

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

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» utilizando, 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.



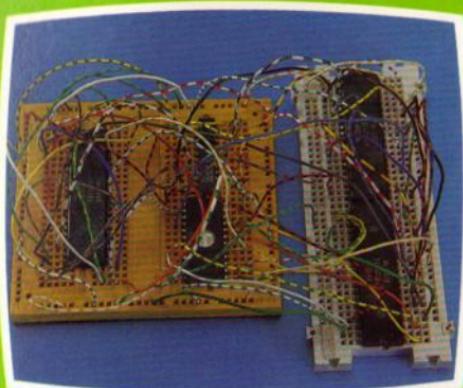
Cómo construir su propio ordenador



15

Cómo construir su propio ordenador

AIA



SINCLAIR
AMSTRAD
CHAMODORF
IBM

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARÍA LUISA SUÁREZ PÉREZ.

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:
JOSÉ OCHOA Y ANTONIO PERERA.

Tomo XV. Cómo construir su propio ordenador

AULA DE INFORMATICA APLICADA
JOAQUÍN SALVACHUÁ, 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. Albarracín, 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.

ÍNDICE

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 hexadecimal 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, microordenador.

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 cassette, 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.

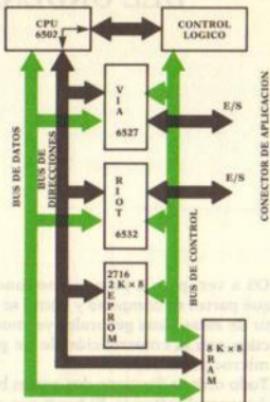


Fig. 1.1. Diagrama de bloques del ordenador.

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 o un 1 o un 0. 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 aplicó en los años cuarenta a la construcción de los primeros ordenadores, y desde entonces no se han realizado realmente diferentes. Actual-

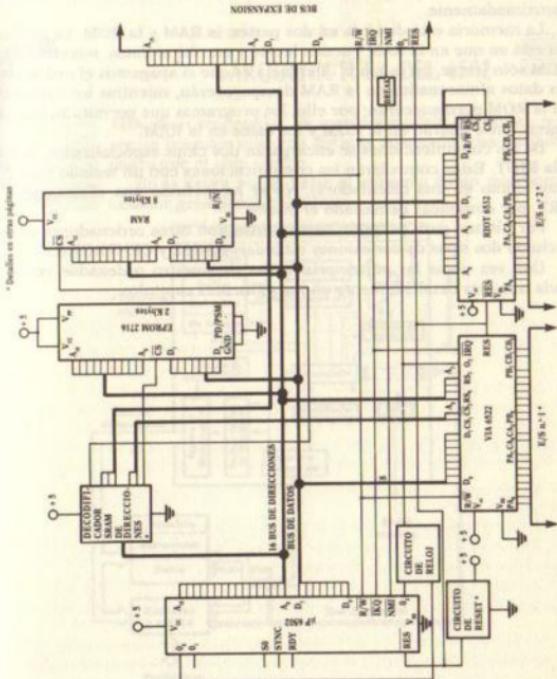


Fig. 1.2. Esquema general del ordenador.

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.

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

ARQUITECTURA DEL 6502

Este 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.

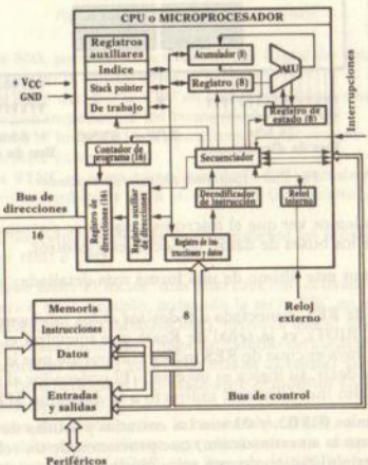


Fig. 2.1. Esquema del ordenador.

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).

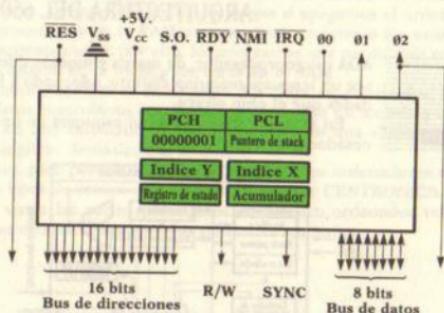


Fig. 2.2. Esquema del microprocesador.

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 \$FFFC \$FFFF.)

- 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.

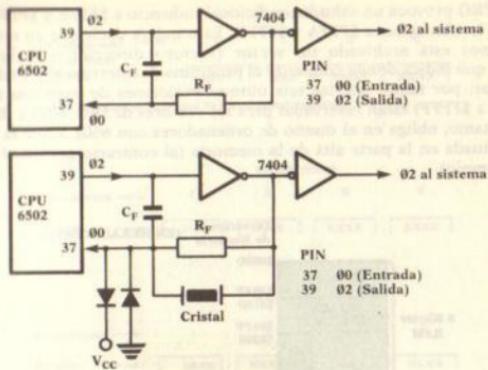


Fig. 2.3. Esquemas posibles de reloj.

- La linea S.O. permite la activación del flag V de sobrepasamiento desde el exterior (no la utilizaremos).
- La linea RDY detiene la CPU en todos los ciclos, excepto en escritura, y deja en alta impedancia los buses (no la usamos).
- La linea R/W indica a los dispositivos si el acceso es de lectura (nivel alto) o escritura (nivel bajo).
- La linea 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, estudiamos 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).

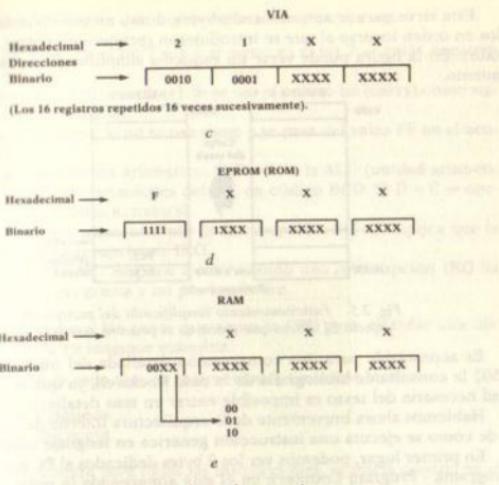
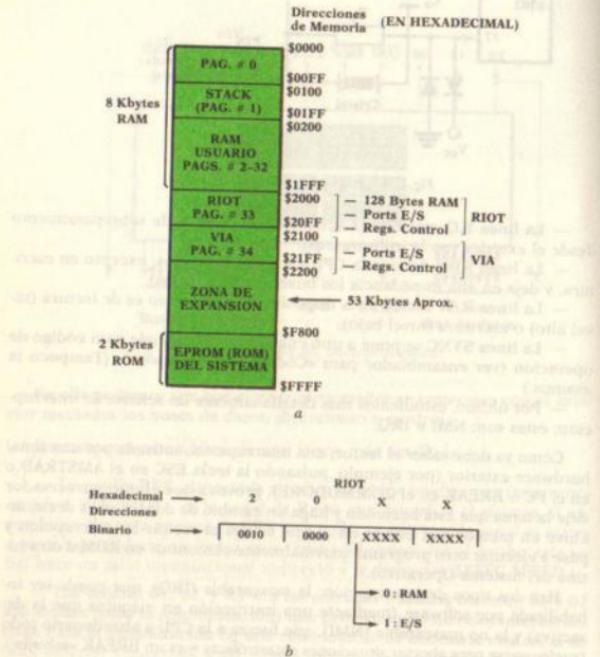


Fig. 2.4. Mapa de memoria.

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 \$XXOO 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.

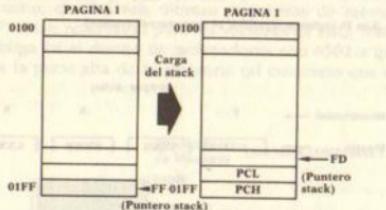


Fig. 2.5. Funcionamiento simplificado del puntero de stack. El último que entra, es el primero que sale.

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 y 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.

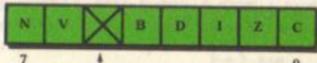


Fig. 2.6. Estructura de los bits del registro de Estado, que actúan como señalizadores o alarmas (flags).

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 aritmética 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 R), 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}

← 8 bits de menos peso de la dirección de M

← 8 bits de más peso de la dirección de M

MEMORIA DATOS

M _n - M _{n+1}	2.º operando



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.

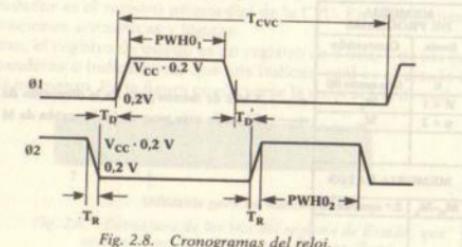


Fig. 2.8. Cronogramas del reloj.

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, D1, ..., D7 (la D viene de Data, es decir, dato).

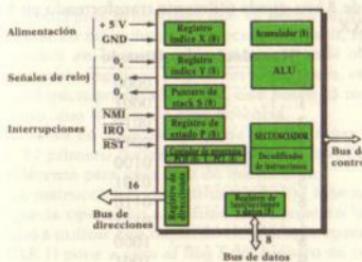


Fig. 2.9. Esquema Riot.

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 incomprendible por el hardware del ordenador (circuitería), 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 (nemotécnicos o nemómicos) 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 Acumulator) significa «Carga en el acumulador el dato...». LDA es traducido por el ensamblador al número binario 1010.1001 (ver figura).

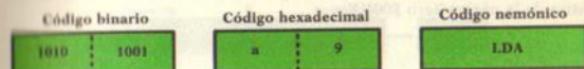


Fig. 2.10. Tres formas de expresar la instrucción de "Carga del Acumulador" mostrando la evolución del lenguaje máquina.

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 dotan 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 0 0 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) \leftarrow (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) \leftarrow (00,61)

4) Direcciónamiento 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) \leftarrow (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) \leftarrow (22,33+X)

8) Direcciónamiento 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 \leftarrow 0

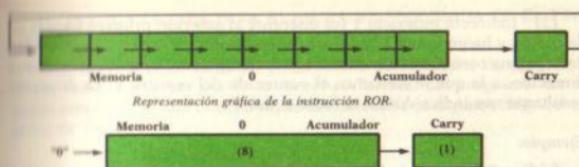


Fig. 2.11. Representación gráfica de la instrucción LSR.

