBD: F0 32 516 BEQ IROI (DE CENFLAG REALIZA ENTRADA
FECE: A0 0D 21 517 LDR IFRV (DE CENFLAG REALIZA ENTRADA
FECD: 2F 82 518 AND #C0000010 (DEPENDIENDO DEL CONTENIDO
FECE: F0 2B 519 BEQ IROI (DE CENFLAG REALIZA ENTRADA
FEC8: A5 15 520 LDR CENFLAG
FEC9: F0 14 521 BEQ CENIN
FECB: A5 16 522 CENOUT LDR CENBUFF (CENBUFF ES UN VECTOR QUE
FECD: F0 23 523 BEQ IROI (DE CENFLAG REALIZA ENTRADA
FED0: 54 14 524 DEC CENBUFF (INDICA EL INICIO DEL
FED2: 8A 525 TIA (BUFFER DEL INTERFACE

```



```

FE03: 48      524      PWA
FE04: A2 00   527      LDX #0
FE05: A1 16   528      LDA (CENBUFF,X)
FE06: 80 01 21 529      STA PWA
FE0B: 48      530      PLA
FE0C: BA      531      TXA
FE0D: 4C F9 FE 532      JHP IRO1
FE0E: A3 16   533      LDA CENBUFF
FE0F: C9 FF   534      CMP #FF
FE14: F0 00   535      BEQ IRO1
FE15: E4 16   536      JMC CENBUFF
FE16: 8A      537      TXA
FE17: BA      538      PWA
FE1A: A2 00   539      LDX #0
FE1C: A0 01 21 540      LDA PWA
FE1F: B1 16   541      STA (CENBUFF,X)
FE21: 68      542      PLA
FE22: AA      543      TAX
FE23: AD 0D 21 544      IRO1 LDA IFRV ;IRO1 ES LA RUTINA QUE
FE24: 3F 80   545      AND #C0000000 ;ATIENDE A LA INTERRUPCION
FE25: F0 92   546      BEQ IRO2 ;DEL INTERFACE RS - 232
FE2A: A0 0D 21 547      IRO1 LDA IFRV ;IRO1 ES LA RUTINA QUE
FE2B: 2F 1C   548      AND #C0001100 ;RSFLAG INDICA LA DIRECCION
FE2C: F0 98   549      BEQ IRO2 ;DE TRANSMISION (E/S)
FE2D: A5 18   550      LDA RSFLAG
FE2E: F0 14   551      CEG RSIN ;LA RUTINA ES SIMILAR A
FE2F: A5 19   552      LDA RSBUFF ;LA DEL INTERFACE CENTRONICS
FE30: F0 25   553      CEG IRO2
FE31: C6 19   554      DEC RSBUFF
FE32: BA      555      TXA
FE33: 48      556      PWA
FE34: A2 00   557      LDX #0
FE35: A1 19   558      LDA (RSBUFF,X)
FE36: B0 0D 21 559      STA (RSV)
FE37: 48      560      PLA
FE38: AA      561      TAX
FE39: 4C 2C FF 562      JHP IRO2
FE3A: A5 19   563      LDA RSBUFF
FE3B: C9 FF   564      CMP #FF
FE3C: F0 00   565      BEQ IRO2
FE3D: E4 19   566      JMC RSBUFF
FE3E: BA      567      TXA
FE3F: 48      568      PWA
FE40: A2 00   569      LDX #0
FE41: A0 0D 21 570      LDA PWA
FE42: B1 19   571      STA (RSBUFF,X)
FE43: 68      572      PLA
FE44: BA      573      TAX
FE45: 4C 10 00 574      IRO2 JHP (IRUSR) ;IRUSR ES EL VECTOR DE
FE46: * COMIENZO DE RUTINA IRO DEL UCR
FE47: *****
FE48: * RUTINA DE CONTROL DEL *
FE49: * TEMPORIZADOR *
FE4A: * *
FE4B: *****
FE4C: * RUTINA DE APLICACION DE LDS
FE4D: * TEMPORIZADORC PARA LA *
FE4E: * ACTIVACION DE DISPOSITIVOS *
FE4F: *****
FE50: * EXTICIONES
FE51: A9 FF   586      INREL LDA #IREL/256 ;COLOCA VECTOR USUARIO
FE52: 95 1E   587      STA (IRUCR)
FE53: A0 BA FF 588      LDA IREL
FE54: B5 1D   589      STA IRUSR
FE55: AD E4 21 590      LDA IERV ;INICIALIZA CONTADOR I DE VIA
FE56: 99 00   591      ORA #C0
FE57: 8D DE 21 592      STA IERV
FE58: A0 DB 21 593      LDA ACRV
FE59: 09 46   594      ORA #46

```

```

FF45: 80 08 21 595      STA ACRV
FF46: 58      596      CLI
FF47: A9 22   597      LDA #22 ;CARGA EL RELOJ CON CUENTA
FF48: 80 04 21 598      STA CTIBV ;1/16 DE SEGUNDO
FF49: A9 F4   599      LDA #F4
FF50: 8D 05 21 600      STA CTIAV
FF51: A2 00   601      LDX #0
FF52: 0E 00   602      STX HINDP
FF53: A9 04   603      LDA #05
FF54: 8D 04   604      STA MAXBF
FF55: 20 9A F0 605      JSR LEADT
FF56: 20 19 00 606      JCR AUX1
FF57: 85 1F   607      STA HRS
FF58: 20 13 00 608      JCR AUX1
FF59: 85 21   609      STA MIN
FF60: 20 10 00 610      JCR AUX1
FF61: 85 23   611      STA SECS
FF62: A2 00   612      LDX #0
FF63: 86 00   613      STX HINBF ;LDER LA HORA DE ACTUACION
FF64: A9 06   614      LDA #06
FF65: 85 0A   615      STA MAXBF
FF66: 20 9A F0 616      JSR LEADT
FF67: 85 25   617      STA HRS
FF68: 20 13 00 619      JCR AUX1
FF69: 85 26   620      STA MINP
FF70: 20 13 00 621      JCR AUX1
FF71: 85 27   622      STA SECS
FF72: 4C A6 FC 623      JHP RES ;REGRESA AL BUCLE PRINCIPAL
FF73: *****
FF74: * *
FF75: * RUTINA DE ATENCION INTE- *
FF76: * RRUPCION DE APLICACION *
FF77: * DEL TEMPORIZADOR *
FF78: * *
FF79: *****
FF80: 48      630      PWA
FF81: BA      631      TXA
FF82: 48      632      PWA
FF83: C4 2A   633      DEC INTCT ;COMPRUEBA FIN DE CUENTA
FF84: D0 40   634      BNE FINTR ;ACTUALIZANDO LOS CONTADORES
FF85: A9 18   635      LDA #14 ;DE HORA,MINUTOS Y SEGUNDOS
FF86: 85 2A   636      STA INTCT
FF87: A9 30   637      LDA #30
FF88: E4 23   638      JMC SECS
FF89: A2 3A   639      LDA #3A
FF90: E4 23   640      JMC SCCS
FF91: D0 3F   641      BNE FINTR
FF92: 85 23   642      STA SECS
FF93: E4 24   643      JMC SECS10
FF94: D0 3B   644      BNE FINTR
FF95: 85 24   645      STA SECS10
FF96: E4 24   646      JMC MIN
FF97: A2 3A   647      LDA #3A
FF98: E4 24   648      JMC SECS10
FF99: E4 21   649      JMC MIN
FFA0: A2 3A   650      LDA #3A
FFA1: E4 21   651      JMC MIN
FFA2: D0 3B   652      BNE FINTR
FFA3: 85 21   653      STA MIN
FFA4: E4 22   654      JMC INH10
FFA5: A2 3A   655      LDA #3A
FFA6: E4 22   656      JMC MIN10
FFA7: 56 1F   657      STA INH10
FFA8: 85 22   658      STA HRS
FFA9: 56 1F   659      STA HRS
FFAB: E4 1F   660      LDA #3A
FFAC: D0 07   661      BNE OTRODIA
FFAD: 95 1F   662      STA HRS

```



```

FFC9: E6 20 644 INC HR510
FFCB: 4C DE FF 645 JMP FINTR
FFCE: A2 34 556 OTRDIA LDX #B34
FFD0: E4 1F 647 CFC HRS
FFD2: 00 0A 648 BNE JINTR
FFD4: A2 32 649 LDX #B32
FFD6: E4 20 650 CFC HR510
FFD8: 00 04 651 BNE FINTR
FFDA: 95 20 652 STA HR510
FFDC: 85 1F 653 STA HRS
FFDE: A5 23 654 FINTR LDA SECS ;COMPROBAR SI ES LA HORA
FFE0: 85 29 655 CMC SECSP ;PROGRAMADA
FFE2: 00 0F 656 BNE F1
FFE4: A5 21 657 LDA MIN
FFE6: C5 26 658 CMP HRP
FFEB: D0 09 659 BNE F1
FFEA: A5 1F 660 LDA HRS
FFEC: C5 25 661 CMC HRP
FFED: 00 03 662 BNE F1
FFF0: 4C 20 00 663 JPP (USREL) ;EJECUTA FUNCION USUARIO
FFF3: A0 D4 21 664 F1 LDA CTIBU ;RESTAURA FLAG INTER.