10) Relativo (2 bytes): El segundo byte de la instrucción es un valor 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) \leftarrow (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) \leftarrow (3128) Y (3129)

12) Indexado X indirecto (primero indexa y luego indirecciona) (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) \leftarrow ((0030)+X)

NOTA: Cuando se escribe (\leftarrow) 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 ARITMÉTICAS Y LÓGICAS

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

Transfieren 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 SUBRUTINA

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	Compará (resta A a 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

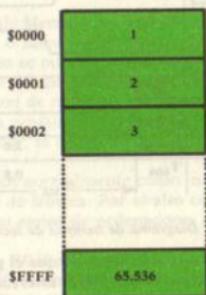
NOP	No hace nada (pierde dos ciclos de máquina)
-----	--



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.

DIRECCION-CASILLA



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 sincrónico 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 01.

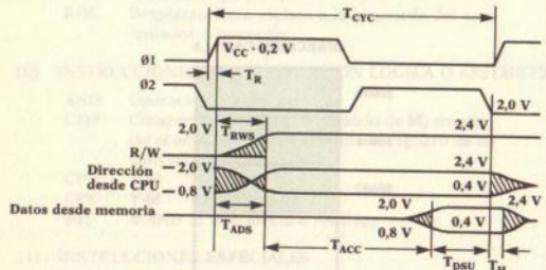


Fig. 3.2. Diagrama de tiempos de lectura.

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 02 pasa de estado bajo a estado alto, pues al final del ciclo alto de 02 el procesador tomará como dato el valor que la memoria coloque en el bus de datos.

El proceso de escritura es prácticamente idéntico.

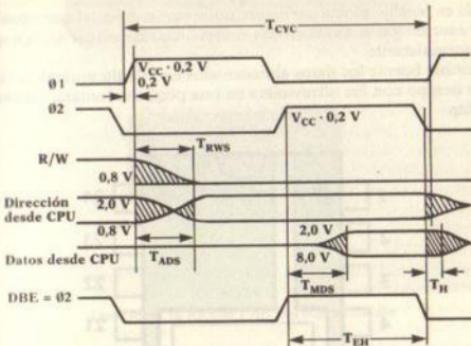


Fig. 3.3. Diagrama de tiempos de escritura.

La diferencia está en que mientras 02 está alto R/W está a nivel bajo y que durante la parte alta de 02 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 lectura).

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.

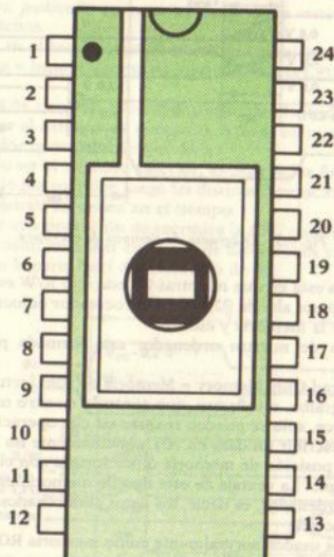
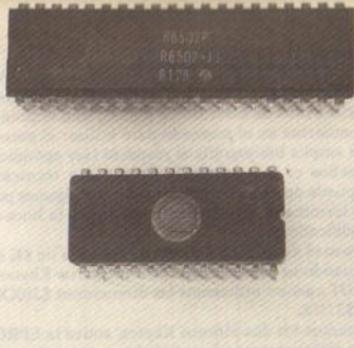


Fig. 3.4. Patillaje de una Eprom.

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 guardarlo en una cinta o casette.



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 patillas 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.

L

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 ($20nS \times n$ como máximo) comparado con el período de reloj ($1\text{ }\mu\text{s} = 10^{-6}\text{ S}$).

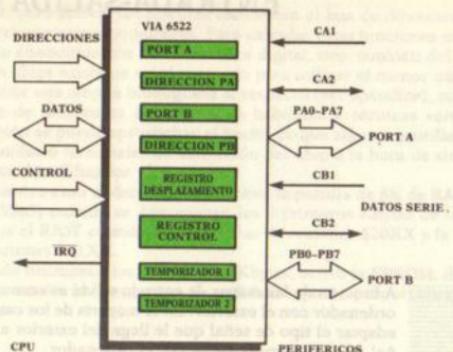


Fig. 4.1. Esquema de la vía.

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

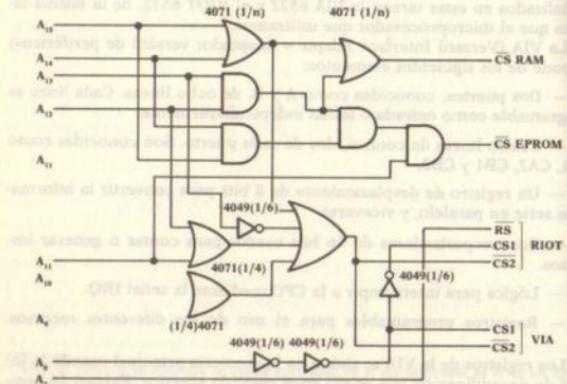
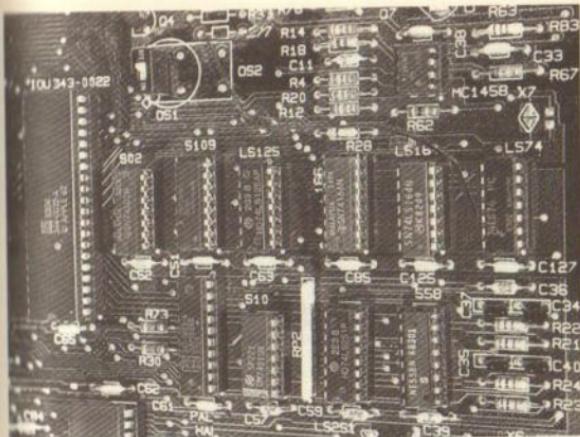


Fig. 4.2. Decodificador de direcciones.



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 pondriamos en su registro de control 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 tuviera llegando del exterior.

Veamos dos ejemplos:

```
:L  
1 LDA #11111111 ;PROGRAMA SALIDAS  
2 STA DRBV  
3 LDA PBV ; LEER DATOS
```

```
:L  
1 LDA #00000000 ;PROGRAMA SALIDAS  
2 STA DRBV  
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.

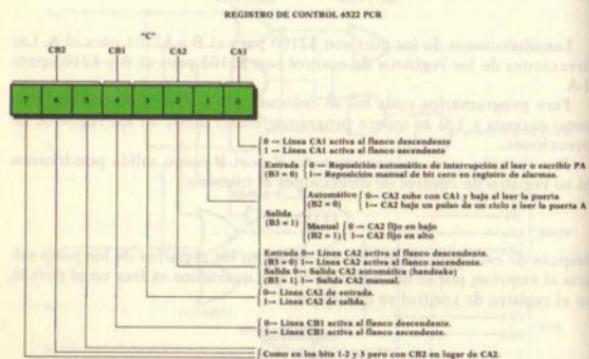


Fig. 4.3. Función de cada bit del registro de control PCR.

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 transmitir 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?

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 PAU ;DEJA EN PORT A  
2 SIGUE LDA PCRV ;COMPROBAR SI LEIDO  
3 AND #000001000  
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.

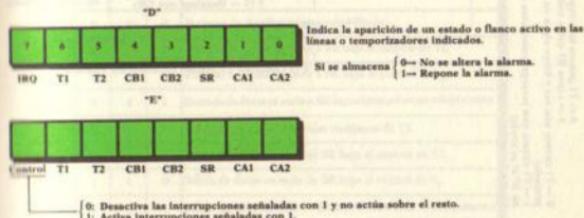


Fig. 4.4. Registro de alarmas de interrupciones IFR.

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.

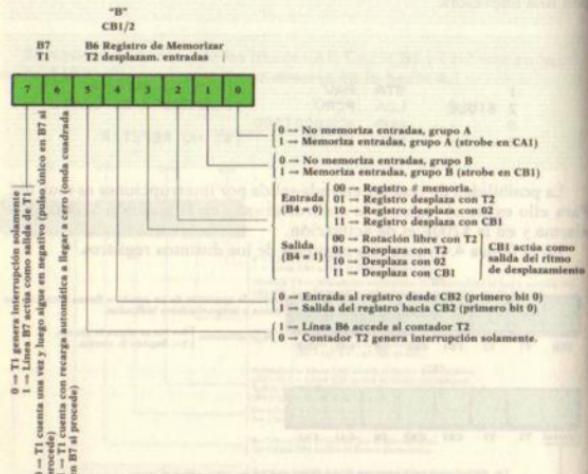


Fig. 4.5. Registro auxiliar de control ACR.

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 es 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.

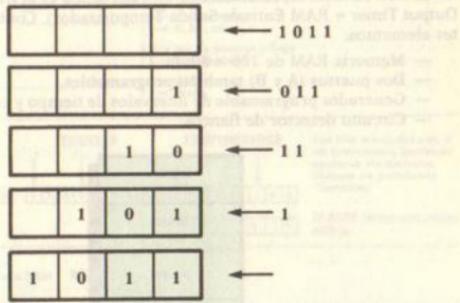


Fig. 4.6.

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 0 _r .
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 0 _r .
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 determinarse por:

- La activación de la bandera T2.
- Con 02.
- 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 128×8 bits.
- Dos puertos (A y B) también programables.
- Generador programable de intervalos de tiempo y con interrupción.
- Circuito detector de flancos.

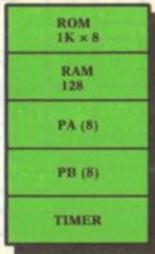


Fig. 4.8. RIOT.

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

Sus direcciones en nuestro mapa de memoria son desde \$2000 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.

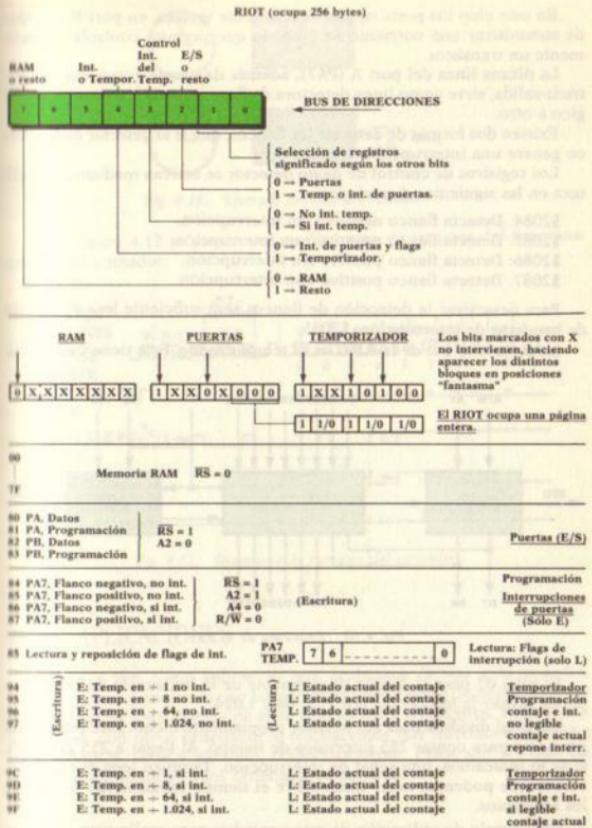


Fig. 4.9.

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.

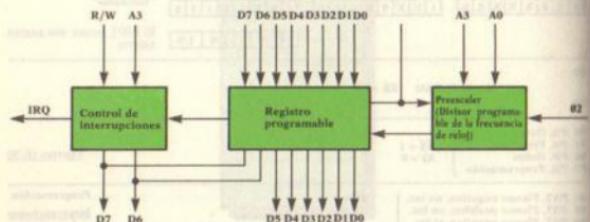


Fig. 4.10. Esquema de la Riot.

El reloj $\varnothing 2$ 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 leer 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.

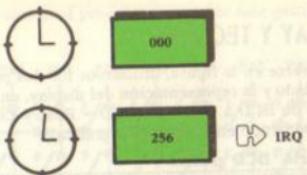


Fig. 4.11. Ejemplo del uso del contador.

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

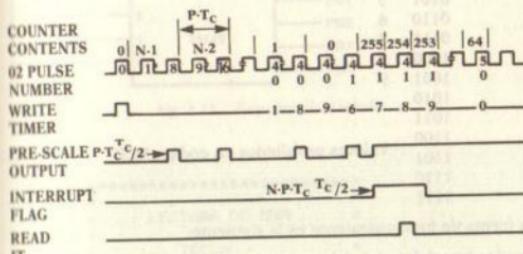


Fig. 4.12. Diagrama de tiempos del contador.

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	Valores prohibidos en código BCD
1101	
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 sobre 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.

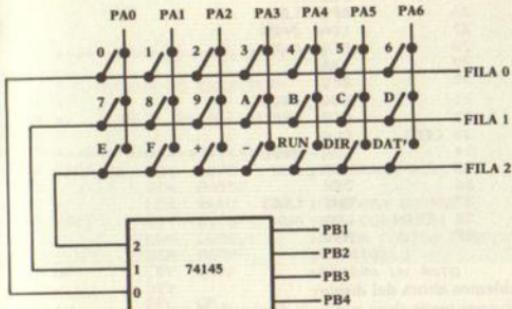


Fig. 4.13. Esquema del teclado.