FFF5: 68 685 PLA ;RESTAURA REGISTROS
FFF7: 44 686 RAV
FFF8: 40 687 PLA
FFF9: 40 688 COO
689 *
690 * DIRECCIONES DE INTERRUPCIONES
691 *
FFFA: A6 692 MHI DFB #RES
FFFB: FC 693 DFB #RES/256
FFFC: A6 694 RESET DFB #RES
FFFD: FC 695 DFB #RES/256
FFFF: BA 696 IRQ DFB #IRQ0
FFFF: FE 697 DFB #IRQ0/256

```

--End assembly--

2048 bytes

Errors: 0

Symbol table - alphabetical order:

```

ACC #B05 ACRV #B210B ALX #B12 ALU1 #B10
AUX1 #F030 BK #F0DC CENBUFF #B12 CENFLAG #B12
CENIN #FEE0 ? CENOUT #F0DC CODE #F0C0 CONT #F850
CONT #FEEB CTIAU #B2105 CTIBU #B2104 DAT #B14
DAT1 #F04E DAT2 #F029 DATF #F020 DATD #F000
DATI #F04C DATY #F029 DIR #B13 DIR1 #F03A
DIRDAT #F062 DIREC #F047 DIRH #B08 DIRL #F000
DRWV #B2103 DRBV #B2102 EN1 #F0CC ENTRADA #F0B0
ESC1 #F0B8 ESCRIBE #F067 EN2 #F0F3 FINTR #F0FE
FUNC #B18 GUARDA #F02C HRS #B210D HR510 #B20
HRSP #B25 IERV #B210E IFRV #B210D HR510 #B20
INREL #F0F2F ? INRS #F078 INTOUT #B2A IREL #F0E0
IRQ #F0FE IRQ0 #F0E8 IRO1 #F0F9 IRO2 #F02C
IRUSR #B10 LOAT #F0E5 LEI #F09C LEDAT #F0D8
LEE1 #F0E1 LEE2 #F0FA LEE3 #F0EA LEEB #F0D7
LEFIN #F0E2 LEHORA #F0F3 ? MAS #B10 MASI #F0CF
MHI #F0FC MHI1 #B21 MIN #B21 MENOS #B11 MENOS1 #F0CF
MINP #B2a ? MHI #FFFF OTRDIA #F0E2 MINDF #B0B
? OUTENT #F0E4 ? OUTRS #F0F3 PADD #B20E1 PADR #B200
? PASV #B210F PAW #B2101 PBDR #B20B3 PBDR #B20B2
PBV #B2100 PCRU #B210C ? PFNCI #B20B6 ? PFI01 #B20B4
? PPF01 #B20B7 ? PFPB1 #B20B5 RES #F0C6 ? RESET #F0FC
RSBUFF #B19 RSFLAG #B18 RBIN #F0F9 ? RSOUT #F0F5

```

```

? RTIA #B2107 ? RTIB #B2106 ? RT2A #B2109 ? RT2B #B2108
? RUN #B12 RUNI #F0DE SECC #B21 ? SECS10 #B24
? SECSP #B27 ? SI #F0F0 ? SRU #B210A STATUS #B3
STIP #B04 ? T1024C1 #B209F ? T1024D1 #B209F ? TIC1 #B209E
? T101 #B2094 ? T64C1 #B209E ? T64B1 #B2096 ? TOC1 #B209D
? T0B1 #B2095 TABLA #F0CB6 UP #F0DCA USIRO #B14
USREL #B25 ? VREG #B01 VRES #B02

```

Symbol table - numerical order:

```

XREG #B01 YREG #B02 STATI0 #B03 STKP #B04
ACC #B05 DIRL #B07 DIRH #B08 PASBP #B0A
HINHFP #B0B OUTFP #B0C ? PAS #B10 PFI05 #B11
AUX #B12 ? RUN #B12 ALU1 #B13 ? DIR #B13
USIRO #B14 CENFLAG #B15 CENBUFF #B16
RSFLAG #B10 RSBUFF #B19 FINC #B18 IRUSR #B1D
HRS #B1F HR510 #B20 MIN #B21 MIN1 #B22
SECS #B23 SECS10 #B24 HRSP #B25 MINP #B26
SECSP #B27 USREL #B28 INTOUT #B2A PADR #B20B0
PAGOR #B20B1 PBDR #B20B2 ? PFNCI #B20B6 ? PFI01 #B20B4
? PFPB1 #B20B5 ? PFNCI #B20B6 ? PFI01 #B20B4
? T0B1 #B2095 ? T64B1 #B209E ? T1024D1 #B209F ? TIC1 #B209E
? T0C1 #B2090 ? T64C1 #B209E ? T1024C1 #B209F PBV #B2100
PAW #B2101 ? DRBV #B2102 DRWV #B2103 CTIBU #B2104
CTIAU #B2105 ? RTIB #B2106 ? RTIA #B2107 ? RT2B #B2108
? RT2A #B2109 ? SRU #B210A ACRV #B210B PCRV #B210C
IFRV #B210D IERV #B210E ? PASV #B210F TABLA #B210A
CODE #F0C0 RES #F0C6 ENTRADA #F0CB6 CNT1 #F0C5
RUNI #F0DE MAS1 #F0CF MENOS2 #F0DA DATO #F0D0 DATI #F0D1
DATFIN #F0D0 ALU1 #F0D0 DIR1 #F0D3A DIREC #F047
DIRDAT #F062 DDAT1 #F0D6 DDAT2 #F0D2 LCDAT #F0D9A
LEI #F09C BK #F0DC ESCRIBE #F0B0 ESC1 #F0B8
UP #F0FC LEEB #F0D7 LEE1 #F0E1 SI #F0F1
LEE2 #F0FA LEE3 #F0EA LEFIN #F0E2 LOAT #F0E5
DIRADA #F0E2 ? INCENT #F0E3 ? OUTCENT #F0E4 ?
? INRS #F078 ? OUTFP #F0C ? PAS2 #F0C ? MENOS1 #F0CF
? CENOUT #F0E0 CENIN #F0E0 IRO1 #F0F3 ? RSOUT #F0F5
RSIN #F0F9 IRO2 #F02C ? INREL #F0F2 ? LEHORA #F0F3
IREL #F0FA OTRDIA #F0E2 FINTR #F0FE F1 #F0F3
? MHI #F0FFA ? RESET #F0FC ? IRQ #F0FE

```

APENDICE E LISTADO HEXADECIMAL DEL PROGRAMA MONITOR

Aquí presentamos el listado tal como debe grabarse en la EPROM. Como se observará, no está listado en las mismas direcciones que en el listado fuente; esto se debe a que en el ordenador que utilizamos para desarrollar el programa dichas posiciones están ocupadas por la ROM de dicho ordenador; por ello, se colocó a partir de la dirección \$2000. Para grabar la EPROM este detalle es irrelevante.

Desde la dirección \$2000 hasta la dirección \$2477 no hay programa. El usuario puede colocar aquí sus rutinas antes de grabar la EPROM.

```
2000- FF FF 00 00 FF FF 00 00
2008- FF FF 00 00 FF FF 00 00
2010- FF FF 00 00 FF FF 00 00
2018- FF FF 00 00 FF FF 00 00
2020- FF FF 00 00 FF FF 00 00
2028- FF FF 00 00 FF FF 00 00
2030- FF FF 00 00 FF FF 00 00
2038- FF FF 00 00 FF FF 00 00
2040- FF FF 00 00 FF FF 00 00
2048- FF FF 00 00 FF FF 00 00
2050- FF FF 00 00 FF FF 00 00
2058- FF FF 00 00 FF FF 00 00
2060- FF FF 00 00 FF FF 00 00
2068- FF FF 00 00 FF FF 00 00
2070- FF FF 00 00 FF FF 00 00
2078- FF FF 00 00 FF FF 00 00
2080- FF FF 00 00 FF FF 00 00
2088- FF FF 00 00 FF FF 00 00
2090- FF FF 00 00 FF FF 00 00
2098- FF FF 00 00 FF FF 00 00
20A0- FF FF 00 00 FF FF 00 00
20A8- FF FF 00 00 FF FF 00 00
```


2380- FF FF 00 00 FF FF 00 00
2388- FF FF 00 00 FF FF 7F 12
23C0- FF FF 00 00 FF FF 00 00
23C8- FF FF 00 00 FF FF 00 00
23D0- FF FF 00 00 FF FF 00 00
23D8- FF FF 00 00 FF FF 00 00
23E0- FF FF 00 00 FF FF 00 00
23E8- FF FF 00 00 FF FF 00 00
23F0- FF FF 00 00 FF FF 00 00
23FC- FF FF 00 00 FF FF 57 06
2400- FF FF 00 00 FF FF 00 00
2400- FF FF 00 00 FF FF 00 00
2410- FF FF 00 00 FF FF 00 00
2418- FF FF 00 00 FF FF 00 00
2420- FF FF 00 00 FF FF 00 00
2420- FF FF 00 00 FF FF 00 00
242C- FF FF 00 00 FF FF 00 00
2430- FF FF 00 00 FF FF 00 00
2438- FF FF 00 00 FF FF 00 00
2440- FF FF 00 00 FF FF 00 00
2440- FF FF 00 00 FF FF 00 00
2450- FF FF 00 00 FF FF 00 00
2450- FF FF 00 00 FF FF 00 00
2460- FF FF 00 00 FF FF 00 00
2468- FF FF 00 00 FF FF 00 00
2470- FF FF 00 00 FF FF 00 00
2478- FC E3 FC F1 FC D0 FD FC
2480- 2C FF 40 79 24 30 FC F1
2488- FC FF FC DE FD FD 3A 0D
2490- 40 79 24 30 19 12 02 78
2498- 00 10 08 03 46 24 06 0E
24A0- 08 09 0D 0C 12 7F 08 85
24A8- 05 86 01 84 02 68 85 03
24B0- 68 85 07 68 85 08 A9 0F
24B8- 8D 83 20 D8 78 20 15 FE
24C0- 20 D7 FD C9 10 30 F4 C9
24C8- 14 10 F2 38 E9 10 0A AA
24D0- 8D 86 FC 85 18 E8 BD 86
24D8- FC 85 1C 6C 1B 00 A6 04
24E0- BA A5 08 48 A5 07 48 A5
24E8- 03 48 A6 01 A4 02 A5 05
24F0- 40 18 A5 07 69 01 85 07
24FB- 90 02 E6 08 4C BD FC 38
2500- A5 07 E9 01 85 07 90 02
2508- C6 08 4C BD FC A0 13 A2
2510- 04 86 08 A9 06 05 0A 20
2518- 9A FD A2 04 20 30 FD C0
2520- 13 F0 06 99 01 00 18 90