```

1 ****
2 *      *
3 * LECTURA DE UNA *
4 *      *
5 * TECLA   *
6 *      *
7 ****
8 LEEKB  LDA #$00 ;PROGRAMA PORT
9 STA PADR
10 LDX #$06
11 STX PBDR
12 LEEI  LDA PADR
13 BNE SI
14 INX
15 CPX #$09
16 BNE LEEI
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 LEFIN
33 LEE3    CLC
34 LDA #\$06
35 ADC AUX
36 DEX
37 BNE LEE3
38 LEFIN   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 refresharemos 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.

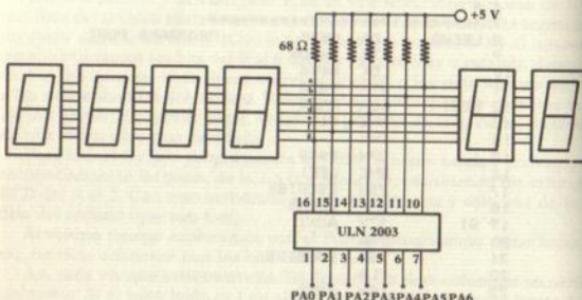


Fig. 4.14. Esquema del «Display».

Veamos el programa de control.

1.

```

1 ****
2 *
3 * RUTINA DE ESCRITURA *
4 *
5 * EN EL DISPLAY *
6 *
7 ****
8 ESCRIBE LDA #\$7F ;PROGRAMA PORT A
9 STA PADR
10 LDY #\$05 ;PREPARA BUFFER
11 ESC1 LOY OUTBF,X ;LEE CODIGO
12 LDA CODE,Y ;CARGA CODIGO DISPLAY
13 STA PADR ;ESCRIBELO
14 UP LOY #\$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 ESC1 ;FIN ?
22 RTS

```

Utilizamos la tabla para encender los segmentos de cada dígito.

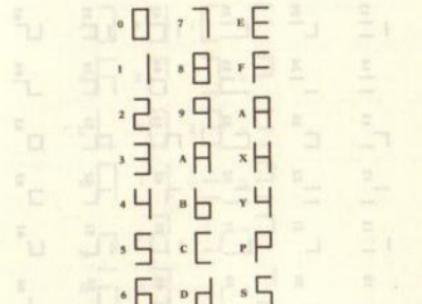


Fig. 4.15. Códigos para «Display».

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

-	-	1	02	03	04	-	05	06	07	-	08	-	09	0A	-	0B	0C	-	0D	0E	0F	
	10	-	11	1	12	1	13	1	14	-	15		16	17	L	18	L	19	L	1A	L	1B
	20	-	21	11	22	11	23	1	24	1	25		26	1	27	1	28	1	29	1	2A	
	30	-	31	1	32	1	33	1	34	1	35		36	1	37	1	38	1	39	1	3A	
	40	-	41	1	42	1	43	1	44	-	45		46	1	47	-	48	1	49	-	4A	
	50	-	51	1	52	1	53	1	54	-	55		56	1	57	1	58	1	59	1	5A	
	60	-	61	1	62	1	63	1	64	1	65	1	66	1	67	-	68	-	69	-	6A	
	70	-	F ₁	H ₁	H ₂	P ₁	h ₁	h ₂	H ₃	H ₄	H ₅	H ₆	R ₁	H ₇	E ₁	H ₈	E ₂	b ₁	b ₂	H ₉		

Fig. 4.16. Figuras posibles del «Display».

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

:	-	O	4
A	-	P	5
B	-	Q	6
C	-	R	7
D	-	S	8
E	-	T	9
F	-	U	+
G	-	V	-
H	-	W	-
I	-	X	-
J	-	Y	/
K	-	Z	-
L	0	\$	-
M	1	1	&
N	2	^	-
S	3	~	-

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.

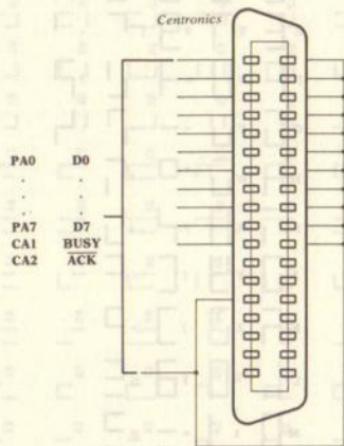
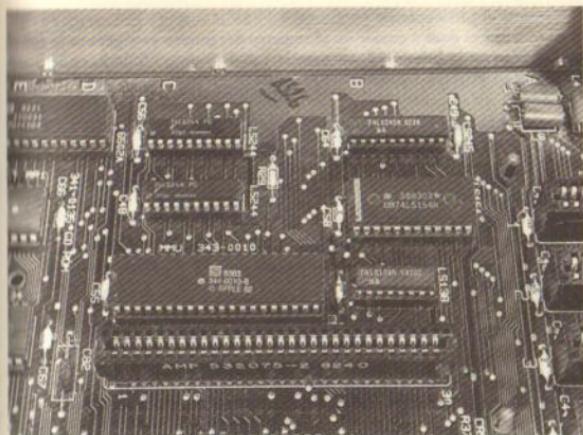


Fig. 4.18. Conexión interfaz Centronics.

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 lí-



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.

PHASE TWO CLOCK
WRITE ORA
OPERATION¹
DATA AVAILABLE-
HANDSHAKE MODE
(CA2.CB2)
DATA AVAILABLE
PULSE MODE
(CA2.CB2)
DATA TAKEN
(CA1.CB1)
IRQ OUTPUT²

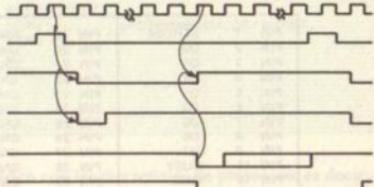


Fig. 4.19. Diagrama de tiempos del «Handshake» de escritura.

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:

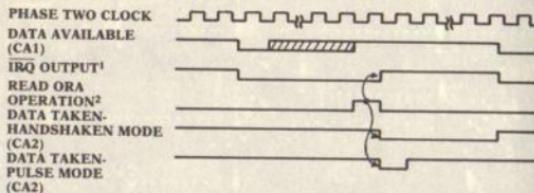


Fig. 4.20. Diagrama de tiempos del «Handshake» de lectura.

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

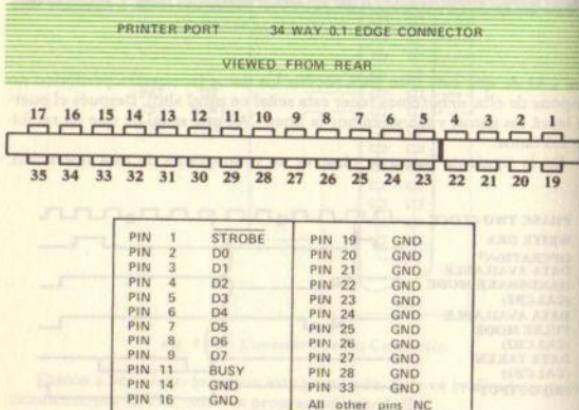


Fig. 4.21. Conector paralelo Centronics.

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 DR4V ;PROGRAMA REG. DIRECCION
18 STA CENFLAG
19 JMP CONT
20 OUTCENT LDA #FF ;RUTINA SALIDA
21 STA DR4V ;PROGRAMA RES. DIRECCION
22 LDA #B01
23 STA CENFLAG
24 CONT LDA #0 ;PARTE COMUN
25 STA PAU ;REGISTRO FLAGS INT.
26 LDA PCRU ;REGISTRO DE CONTROL
27 AND #M11111000
28 ORA #M200001000
29 STA PCRU
30 *
31 LDA ACRV ;REGISTRO AUXILIAR DE CONTROL
32 AND #M11111110
33 ORA #M00000001
34 STA ACRV
35 *
36 LDA #M10000011
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.