2528- 04 A2 00 81 07 4C BD FC
2530- 85 0C 0A 0A 0A 0A E8 15
2538- 0C 60 20 D7 FD C9 10 30
2540- 06 C5 14 00 F5 F0 1B A2
2548- 00 86 0B A9 04 85 0A 20
2550- 9A FD A9 00 20 12 00 85
2558- 08 E8 20 12 00 85 07 4C
2560- BD FC 20 D7 FD C9 0A 00
2568- 03 A9 04 A8 A9 7F 85 12
2570- 85 0C 85 0D 85 0E B9 90
2578- FC 85 0F A2 04 20 2C FE
2580- 20 B7 FD 20 D7 FD C5 14
2588- F0 08 48 20 15 FE 68 4C
2590- C3 FC A4 12 20 0F FD 4C
2598- BD FC A6 08 20 D7 FD C5
25A0- 11 F0 09 95 0C 20 B7 FD
25A8- E0 E4 0A 60 A9 7F 95 0C
25B0- CA E4 0B F0 E5 D0 E5 A9
25B8- 7F 8D 81 20 A2 05 84 0C
25C0- B9 90 FC 8D 80 20 A0 7F
25C8- 88 10 FB 8C 80 20 A9 06
25D0- 8D 82 20 CA 10 E8 60 A9
25D8- 00 8D 81 20 A2 06 8E 82
25E0- 20 AD 80 20 00 08 E8 E0
25E8- 09 D0 F6 20 B7 FD 4C D7
25F0- FD 86 13 48 20 B7 FD 68
25F8- A6 13 A0 7F 88 10 FB AD
2600- 80 20 85 12 8A 38 E9 06
2608- F0 08 10 A9 06 65 12 CA
2610- D0 F8 A5 12 60 A5 08 A2
2618- 00 20 2C FE A5 07 20 2C
2620- FE A0 00 B1 07 20 2C FE
2628- 20 B7 FD 60 48 29 F0 0A
2630- 0A 0A 0A 95 0C E8 68 29
2638- 0F 95 0C E8 60 A9 00 8D
2640- 03 21 85 15 4C 50 FE A9
2648- FF 8D 03 21 A9 01 85 15
2650- A9 00 8D 01 21 AD 0C 21
2658- 29 F0 09 08 8D 0C 21 AD
2660- 08 21 29 FE 09 01 8D 08
2668- 21 A9 83 0D 0E 21 8D 0E
2670- 21 A9 00 8D 0D 21 58 60
2678- AD 08 21 29 E1 09 0E 8D
2680- 08 21 AD 0C 21 29 0F 09
2688- 80 8D 0C 21 A9 00 85 18
2690- 4C AB FE AD 08 21 29 E1
2698- 09 1E 8D 08 21 AD 0C 21
26A0- 29 0F 09 00 8D 0C 21 A9

APENDICE F CODIGOS PROGRAMACION

24A8- 01 85 18 A9 00 8D 0D 21
 24B0- AD 0E 21 09 84 8D 0E 21
 24B8- 58 40 AD 0D 21 29 80 F0
 24C0- 32 AD 0D 21 27 02 F0 2B
 24C8- A5 15 F0 14 A5 16 F0 23
 24D0- C6 16 8A 40 A2 00 A1 16
 24D8- 8D 01 21 68 8A 4C F3 FC
 24E0- A5 16 C9 FF F0 0D E6 16
 24E8- 8A 48 A2 00 AD 01 21 91
 24F0- 16 68 AA AD 0D 21 29 80
 24F8- F0 32 AD 0D 21 29 1C F0
 2700- 2B A5 18 F0 14 A5 19 F0
 2708- 23 C6 19 8A 48 A2 00 A1
 2710- 19 8D 00 21 68 AA 4C 2C
 2718- FF A5 19 C9 FF F0 0D E6
 2720- 19 8A 40 A2 00 AD 00 21
 2728- 81 19 68 AA 6C 1D 00 A9
 2730- FF 85 1E AD 8A FF 85 1D
 2738- AD 0E 21 09 C0 8D 0E 21
 2740- AD 0B 21 09 40 8D 0B 21
 2748- 58 A9 22 8D 04 21 A9 F4
 2750- 8D 05 21 A2 00 86 0B A9
 2758- 06 85 0A 20 9A FD 20 13
 2760- 00 85 1F 20 13 00 85 21
 2768- 20 13 00 85 23 A2 00 86
 2770- 0B A9 06 85 0A 20 9A FD
 2778- 20 13 00 85 25 20 13 00
 2780- 85 26 20 13 00 05 27 4C
 2788- A6 FC 48 8A 48 C6 2A D0
 2790- 4D A9 10 05 2A A9 30 E6
 2798- 23 A2 3A E4 2D D0 3F 85
 27A0- 23 E6 24 A2 36 E4 24 D0
 27A8- 35 85 24 E6 21 A2 3A E4
 27B0- 21 D0 28 85 21 E6 22 A2
 27B8- 36 E4 22 D0 21 85 22 E6
 27C0- 1F A2 3A E4 1F D0 07 85
 27C8- 1F E6 20 4C DE FF A2 34
 27D0- E4 1F D0 0A A2 32 E4 20
 27D8- D0 04 85 20 85 1F A5 23
 27E0- C5 27 D0 0F A5 21 C5 26
 27E8- D0 09 A5 1F C5 25 D0 03
 27F0- 4C 28 D0 AD 04 21 60 A4
 27F8- 68 40 A6 FC A6 FC BA FE

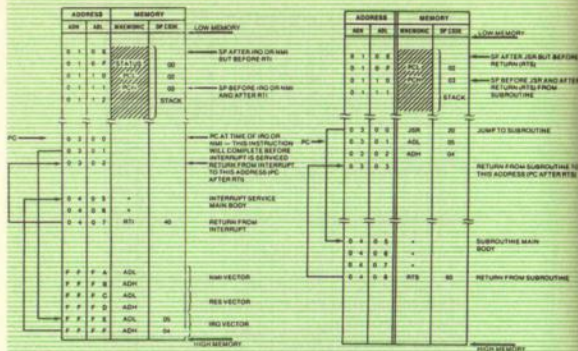
OPERATION CODE TABLE

Op	R	A	X	E	C	D	F	Y
0	000	000-000 X					000-0 PAGE	000-0 PAGE
1	001	000-000 X					000-1 PAGE X	000-1 PAGE X
2	002	000-000 X				001-2 Page	000-2 PAGE	000-2 PAGE
3	003	000-000 X					000-3 PAGE X	000-3 PAGE X
4	004	000-000 X					000-4 PAGE	000-4 PAGE
5	005	000-000 X					000-5 PAGE X	000-5 PAGE X
6	006	000-000 X					000-6 PAGE	000-6 PAGE
7	007	000-000 X					000-7 PAGE X	000-7 PAGE X
8	008	000-000 X					000-8 PAGE X	000-8 PAGE X
9	009	000-000 X				001-9 Page	000-9 PAGE	000-9 PAGE
A	010	000-000 X					000-A PAGE	000-A PAGE
B	011	000-000 X	100-000				100-0 PAGE	100-0 PAGE
C	012	100-000 Y				101-2 Page	100-2 PAGE	100-2 PAGE
D	013	100-000 Y					100-3 PAGE	100-3 PAGE
E	014	100-000 Y				101-4 Page	100-4 PAGE	100-4 PAGE
F	015	100-000 Y					100-5 PAGE	100-5 PAGE
10	016	200-000 Y				201-2 Page	200-2 PAGE	200-2 PAGE
11	017	200-000 Y					200-3 PAGE	200-3 PAGE
12	018	200-000 Y				201-4 Page	200-4 PAGE	200-4 PAGE
13	019	200-000 Y					200-5 PAGE	200-5 PAGE
14	020	300-000 X				301-2 Page	300-2 PAGE	300-2 PAGE
15	021	300-000 X					300-3 PAGE	300-3 PAGE

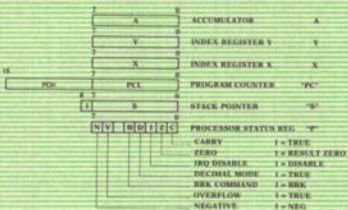
Op	R	A	X	E	C	D	F	Y
000	000	000-000	000-0			000-000	000-000	000-000
001	000	000-000 X	000-1			000-000 X	000-000 X	000-000 X
002	000	000-000	000-2			000-000	000-000	000-000
003	000	000-000 X	000-3			000-000 X	000-000 X	000-000 X
004	000	000-000	000-4			000-000	000-000	000-000
005	000	000-000 X	000-5			000-000 X	000-000 X	000-000 X
006	000	000-000	000-6			000-000	000-000	000-000
007	000	000-000 X	000-7			000-000 X	000-000 X	000-000 X
008	000	000-000	000-8			000-000	000-000	000-000
009	000	000-000 X	000-9			000-000 X	000-000 X	000-000 X
010	100	100-000	100-0			100-000	100-000	100-000
011	100	100-000 X	100-1			100-000 X	100-000 X	100-000 X
012	100	100-000	100-2			100-000	100-000	100-000
013	100	100-000 X	100-3			100-000 X	100-000 X	100-000 X
014	100	100-000	100-4			100-000	100-000	100-000
015	100	100-000 X	100-5			100-000 X	100-000 X	100-000 X
016	200	200-000	200-0			200-000	200-000	200-000
017	200	200-000 X	200-1			200-000 X	200-000 X	200-000 X
018	200	200-000	200-2			200-000	200-000	200-000
019	200	200-000 X	200-3			200-000 X	200-000 X	200-000 X
020	300	300-000	300-0			300-000	300-000	300-000
021	300	300-000 X	300-1			300-000 X	300-000 X	300-000 X

FIG. 1 IRO, NMI, RTI, BRK OPERATION

FIG. 2 JSR, RTS OPERATION



PROCESSOR PROGRAMMING MODEL



BACKWARD RELATIVE BRANCH TABLE

MSB	150	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	
A	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	
B	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	
C	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	
D	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	
E	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	
F	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

FORWARD RELATIVE BRANCH TABLE

MSB	150	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	

COMPARE INSTRUCTION RESULTS

Condition	Z	C
A.X. or Y<C Memory	1*	0
A.X. or Y<C Memory	0	1
A.X. or Y>C Memory	0	0
A.X. or Y>C Memory	0	1

*N is valid only for 2's complement compare.