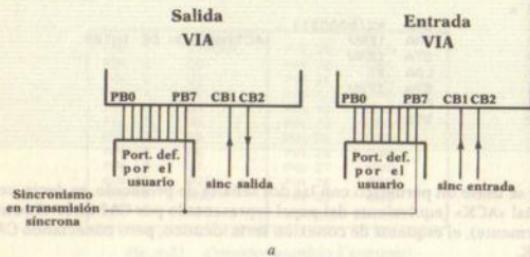
Muchos periféricos de ordenador disponen del mismo, como estándar, por lo que es obvia su gran utilidad y la conveniencia de disponer de él para conseguir una mayor versatilidad en un equipo informático. Sin embargo, y tal vez por la dificultad de obtener sus tensiones, muchos de los ordenadores personales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

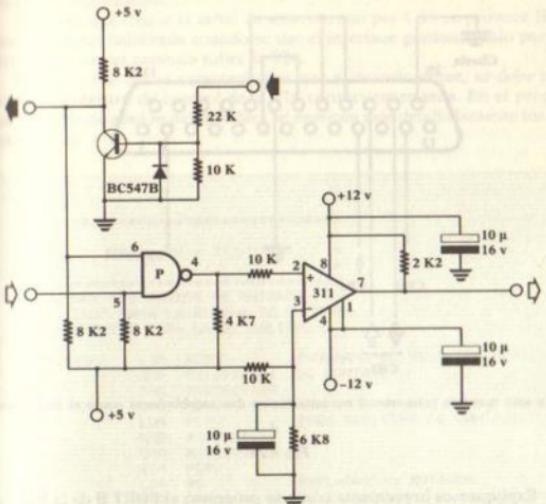
Como la transmisión es serie necesitaremos programar el Port B de la VIA adecuadamente, ya que el Port A no permite la adaptación de forma automática de paralelo (del bus de datos: ocho bits) al canal serie que necesitamos.

El RS-232 funciona con lógica negativa. Esto quiere decir que cuando queremos 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



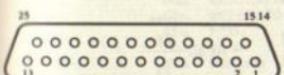
Por cada canal de I/O necesitaremos un adaptador de nivel como el de la figura:



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

b

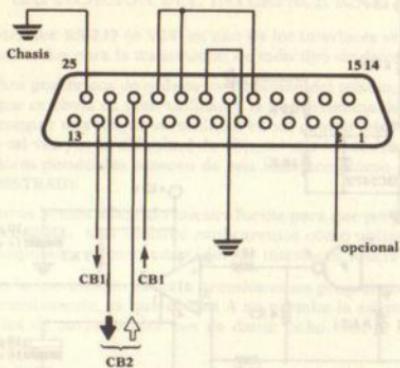
El conector RS 232/V24



- | | |
|-----------------------------|--|
| 1. Masa del chasis * | 9-14. No utilizados |
| 2. Emisión de datos ← a CB2 | 15. Bit de sincronización interno para emisión, a CB1 |
| 3. Recepción de datos → | 16. No utilizado |
| 4. Solicitud de emisión * | 17. Bit de sincronización de recepción, procede del emisor, a CB1 |
| 5. Lista para emitir * | 18-19. No utilizados |
| 6. Bloque de datos listo * | 20. Terminal de datos listo |
| 7. Masa de señal ⊥ | 21. No utilizado |
| 8. Detección de portadora | 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.

Fig. 4.22.
d

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

Como entrada

bits ACR
234

Asíncrono

011	\rightarrow ACR: 76
001	

Síncrono

11	
----	--

Como salida

bits ACR
234

Asíncrono

111	\rightarrow 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 interfaz 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.

```

1 ****
2 *          ;PROGRAMA EL REGISTRO AUXILIAR
3 *          ;DE CONTROL
4 *          ;IDEM REGISTRO DE CONTROL
5 *          ;HAY DOS PUNTOS DE ENTRADA
6 *          ;INRS PARA LA RUTINA DE ENTRADA
7 *          ;Y OUTRS PARA LA DE SALIDA
8 *
9 *
10 INRS    LDA ACRV      ;PROGRAMA EL REGISTRO AUXILIAR
11       AND #11100001  ;DE CONTROL
12       ORA #00001110
13       STA ACRV      ;IDEM REGISTRO DE CONTROL
14       LDA PCRV      ;RSFLAG=0 -> ENTRADA
15       AND #00001111
16       ORA #00000000
17       STA PCRV      ;IDEM REGISTRO DE CONTROL
18       LDA #0          ;RSFLAG=0 -> ENTRADA
19       STA RSFLAG
20       JMP CONTI
21 OUTRS   LDA ACRV      ;IDEM REGISTRO AUXILIAR DE
22       AND #11100001  ;CONTROL
23       ORA #00001110
24       STA ACRV      ;IDEM REGISTRO DE CONTROL
25       LDA PCRV      ;IDEM REGISTRO DE CONTROL
26       AND #00000111
27       ORA #00000000
28       STA PCRV      ;RSFLAG=1 -> SALIDA
29       LDA #1          ;RSFLAG=1 -> SALIDA
30       STA RSFLAG
31 CONTI   LDA #0          ;ACTIVAMOS LA INTERRUPCION
32       STA IFRU      ;CORRESPONDIENTE
33       LDA IERU
34       ORA #10000100
35       STA IERU
36       CLI
37       RTS

```

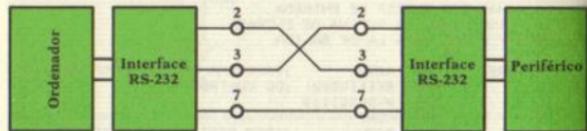
Como se observa en el programa, la velocidad de transmisión es programable por software, por lo que podrá variar según la aplicación.

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 combinacional 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, depurárselos 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 podemos 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.

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.

NECESIDADES SOFTWARE

Si realizas el que aquí aparece entenderás lo siguiente:
en cada sección de este apartado se explica un procedimiento para realizar una operación en la memoria del ordenador. Los pasos son los siguientes: 1) Se introduce la dirección de memoria en la que se encuentra la rutina que queremos ejecutar. 2) Se introduce el código de la primera instrucción. 3) Se introduce el código de la segunda instrucción. 4) Se introduce el código de la tercera instrucción. 5) Se introduce el código de la cuarta instrucción. 6) Se introduce el código de la quinta instrucción. 7) Se introduce el código de la sexta instrucción. 8) Se introduce el código de la séptima instrucción. 9) Se introduce el código de la octava instrucción. 10) Se introduce el código de la novena instrucción. 11) Se introduce el código de la décima instrucción. 12) Se introduce el código de la undécima instrucción. 13) Se introduce el código de la duodécima instrucción. 14) Se introduce el código de la treceña instrucción. 15) Se introduce el código de la cuadragésima instrucción. 16) Se introduce el código de la cuadragésima primera instrucción. 17) Se introduce el código de la cuadragésima segunda instrucción. 18) Se introduce el código de la cuadragésima tercera instrucción. 19) Se introduce el código de la cuadragésima cuarta instrucción. 20) Se introduce el código de la cuadragésima quinta instrucción. 21) Se introduce el código de la cuadragésima sexta instrucción. 22) Se introduce el código de la cuadragésima séptima instrucción. 23) Se introduce el código de la cuadragésima octava instrucción. 24) Se introduce el código de la cuadragésima novena instrucción. 25) Se introduce el código de la cuadragésima décima instrucción. 26) Se introduce el código de la cuadragésima undécima instrucción. 27) Se introduce el código de la cuadragésima duodécima instrucción. 28) Se introduce el código de la cuadragésima treceña instrucción. 29) Se introduce el código de la cuadragésima cuadragésima instrucción. 30) Se introduce el código de la cuadragésima cuadragésima primera instrucción. 31) Se introduce el código de la cuadragésima cuadragésima segunda instrucción. 32) Se introduce el código de la cuadragésima cuadragésima tercera instrucción. 33) Se introduce el código de la cuadragésima cuadragésima cuadragésima instrucción. 34) Se introduce el código de la cuadragésima cuadragésima cuadragésima primera instrucción. 35) Se introduce el código de la cuadragésima cuadragésima cuadragésima segunda instrucción. 36) Se introduce el código de la cuadragésima cuadragésima cuadragésima tercera instrucción. 37) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 38) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 39) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 40) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 41) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 42) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 43) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 44) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 45) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 46) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 47) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 48) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 49) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 50) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 51) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 52) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 53) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 54) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 55) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 56) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 57) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 58) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 59) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 60) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 61) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima instrucción. 62) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima primera instrucción. 63) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima segunda instrucción. 64) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 65) Se introduce el código de la cuadragésima instrucción. 66) Se introduce el código de la cuadragésima primera instrucción. 67) Se introduce el código de la cuadragésima segunda instrucción. 68) Se introduce el código de la cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima cuadragésima tercera instrucción. 69) Se introduce el código de la cuadragésima instrucción. 70) Se introduce el código de la cuadragésima primera instrucción. 71) Se introduce el código de la cuadragésima segunda instrucción. 72) Se introduce el código de la cuadragésima tercera instrucción. 73) Se introduce el código de la cuadragésima instrucción. 74) Se introduce el código de la cuadragésima primera instrucción. 75) Se introduce el código de la cuadragésima segunda instrucción. 76) Se introduce el código de la cuadragésima tercera instrucción. 77) Se introduce el código de la cuadragésima instrucción. 78) Se introduce el código de la cuadragésima primera instrucción. 79) Se introduce el código de la cuadragésima segunda instrucción. 80) Se introduce el código de la cuadragésima tercera instrucción. 81) Se introduce el código de la cuadragésima instrucción. 82) Se introduce el código de la cuadragésima primera instrucción. 83) Se introduce el código de la cuadragésima segunda instrucción. 84) Se introduce el código de la cuadragésima tercera instrucción. 85) Se introduce el código de la cuadragésima instrucción. 86) Se introduce el código de la cuadragésima primera instrucción. 87) Se introduce el código de la cuadragésima segunda instrucción. 88) Se introduce el código de la cuadragésima tercera instrucción. 89) Se introduce el código de la cuadragésima instrucción. 90) Se introduce el código de la cuadragésima primera instrucción. 91) Se introduce el código de la cuadragésima segunda instrucción. 92) Se introduce el código de la cuadragésima tercera instrucción. 93) Se introduce el código de la cuadragésima instrucción. 94) Se introduce el código de la cuadragésima primera instrucción. 95) Se introduce el código de la cuadragésima segunda instrucción. 96) Se introduce el código de la cuadragésima tercera instrucción. 97) Se introduce el código de la cuadragésima instrucción. 98) Se introduce el código de la cuadragésima primera instrucción. 99) Se introduce el código de la cuadragésima segunda instrucción. 100) Se introduce el código de la cuadragésima tercera instrucción.

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 +, DATO 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 ultimo, 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 queremos que se ejecute una rutina, cuya dirección colocaremos previamente en las posiciones de memoria \$28 y \$29.

Tras esto, el reloj entrará en funcionamiento.



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.

Habrá, por tanto, dos maneras de utilizar la placa microcomputadora como interfaz:

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:

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

Para la segunda:

```
1 *PROGRAMA MISMOS BUFFER
2 *PARA LA ENTRADA Y SALIDA
3 LDA #$00
4 STA CENBUFP
5 STA RSBUFP
6 LDA #$08
7 STA CENBUFP+1
8 STA RSBUFP+1
9 JSR CENIN ;PROGRAMA CENTRONICS ENTRADA
10 JSR RSOUT ;PROGRAMA RS-232 SALIDA
11 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.

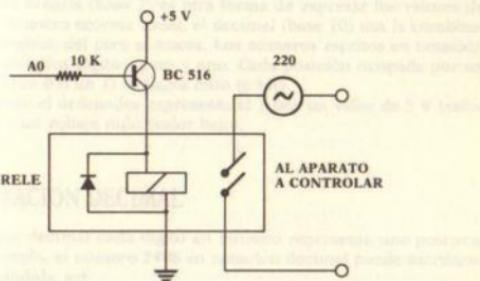


Fig. 6.1. Controlador utilizado por el temporizador.

Aquí se observa el programa encargado de ello:

```
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 PWR ;FINALIZA
6 RTS
```

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:

$$\begin{array}{r} 2 \times 10^3 = 2 \times 1000 = 2000 \\ 4 \times 10^2 = 4 \times 100 = 400 \\ 0 \times 10^1 = 0 \times 10 = 0 \\ 8 \times 10^0 = 8 \times 1 = 8 \\ \hline 2408 \end{array}$$

(Obvio.)

NOTACIÓN 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}	\dots	2^3	2^2	2^1	2^0
16384	8192	4096	\dots	8	4	2	1
			\dots				

El equivalente decimal de 101101 puede calcularse así:

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

$$\begin{array}{ll} 1 \times 2^4 = 0 \times 16 = & 0 \\ 1 \times 2^3 = 1 \times 8 = & 8 \\ 1 \times 2^2 = 1 \times 4 = & 4 + \\ 1 \times 2^1 = 1 \times 2 = & 2 \\ 1 \times 2^0 = 1 \times 1 = & 1 \end{array}$$

45



OPERACIONES LÓGICAS

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 0000000001001101 y el número 67 se representa en 16 bits, como 0000000001000001. 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	0000000001001101
Cambiando bits	1111111101100100
Sumando 1	1
-77 en binario	1111111101100111

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

Para una descripción más detallada de la aritmética binaria búsquese 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). Pongamos 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:

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

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:

$$\begin{array}{r} + 3A \\ BC \\ \hline EC \end{array}$$

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 %: %111111.
- Hexadecimal: Se antecede de \$: \$FF.

APÉNDICE B CONSEJOS A LA HORA DE MONTAR EL CIRCUITO

$$\begin{array}{rcl} & \text{Sumando los términos semejantes:} \\ & 1 + 1 = 2 \quad 2 + 2 = 4 \quad 3 + 3 = 6 \\ & 4 + 4 = 8 \quad 5 + 5 = 10 \quad 6 + 6 = 12 \\ & 7 + 7 = 14 \quad 8 + 8 = 16 \quad 9 + 9 = 18 \end{array}$$

Siendo consciente la ecuación de la complejidad:

$0.2^2 + 0.2^2 + 0.2^2 + 0.3^2 + 0.3^2 + 0.3^2 + 0.4^2 + 0.4^2 + 0.4^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.6^2 + 0.6^2 + 0.6^2 + 0.7^2 + 0.7^2 + 0.7^2 + 0.8^2 + 0.8^2 + 0.8^2 + 0.9^2 + 0.9^2 + 0.9^2$

NOTACIÓN HEXADECIMAL

Las tablas que aparecen en las páginas 8 y 9 de este libro son tablas de conversión entre el sistema decimal (0-9), los sistemas binarios y sus representaciones, así como entre el sistema decimal (0-9) y los sistemas octales y sus representaciones. Asimismo, se incluye una tabla de conversiones entre los sistemas binarios, que se aplica para aquellos que se utilizan para hacer la representación de variables binarias.

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

Por otra parte, una memoriailiaria de 8 bits se compone necesariamente de dos chips hexadecimales.

1000 1000 0000 0000

Algunos resultados de esos resultados de memoriailiaria de 8 bits se expusieron en páginas 10 y 11 de este libro. De acuerdo con lo visto en las páginas 10 y 11, los resultados de memoriailiaria de 8 bits se obtienen sumando los resultados de memoriailiaria de 4 bits cada uno de los cuales tiene una menor complejidad.

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.)

Otro tema importante es el de los conectores. Cuanto mejores, más caros, pero más fiables. En nuestro caso no es necesario demasiado desembolsar, 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, podemos 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 glices. 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 10 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 estanco 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 hablamos 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 hacerlo, 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 patí-

llas (terminales) con los dedos, pueden deteriorarse. Por eso suelen venirse 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.).

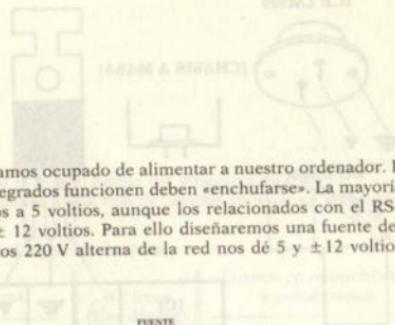
Deseamos suerte y aconsejamos paciencia a todos.

APENDICE C

FUENTE DE ALIMENTACION

SECCIONES

PIEZA 1.C.1



Aún no nos habíamos ocupado de alimentar a nuestro ordenador. Para que los circuitos integrados funcionen deben «encufarse». 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.

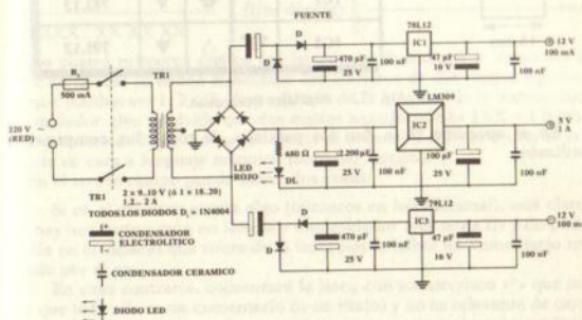


Fig. C.1.

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.

APÉNDICE D LISTADO FUENTE DEL PROGRAMA MONITOR

No debe olvidarse el colocar un disipador de potencia al circuito integrado LM 309 (C12), pues de lo contrario éste se sobre calentaría demasiado, pudiendo llegar a destruirse.

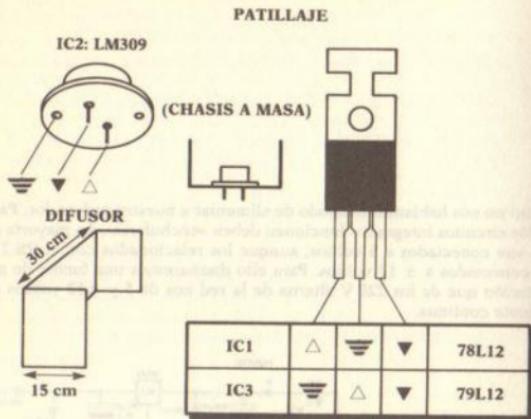


Fig. C.2. Patillajes integrados.

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

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 linea 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)

1. ASM