HEXADECIMAL AND DECIMAL CONVERSION

HEXADECIMAL COLUMNS											
8		9		A		B		C		D	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	85,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	1,710,720	2	8,192	2	512	2	32	2	2
3	3,145,728	3	2,566,800	3	12,288	3	768	3	48	3	3
4	4,194,304	4	3,422,880	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	4,278,960	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	5,135,040	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	6,000,120	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	6,865,200	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	7,730,280	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	8,595,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	9,460,440	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	10,325,520	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	11,190,600	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	12,055,680	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	12,920,760	F	61,440	F	3,840	F	240	F	15
	7654		3210		7654		3210		7654		3210
	Byte		Byte		Byte		Byte		Byte		Byte

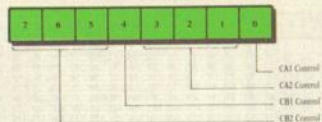
POWERS OF 2

2 ⁿ	n	2 ⁿ × 10 ³
256	8	2 ⁸ × 10 ³
512	9	2 ⁹ × 10 ³
1,024	10	2 ¹⁰ × 10 ³
2,048	11	2 ¹¹ × 10 ³
4,096	12	2 ¹² × 10 ³
8,192	13	2 ¹³ × 10 ³
16,384	14	2 ¹⁴ × 10 ³
32,768	15	2 ¹⁵ × 10 ³
65,536	16	2 ¹⁶ × 10 ³
131,072	17	2 ¹⁷ × 10 ³
262,144	18	2 ¹⁸ × 10 ³
524,288	19	2 ¹⁹ × 10 ³
1,048,576	20	2 ²⁰ × 10 ³
2,097,152	21	2 ²¹ × 10 ³
4,194,304	22	2 ²² × 10 ³
8,388,608	23	2 ²³ × 10 ³
16,777,216	24	2 ²⁴ × 10 ³

POWERS OF 16

16 ⁿ	n
1	0
16	1
256	2
4,096	3
65,536	4
1,048,576	5
16,777,216	6
268,435,456	7
4,294,967,296	8
68,719,476,736	9
1,099,511,627,776	10
17,592,188,044,416	11
281,474,078,710,656	12
4,503,599,627,710,496	13
72,057,594,037,927,936	14
1,152,921,504,606,846,976	15

R6522 PERIPHERAL CONTROL REGISTER (PCR)

**CA1 CONTROL**

PCR0 = 0 The CA1 Interrupt Flag (IFR1) will be set by a negative transition (high to low) on the CA1 pin.
 = 1 The CA1 Interrupt Flag (IFR1) will be set by a positive transition (low to high) on the CA1 pin.

CA2 CONTROL

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 negative edge interrupt (IFR0/ORA clear mode - Set CA2 interrupt flag (IFR0) on a negative transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	0	1	CA2 negative edge interrupt (IFR0 clear) mode - Set IFR0 on a negative transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
0	1	0	CA2 positive edge interrupt (IFR0/ORA clear) mode - Set CA2 interrupt flag (IFR0) on a positive transition of the CA2 input signal. Clear IFR0 on a read or write of the ORA or by writing logic 1 into IFR0.
0	1	1	CA2 positive edge interrupt (IFR0 clear) mode - Set IFR0 on a positive transition of the CA2 input signal. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 handshake output mode - Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 pulse output mode - CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 low output mode - The CA2 output is held low in this mode.
1	1	1	CA2 high output mode - The CA2 output is held high in this mode.

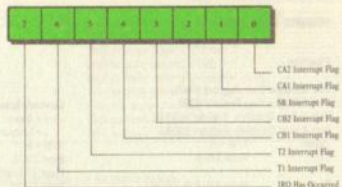
CB1 CONTROL

PCR4 = 0 The CB1 Interrupt Flag (IFR4) will be set by a negative transition (high to low) on the CB1 pin.
 = 1 The CB1 Interrupt Flag (IFR4) will be set by a positive transition (low to high) on the CB1 pin.

CB2 CONTROL

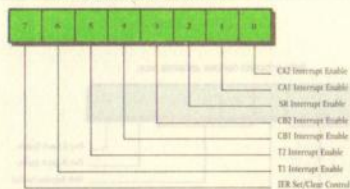
PCR7	PCR6	PCR5	Mode
0	0	0	CB2 negative edge interrupt (IFR3/ORB clear) mode - Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	0	1	CB2 negative edge interrupt (IFR3 clear) mode - Set IFR3 on a negative transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 positive edge interrupt (IFR3/ORB clear) mode - Set CB2 interrupt flag (IFR3) on a positive transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	1	1	CB2 positive edge interrupt (IFR3 clear) mode - Set IFR3 on a positive transition of the CB2 input signal. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 handshake output mode - Set CB2 output low on a write of the Peripheral B Output Register. Reset CB2 high with an active transition on CB1.
1	0	1	CB2 pulse output mode - CB2 goes low or one cycle following a read or write of the Peripheral B Output Register.
1	1	0	CB2 low output mode - The CB2 output is held low in this mode.
1	1	1	CB2 high output mode - The CB2 output is held high in this mode.