```

1  ****
2  *
3  *      MONITOR MICRO
4  *      ****
5  *
6  *
7  ****
8  *
9  *      VARIABLES DEL SISTEMA
10 *     INCLUIDA EN PAG. 9
11 *
12 XREG EQU $0001
13 YREG EQU $0002 XOR #1
14 STATUS EQU $0003
15 STKP EQU STATUS#1
16 ACC EQU STKP#1
17 DRL EQU ACC+2
18 DIRH EQU DRL#1
19 RWCDF EQU DIRH#2
20 RWDIF EQU DIRH#3
21 OUTRF EQU H1H4#1
22 AUX EQU OUTRF#6
23 AUX1 EQU AUX#1
24 USTRO EQU AUX#1
25 CENFLAQ EQU USIRO#1
26 CENDUFF EQU CENFLAQ#1
27 RSBUFF EQU CENDUFF#2
28 RSBUFF EQU RSBUFF#1
29 FUNC EQU RSBUFF#2
30 IUSR EQU FUNC#2
31 HRC EQU IURC#2
32 HRS10 EQU HRS#1
33 HRS11 EQU HRS10#1
34 HMIN10 EQU HRS11#1
35 SEC10 EQU HMIN10#1
36 SEC510 EQU SEC10#1
37 HRS1P EQU SEC510#1
38 MINP EQU HRS1P#1
39 SECP EQU MINP#1
40 USREL EQU SECP#1
41 INTYFT EQU USREL#2
42 *
43 * DEFINICION DE LOS VALORES
44 * DE LAS TECLAS DE FUNCION
45 *
46 MAS EQU $10
47 MENOS 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 PBOR EQU $2082 ;DIRECCION PORT B RIOT
58 PBDR EQU $2083 ;DIRECCION PROGRAMAR PORT B RIOT
59 PBOR EQU $2084 ;DIRECCION PORT C RIOT
60 PFPSI EQU $2085 ;FLANCO POSITIVO SIN INTER.
61 PFNCI EQU $2086 ;FLANCO NEGATIVO CON INTER.
62 PFPC1 EQU $2087 ;FLANCO POSITIVO CON INTER.
63 TBSI EQU $2094 ;REG CONTADOR 1 SIN INTER.
64 TBSI EQU $2095 ;REG CONTADOR 2 SIN INTER.
65 T44SI EQU $2096 ;REG CONTADOR 44 CON INTER.
66 T1024SI EQU $2097 ;REG CONTADOR 1024 SIN INTER.
67 TICI EQU $209C ;REG CONTADOR 1 CON INTER.
68 TBCI EQU $209D ;REG CONTADOR 4 CON INTER.
69 T44CI EQU $209E ;REG CONTADOR 44 CON INTER.
70 T1024CI EQU $209F ;REG CONTADOR 1024 CON INTER.
71 *
72 * DIRECCION DE LOS REGISTROS
73 * DE LA VIA
74 *
75 PRVU EQU $2100 ;PORT B DE VIA
76 PRAU EQU $2101 ;PORT A DE VIA CON HANDSHAKE
77 DRBU EQU $2102 ;PROGRAMACION PORT B DE VIA
78 DRAV EQU $2103 ;PROGRAMACION PORT A DE VIA
79 CTIBU EQU $2104 ;PROGRAMACION TI BAJA TI ALTO
80 CTIAU EQU $2105 ;PROGRAMACION TI ALTA TI
81 RT1B EQU $2106 ;REGISTRO TI ALTO
82 RT1A EQU $2107 ;REGISTRO TI ALTO
83 RT2B EQU $2108 ;REGISTRO T2 BAJA
84 RT2A EQU $2109 ;REGISTRO T2 ALTA
85 SRU EQU $210A ;REGISTRO DESPLAZAMIENTO
86 ARU EQU $210B ;REGISTRO ACTIVACION CONTROL
87 TCRU EQU $210C ;REGISTRO DE CONTROL
88 IFRU EQU $210D ;REGISTRO ALARMA INTER.
89 IERU EQU $210E ;REGISTRO ACTIVACION INTER.
90 PASV EQU $210F ;PORT A SIN HANDSHAKE
91 *
92 * COMIENZO DEL CONTENIDO
93 * DE LA EPROM
94 *
95 * DRG $FB00
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 $4B6
107 *
108 *
109 * TABLA DE LAS DIRECCIONES
110 * DE LAS FUNCIONES DEL
111 * EDITOR
112 TABLA DFB $M451/256
113 DFD $M451
114 DFB $M451/256
115 DFB $HENOSI
116 DFB $RHINI/256
  
```

FCB1: DE 117 DFB #RUNI
 FCB1: FD 118 DFB #DIR1/256
 FCB1: FD 119 DFB #DATO/256
 FCB1: 0h 120 DFB #DIR1
 FCB1: BD 121 DFB #DATO
 122 *
 123 * DEFINICION DE REPRESENTACION
 124 * DE CARACTERES EN EL
 125 * DISPLAY
 126 *
 FCB1: 40 79 24 127 C00E DFB \$40,\$79,\$04,\$00 1'0',1',2',3'
 FCB1: 19 12 02 128 DFB \$19,\$12,\$02,\$78 1'4',5',6',7'
 FCB1: 00 10 08 129 DFB \$00,\$10,\$00,\$00 1'8',9',A',B'
 FCB1: 00 24 0C 130 DFB \$46,\$24,\$06,\$0E 1'C',D',E',F'
 FCB1: 00 09 90 131 DFB \$00,\$05,\$00,\$00 1'A',X',Y',P'
 FCB1: 12 7F 132 DFB \$12,\$7F 1'8',
 133 *****
 134 * RUTINA DE RESET *
 135 *
 136 *
 137 *****
 FCB1: 00 138 RES PHP
 FCB1: 85 05 139 STA ACC
 FCB1: 84 01 140 STX XREQ ; SALVA LOS REGISTROS DEL USUARIO
 FCB1: 84 02 141 STA YREQ
 FCB1: 85 05 142 PLA ;SACA DIRECCION DEL PROGRAMA
 FCB1: 00 03 143 PLA STATUS ;TOMA EL STATUS DEL STACK
 FCB1: 00 144 PLA
 FCB1: 00 07 145 STA DIRL
 FCB1: 00 146 PLA
 FCB1: 00 08 147 STA DIRM
 FCB1: A9 0F 148 LDA #00F ;PROGRAMA PORT D RIOT
 FCB1: 00 93 20 149 STA PBDR
 FCB1: 00 00 150 LDY #0000 ;OPCALCUL IN BINARIO
 FCB1: 7B 151 SEI ;INHIBE INTERRUPCIONES
 152 *****
 153 *
 154 * BUCLE PRINCIPAL *
 155 *
 156 *
 FCB1: 20 15 FE 157 ENTRADA JSR LOAD
 FCCD1: 20 07 FD 158 JSR LEENB ;LEE TECLA
 FCC3: C9 10 159 EHTI CMP #10 ;SI ES UN NUMERO
 FCC5: 30 F4 160 BMI ENTRADA ;IGNORALO
 FCC7: C9 161 CMP #014 ;SI ERROR
 FCCY: 10 F2 162 BPL ENTRADA ;IGNORALO
 FCC1: C9 163 BEQ .
 FCC1: C9 164 SBC #010 ;TRANSFORMAR TECLA
 FCC1: 00 165 ASL ;EN VALOR
 FCC1: AW 166 TAX ;PARA CALCULAR DIRECCION
 FCD1: 00 84 FC 167 LDA TABLA,X ;DE LA FUNCION
 FCD3: 05 1B 168 STA FUNC
 FCD5: EB 169 INX
 FCD1: 00 8C FC 170 LDY TABLA,X
 FCB9: 05 1C 171 STA FUNC+1
 FCB1: 4C 1B 00 172 JMP -(FUNC) ;EJECUTA LA FUNCION
 173 *****
 174 *
 175 * RUTINA DE ATENCION *
 176 *
 177 * A RUN *
 178 *
 179 *
 FCD1: A6 04 180 RUNI LDY STKP ; RESTAURA LOS REGISTROS

FCE1: 1B 181 TSX
 FCE1: A5 08 182 LDA DIRH
 FCE1: 00 09 183 PHA
 FCE1: A5 07 184 LDA DIRL
 FCE1: A6 08 185 FHN
 FCE1: A5 03 186 LDA STATUS
 FCE1: 48 187 PHA
 FCE1: A6 01 188 LDY XREQ
 FCE1: A6 02 189 LDY YREQ
 FCE1: A6 05 190 LDA ACC
 FCF1: 40 191 STA . ; EJECUTA LA RUTINA DEL USUARIO
 192 *****
 193 *
 194 * RUTINA DE ATENCION *
 195 *
 196 * A *
 197 *
 198 *
 199 *****
 FCF1: 1B 199 M601 CLC ;INCREMENTA DIRECCION USUARIO
 FCF1: A5 07 200 LDA DIRL
 FCF1: 00 01 201 ADC #01
 FCF1: 05 07 202 STA DIRL
 FCF1: 00 02 203 BCC M602
 FCF1: E8 08 204 INC DIRL
 FCF1: 4C BD FC 205 M602 LDY #0002 ;PR ENTRADA ;IR BUCLE PRINCIPAL
 206 *****
 207 *
 208 * RUTINA DE ATENCION *
 209 *
 210 * A *
 211 *
 212 *****
 FCF1: 3B 213 M603 SEC ; DECREMENTA DIRECCION USUARIO
 FDF0: A5 07 214 LDA DIRL
 FDF0: E9 01 215 SBC #001
 FDF0: 85 07 216 STA DIRL
 FDF0: 90 02 217 BCC M602
 FDF0: C4 08 218 DEC DIRL
 FDF0: 4C BD FC 219 M602 LDY #0002 ;VOLVER BUCLE PRINCIPAL
 220 *****
 221 *
 222 * RUTINA DE ATENCION *
 223 *
 224 * A DAT *
 225 *
 FDD0: A0 19 226 DATO LDY #015 ;PREPARA BUFFER
 FDD0: A2 04 227 DATO LDY #004
 FD13: B4 CB 228 STX MINDF
 FD13: A9 06 229 LDA #004
 FD17: 05 0A 230 STA MAHBF
 FD17: 20 9A FD 231 JSR LEANT ;LEE LOS DATOS
 FD17: A2 00 232 LDY #001
 FD1C: C9 30 FD 233 JSR AUX1
 FD1F: C9 19 234 CPY #013 ;LOS CONVIERTA EN UN BYTE
 FD21: F0 06 235 BEQ DATA2
 FD23: 99 01 00 236 STA XREQ,Y
 FD24: 1B 237 CLC ;SI ES UN REG, LO ALMACENA
 FD24: 20 04 238 BCC DATA2 ;ALMACENA DATO
 FD24: 00 00 239 DAT2 LDY #0001 ;ALMACENA DATO
 FD2B: B1 00 240 STA (DIRL),X
 FD2D: 4C BD FC 241 DATA2 JMP ENTRADA
 242 *
 243 * RUTINA AUXILIAR DE DATO
 244 *
 F030: B5 0C 245 AUX1 LDA OUTBF,X ;CONVIERTA DOS NUMEROS
 F032: 04 246 ASL ;DE CUATRO BITS EN
 F033: 04 247 ASL ;UNO DE OCHO.
 F034: 04 248 ASL
 F035: 04 249 ASL

FD36: E8 250 170C
 FD37: C9 00 251 ORA OUTBF,X
 FD39: 40 252 RTS
 253 *****
 254 * RUTINA DE ATENCION *
 255 *
 256 * A DIR *
 257 *
 258 *
 259 *****
 FD3A: 20 07 FD 260 DIRT JSR LEEKB ;LEE TECLA
 FD3D: C9 10 261 CMP #10
 FD3F: 30 06 262 BMI DIREC ;VER SI ES DAT
 FD41: C5 04 263 CMP DAT ;VA A DIRDAT
 FD43: 00 F5 264 BNE DIRI ;PROGRAMA EL BUFFER DE SALIDA
 FD45: F0 1B 265 BEQ DIRDAT
 FD47: A2 00 266 DIREC LDW #800
 FD48: 00 00 267 STY MINBF
 FD4B: 09 04 268 LDW #1000
 FD4D: 85 06 269 STA MAXBF
 FD4F: 20 94 FD 270 JSR LEDAT ;LEE DIRECCION ALMACINADA
 FD52: A9 00 271 LDA #800
 FD54: 20 12 00 272 JSR AUX
 FD57: 05 08 273 STA DIRH
 FD59: E9 00 274 INX
 FD5A: 20 12 00 275 JSR AUX
 FD5D: 85 07 276 STA DIRL
 FD5F: 4C 80 FC 277 JHP ENTRADA ;VA AL BUCLE PRINCIPAL
 278 *****
 279 *
 280 * RUTINA DE ATENCION *
 281 *
 282 * A DIR DAT *
 283 *
 284 *****
 FD62: 20 07 FD 285 DIRDAT JSR LEEKB ;LEE TECLA
 FD65: C9 04 286 CMP #80A
 FD67: D0 03 287 BNE COATI ;VER SI ES EL ACUMULADOR
 FD68: A0 04 288 LDW #804
 FD6B: 00 00 289 TAY
 FD6C: A9 7F 290 DDATI LDA #87F ;COLOCA ESPACIOS VACIOS
 FD6E: 85 12 291 STA ALDI ;EN EL BUFFER DE SALIDA
 FD70: 85 0C 292 STA OUTBF
 FD72: 05 00 293 STA OUTBF+1
 FD74: 05 0E 294 STA OUTBF+2
 FD75: 89 00 FC 295 LDW CODE,Y
 FD76: 00 00 296 STA OUTBF+3
 FD78: A3 04 297 LDW #804-1
 FD7D: 20 2C FE 298 JSR GUARD ;LO-ESCRIBE
 FD80: 20 87 FD 299 JSR ESCRIBE
 FD83: 20 07 FD 300 JSR LEEKB ;LEE TECLA
 FD86: C5 14 301 CMP DAT ;VE SI ES DAT
 FD88: F0 08 302 BEQ DDATZ
 FD8B: 48 00 303 PWD
 FD8F: 00 15 FE 304 JSR LDAT
 FD8E: 68 00 305 PLA ;CONVIERTER DATOS PARA DISPLAY
 FD8F: 4C C3 FC 306 JHP ENTI
 FD92: A4 12 307 DDATZ LDY AUX ;IR A EJECUTAR DIRDAT
 FD94: 20 0F FD 309 JSR DATI
 FD97: 4C BD FC 309 JHP ENTRADA
 310 *****
 311 *
 312 * LEE UNA DIRECCION *
 313 *
 314 * O UN DATO DEL *
 315 *
 316 * TECLADO *
 317 *
 318 *****

FD9A: A4 08 319 LEDAT LDW MINBF
 FD9B: 20 87 FD 320 LEE1 ;LEE TECLA
 FD9E: C5 11 321 CMP MENOS ;COMPROUEA SI ES MENOS
 FD9F: 00 09 322 BEQ BK
 FD9G: 20 9C 00 323 STA OUTBF,X
 FD9H: 51 20 87 FD 324 JSR ESCRIBE
 FD9I: EB 325
 FD9P: E4 0A 326 CPX MAXBF
 FD9R: 00 00 327 RTI
 FD9C: AF 7F 328 BX LDA #87F ;SI ES MENOS
 FD9E: 95 0C 329 STA OUTBF,X ;RETROcede UNO MAS
 FD9G: CA 330 DEX
 FD9I: E4 0B 331 CPX MINBF
 FD9L: F0 05 332 BEQ LEDAT
 FD9S: 00 05 333 BNE LEI
 334 *****
 335 *
 336 * RUTINA DE ESCRITURA *
 337 *
 338 * EN EL DISPLAY *
 339 *
 340 *****
 FD97: A9 7F 341 ESCRIBE LDW #87F ;PROGRAMA PORT A
 FD99: 80 01 20 342 LDW #805 ;PREPARA BUFFER
 FD9C: A2 05 343 LDW #805
 FD9E: B4 0C 344 ESCI LDY OUTBF,X ;LEE CODIGO
 FD99: 80 90 FC 345 LDA CODE,Y ;CARGA CODIGO DISPLAY
 FD9C: 80 80 20 346 STA PADR ;ESCRIBELO
 FD9E: A4 7F 347 UP LDY #87F ;ESPERA UN RATO
 FD9F: 00 00 348 DEX
 FD9G: 10 FB 349 BPL UP
 FD9C: 80 00 20 350 STY PADR ;REPROGRAMA PORT A
 FD9E: A9 05 351 LDA #805
 FD99: 80 80 20 352 STA PADR ;REPROGRAMA PORT B
 FD93: CA 353 DEX
 FD94: 10 EB 354 BPL ESCI ;FIN ?
 FD9E: 60 355 RTS
 356 *****
 357 *
 358 * LECTURA DE UNA *
 359 *
 360 * TECLA *
 361 *
 362 *****
 FD97: A9 00 363 LCCIB LDA #800 ;PROGRAMA PORT
 FD99: 80 81 20 364 CTA PADR
 FD9C: A2 06 365 LDW #805
 FD9E: 8E 20 366 STX PADR
 FD9E: A4 80 20 367 LEE1 LDA PADR
 FD94: D6 08 368 BNE SI
 FD9E: 10 08 369 HLT
 FD9E: E0 0F 370 CPX #809
 FD9E: D0 06 371 BNE LEE1
 FD9B: 20 87 FD 372 JSR ESCRIBE
 FD9E: 4C D7 FD 373 JHP LEEKB
 FD9F: 84 13 374 SI STX AUX1
 FD93: 48 375 PHA
 FD94: 20 87 FD 376 JSR ESCRIBE
 FD9E: 00 00 377 PEA
 FD9B: A4 13 378 LDX AUX1
 FD9A: A9 7F 379 LEE2 LDY #87F
 FD9C: 88 380 DEY
 FD9D: 1D FB 381 DPL LEE2
 FD9F: AD 90 382 LDA PADR
 FD92: 85 12 383 STA AUX1
 FD9E: 00 00 384 TAY
 FD9C: 38 385 SEC
 FD9A: E9 04 386 SBC #806
 FD9B: F0 08 387 BEQ LEFIN

FEDAI_18 388 LEE3 CLC
 FE007_49 16 389 LDA #005
 FE00D_45 12 390 AND ADR
 FE0F1_0A 391 DEX
 FE1G1_00 FB 392 BNE LEE3
 FE1G1_05 12 393 LEFIN LDA ADR
 FE1E1_60 394 RTS
 395 ****
 396 *
 397 * CARGA DATOS EN *
 398 *
 399 * BUFFER DEL DISPLAY *
 400 *
 401 ****
 FE1G1_05 08 402 LDAT LDA DIRH ;ALMACENA PARTE ALTA Y
 FE1G1_05 09 403 LDA DCRH ;BAJA DE DIRECCION EN EL
 FE1F1_20 3C FE 404 JSR GUARDA ;BUFFER DE SALIDA
 FE1C1_05 07 405 LDA DCRH
 FE1C1_20 2C FE 406 JSR GUARDA
 FE21_40 00 407 LDY #0
 FE23_81 07 408 LDA (DIRH),Y ;CARGA CONTENIDO-DIRECCION
 FE23_20 2C FE 409 JSR GUARDA
 FE28_20 87 FD 410 JSR ESCRIBE
 FE2B1_40 411 RTS
 412 *
 413 * RUTINA AUXILIAR DE LDAT
 414 *
 FE2C1_40 415 GUARDA PHA ;CONVIERTER NUMERO
 FE2D1_25 F0 416 AND #FF0 ;DE OCHO BITS
 FE2D1_04 417 ASL ;EN DOS DE CUATRO BITS
 FE301_04 418 ASL
 FE31_04 419 ASL
 FE32_04 420 ASL
 FE33_15 95 DC 421 STA OUTBUF,X
 FE35_15 EB 422 INX
 FE36_15 423 PLA
 FE37_15 0F 424 AND #FF
 FE39_15 95 DC 425 STA OUTBUF,X
 FE3B_15 EB 426 THY
 FE3C1_40 427 RTS
 428 ****
 429 *
 430 * PROG. CENTRONICS *
 431 *
 432 ***
 433 *
 434 * PROGRAMACION DEL PORT A
 435 * COMO ENTRADA/SALIDA
 436 * INTERFACE CENTRONICS
 437 *
 438 * PARA LEER EL PUNTO DE ENTRADA
 439 *
 440 * ES INCON
 441 * PARA ESCRIBIR ES OUTCENT
 442 *
 FE3D1_A9 00 443 INCENT LDA #0 ;RUTINA DE ENTRADA
 FE3F1_80 09 21 444 STA DRW ;PROGRAMA REG. DIRECCION
 FE42_85 15 445 STA CENFLAG
 FE42_85 50 FE 446 JSR CNT
 FE47_15 447 OUTCENT STA DRW ;RUTINA SALIDA
 FE49_80 09 21 448 STA DRW ;PROGRAMA REG. DIRECCION
 FE4C1_09 01 449 LDA #001
 FE4E1_85 15 450 STA CENFLAG
 FE50_15 00 451 CNT LDA #0 ;PARTE COMUN
 FE52_80 01 21 452 STA PWV ;REGISTRO FLAGS INT.
 FE53_15 DC 21 453 LDA PCRV ;REGISTRO DE CONTROL
 FE54_15 00 454 AND #0000110000
 FE5A1_0F 08 455 ORA #000001000
 FE5C1_80 0C 21 456 STA PORU
 FE5F1_4D AD 21 457 *
 FE62_29 FE 458 *
 FE641_09 01 460 AND #000000001
 FE651_80 08 21 461 STA ACRU
 FE771_4D 462 *
 FE691_4F 83 463 LDA ACRU ;REGISTRO AUXILIAR DE CONTROL
 FE77_10 0E 21 464 ORA IFRV ;ACTIVACION DE INTER.
 FE6E1_80 0E 21 465 STA IERU
 FE73_49 00 466 LDA #0
 FE73_80 00 21 467 STA IFRV
 FE76_38 468 CLI
 FE77_40 469 RTS
 470 ****
 471 *
 472 * PROG. RS = 232/V 24 *
 473 *
 474 *
 475 * MAY. DEC. PUNTOS DE ENTRADA
 476 * INRS PARA LA RUTINA DE ENTRADA