R6522 INTERRUPT FLAG REGISTER (IFR)



IFR Bit	Set By	Cleared By
0	Active transition on CA2	Reading or writing the ORA (\$A001 or \$A00F)
1	Active transition on CA1	Reading or writing the ORA (\$A001 or \$A00F)
2	Completion of eight shifts	Reading or writing the SR (\$A00A)
3	Active transition on CB2	Reading or writing the ORB (\$A000)
4	Active transition on CB1	Reading or writing the ORB (\$A000)
5	Time-out of Timer 2	Reading TIC-L (\$A008) or writing TIC-H (\$A009)
6	Time-out of Timer 1	Reading TIC-L (\$A004) or writing TIC-H (\$A005 or \$A007)
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 (\$A000) or IER0-IER6 (\$A00E)

R6522 INTERRUPT ENABLE REGISTER (IER)

**INTERRUPT ENABLE BITS (IER0-6)**

IERn = 0 Disable interrupt
 = 1 Enable interrupt

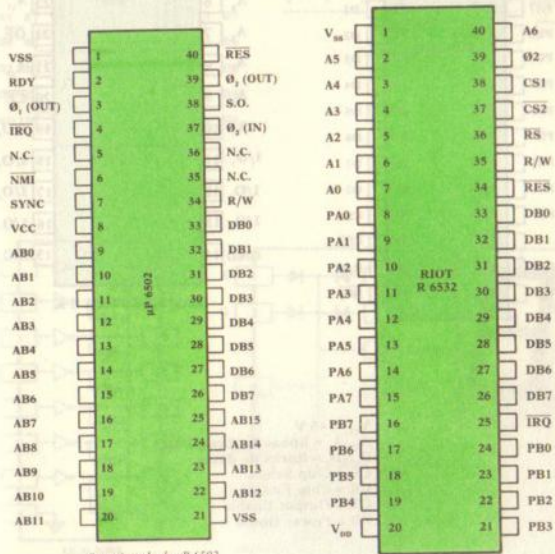
IER SET/CLEAR CONTROL (IER7)

IER7 = 0 For each data bus bit set to logic 1, clear corresponding IER bit.
 = 1 For each data bus bit set to logic 1, set corresponding IER bit.

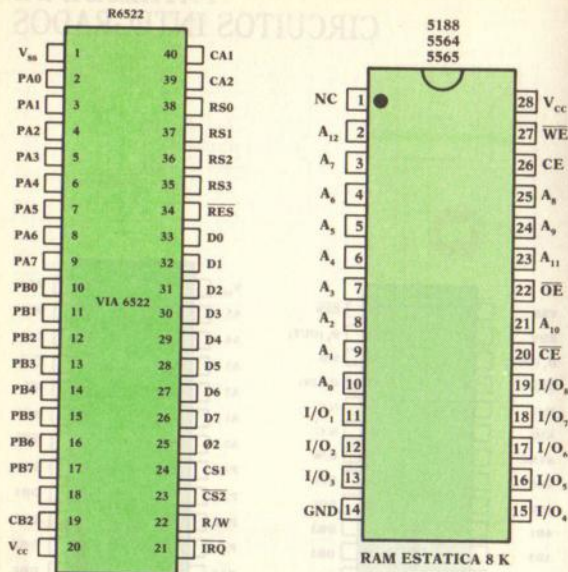
Note: IER7 is active only when R/W = L; when R/W = H, IER7 will read logic 1.

Instrucciones de la VÍA R6522.

APENDICE G PATILLAJE DE CIRCUITOS INTEGRADOS

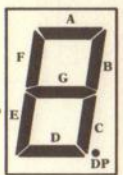
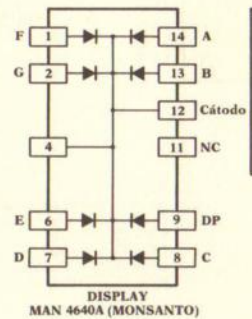
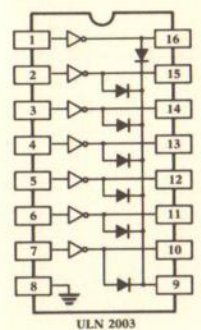
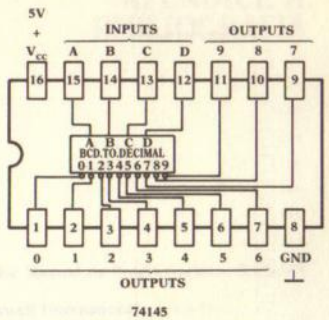
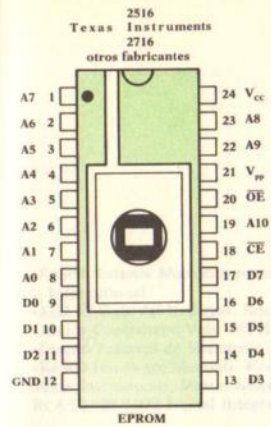


Conexión de µP 6502.



RAM ESTÁTICA 8 K

V_{cc} = +5 V
 A₀...A₇ = líneas de direcciones
 D₀...D₇ = líneas de datos
 CS = Chip Select
 CE = Chip Enable
 OE = Output Enable
 PD = Power Down



APENDICE H. BIBLIOGRAFIA

- «R6500 Sistema Microcomputador Manual de Programación». Rockwell International.
- «AIM 65 Guía del Usuario». Rockwell International.
- «Junior Computer». Vols. I-II-III-IV. Ingelek, S.A.
- «Diseño Práctico de Sistemas». Angulo.
- «R6500 Hardware Manual». Rockwell International.
- Texas Instruments, Master Selection Guide 1984/85.
- RCA CMOS/MOS Digital Integrated Circuits. 1974.