 477 * Y OUTRS PARA LA DE SALIDA
 478 *
 FE781_80 21 479 INRS LDA ACRU ;PROGRAMAR EL REGISTRO AUXILIAR
 FE77_29_EI 480 AND #011100001 ;DE CONTROL
 FE7D_09 0E 481 ORA #000000110
 FE7F_80 00 21 482 STA ACRU
 FE82_80 00 21 483 LDA PCRV ;IDEM REGISTRO DE CONTROL
 FE85_29 0F 484 AND #000000111
 FE86_80 00 485 STA PCRV
 FE87_80 00 21 486 STA PCRV
 FE8C_80 00 487 LDA #0 ;RSFLAG0 -> ENTRADA
 FE8E_90 18 488 STA RSFLAG
 FE90_4C AB FE 489 JHP (CONT)
 FE93_80 21 490 OUTRS LDA ACRU ;IDEM REGISTRO AUXILIAR DE
 FE94_29 EI 491 AND #011100001 ;CONTROL
 FE95_01 492 ORA #000000110
 FE96_80 00 21 493 STA PCRV
 FE97_80 00 21 494 LDA PCRV ;IDEM REGISTRO DE CONTROL
 FE93_29 0F 495 AND #000000111
 FE94_09 00 496 ORA #000000000
 FE94_80 00 21 497 STA PCRV
 FE97_80 00 498 LDA #1 ;RSFLAG -> SALIDA
 FE98_4C AB FE 499 STA RSFLAG
 FE99_80 00 21 500 STA IFRV ;ACTIVANDO LA INTERRUPCION
 FE9D_80 00 21 501 STA IERU ;CORRESPONDIENTE
 FE9D_80 0E 21 502 STA IFRV
 FE93_09 84 503 ORA #010000100
 FEB91_80 0E 21 504 STA ICRU
 FEB91_58 505 CLI
 FEB91_40 506 RTS
 507 ****
 508 *
 509 * RUTINA DE ATENCION A *
 510 *
 511 * IRQ *
 512 *
 513 ****
 FEB91_4D AD 21 510 IRQ0 LDA IFRV ;IRON = LA RUTINA QUE
 FEB91_29 80 514 AND #010000000 ;ATIENDE A LA INTERRUPCION
 FEB91_00 32 514 BEQ IROI ;DEL INTERFACE CENTRONICS
 FEC1_40 00 21 517 LDA IFRV
 FEC1_29 02 518 AND #000000010 ;DEPENDIENDO DEL CONTENIDO
 FEC1_40 28 519 BEQ IROI ;DE CENFLAG REALIZA ENTRADA
 FEC6_40 00 520 LDA CENFLAG
 FEC9_04 14 521 BEQ IROI ;O SALIDA
 FEC1_45 15 522 CENDOUT LDA CENBUFF ;CENDOUT = ES UN VECTOR QUE
 FEC1_40 23 523 BEQ IROI ;INDICA EL INICIO DEL
 FED9_04 16 524 DEC CENBUFF ;BUFFER DEL INTERFACE
 FED2_8A 525 TIA

FED3: 48 526 PNA ;(DEFINIBLE POR EL USUARIO)
 FED4: A2 00 527 LDY #0
 FED6: A1 16 528 LDY (CENBUFF,X)
 FED8: B0 01 529 STA PWV
 FEDC: 48 530 PLA
 FEDC: E4 531 TXA
 FEDD: 4C F3 FE 532 JMP IRO1
 FEE0: A5 14 533 CENIN LDA CENBUFF
 FEE2: C9 FF 534 CMP #0
 FEE4: F0 00 535 BEQ IRO1
 FEE6: E6 14 536 INC CENBUFF
 FEE9: 49 537 TXA
 FEEA: A2 00 538 PNA
 FEEC: 00 539 LDY #0
 FEC1: AD 01 21 540 LDY PWV
 FEEF: B1 14 541 STA (CENBUFF,X)
 FEF1: 68 542 PLA
 FEF2: A4 543 TAX
 FEF3: AD 00 21 544 IRO1 LDA IFRV
 FEF6: 29 80 545 AND #0100000000 ;IRO1 ES LA RUTINA QUE
 FEF7: 29 80 546 BEQ IRO2 ;ATIENDE A LA INTERRUPCIÓN
 FEF9: 21 1C 548 LDY #0
 FEF9: F0 28 549 AND #00000000100 ;DEL INTERFACE RS - 232
 FEF9: F0 14 550 BEQ IRO2 ;RSFLAG INDICA LA DIRECCIÓN
 FEF9: F0 14 551 LDY RSFLAG
 FEF9: F0 19 552 RSBOUT LDY RSBUF
 FEF9: F0 19 553 DEO RSIN ;LA RUTINA ES SIMILAR A
 FEF9: F0 19 554 DEO IRO2 ;LA DEL INTERFACE CENTRONICS
 FEF9: F0 19 555 DEC RSBUF
 FFF0: B8 556 LDY #0
 FFF0: C1 48 556 PNA
 FFF0: A2 00 557 LDY #0
 FFF0: A1 19 558 LDA (RSBUFP,X)
 FFF0: AD 00 21 559 STA PBU
 FFI4: 40 560 PLA
 FFI5: A4 561 TAX
 FFI6: 4C 2C FF 562 JMP IRO2
 FFI9: A5 19 563 RSIN LDA RSBUF
 FFI9: C9 FF 564 CMP #0
 FFI9: F0 00 565 BEQ IRO2
 FFI9: E4 19 566 INC RSBUF
 FFI2: 21 00 567 TXA
 FFI2: 21 00 568 PNA
 FFI2: A2 00 569 LDY #0
 FFI2: AD 00 21 570 LDA PWV
 FFI6: B1 19 571 STA (RSBUFP,X)
 FFI6: 68 572 PLA
 FFI6: A4 573 TAX
 FFI6: 4C 1D 00 574 IRO2 JMP (IRUSR) ;IRUSR ES EL VECTOR DE
 575 * COMIENZO DE RUTINA IBO DEL UCR.
 576 * *****
 577 *
 578 * RUTINA DE CONTROL DEL
 579 * TEMPORIZADOR
 580 *
 581 *
 582 * RUTINA DE APLICACIÓN DE LOS
 583 * TEMPORIZADORES PARA LA
 584 * ACTIVACIÓN DE 010000111008
 585 * EXTRACIÓN
 FFF1: A9 FF 586 INREL LDA #IREL/254 ;COLOCA VECTOR USUARIO
 FFF1: 05 1E 587 STA IRUSR+1
 FFF3: 01 00 588 STA IRUSR+1
 FFF3: 01 00 589 STA IRUSR
 FFF8: AD 00 21 590 STA IERU ;INICIALIZA CONTADOR I DE VÍA
 FFF8: 09 C0 591 STA #0
 FFF3: 01 00 592 STA IRUSR
 FFF3: 01 00 593 LDA ACRV
 FFF3: 09 40 594 ORA #0

FF45: 80 00 21 595 STA ACRV
 FF46: A8 22 596 CLI
 FF49: A9 22 597 LDY #022 ;CARGA EL RELOJ CON CUENTA
 FF4B: 80 00 21 598 STA CT1BV ;1/16 DE SEGUNDO
 FF4C: 80 00 21 599 STA #074
 FF50: 80 05 21 600 STA CT1BV
 FF53: A2 00 401 LEHORA LDY #0 ;INICIALIZA CON LA HORA
 FF55: 00 00 402 STA MNDF ;ACTUAL
 FF57: A9 04 403 STA MNDFR
 FF59: B5 04 404 STA SECS
 FF5B: 20 9A FD 405 STA LEDAT ;ELDER LA HORA DE ACTUACION
 FF5F: 20 13 00 406 STA #005
 FF61: 20 13 00 407 STA HR1
 FF63: 20 13 00 408 JSR AUX1
 FF66: 05 21 409 STA MH1
 FF69: 20 10 00 410 JSR AUX1
 FF6B: B5 23 411 STA SECS
 FF6D: A2 00 412 LDY #0
 FF6F: 20 00 413 STA MNDF
 FF70: 00 00 414 LDA #005
 FF73: B5 04 415 STA MAXBF
 FF75: 20 9A FD 416 JSR LEDAT
 FF76: 20 13 00 417 JSR AUX1
 FF7B: 05 25 418 STA #005
 FF7D: 20 13 00 419 JSR AUX1
 FF7E: 20 13 00 420 STA MH1P
 FF81: 20 13 00 421 JSR AUX1
 FF85: 05 27 422 STA DECSF
 FF87: 4C A4 FC 423 JMP RES ;REGRESA AL BUCLE PRINCIPAL

624 *
 625 *
 626 * RUTINA DE ATENCIÓN INTEN-
 627 * RRUCCIÓN DE APLICACIÓN *
 628 * DEL TEMPORIZADOR *
 629 *
 630 ****
 FFB1: 4B 631 IREL PNA ;SALIDA REGISTROS
 FFB1: B4 632 STA PNA
 FFB1: 00 633 PNA
 FFB1: C4 2A 634 DEC INTENT ;COMPRUEBA FIN DE CUENTA
 FFB1: D0 40 635 BNE FINTR ;ACTUALIZANDO LOS CONTADORES
 FFB1: A9 10 636 LDA #16 ;DE HORA,MINUTOS Y SEGUNDOS
 FFB3: 05 2A 637 STA INTNT
 FFB3: A0 30 638 LDA #030
 FFB7: E4 23 639 INC SECS
 FFB9: A2 3A 640 LDY #004
 FFB9: E4 23 641 CPX SECS
 FFB9: E4 23 642 BNE FINTR
 FFB9: B5 23 643 STA DECS
 FFA1: E4 24 644 INC SECS10
 FFA3: A2 36 645 LDY #036
 FFA5: E4 24 646 CPX SECS10
 FFA7: 05 25 647 BNE FINTR
 FFA7: 05 25 648 STA SECS10
 FFB8: E4 21 649 INC MIN
 FFB8: A2 30 650 LDY #030
 FFB8: E4 21 651 CPX MIN
 FFB8: 00 26 652 BNE FINTR
 FFB8: 05 21 653 STA MIN10
 FFB8: E5 22 654 INC MIN10
 FFB8: 05 22 655 LDY #036
 FFB8: E4 22 656 CPX MIN10
 FFB8: D0 21 657 BNE FINTR
 FFB8: B5 22 658 STA MIN10
 FFB8: E6 1F 659 INC HRS
 FFC1: A2 3A 660 LDY #03A
 FFC3: E4 1F 661 CPX HRS
 FFC5: D0 07 662 BNE OTRODIA
 FFC7: B5 05 663 STA HRS

```

FFC9: E6 20 664 INC HRS10
FFC9: AC DE FF 665 JMP FINTR
FFC9: C1 A2 34 666 LDX #834 ;SI SON 24 HORAS, CAMBIA DIA
FFC9: 64 670 CPX HRS
FFC9: 64 671 BNE FINTR
FFD4: A2 32 649 LDX #832
FFD4: E4 20 670 CPX HRS10
FFD4: 00 04 671 BNE FINTR
FFD4: 85 20 672 STA HRS10
FFC1: 85 1F 673 STA HRS
FFD4: 00 03 674 FINTR LDA SECS ;COMPROBAR SI ES LA HORA
FFD4: 00 0F 675 CMP #E83P ;PROGRAMADA
FFE4: A5 21 677 LDA MIN
FFE4: C5 26 678 CMP MINP
FFE8: D0 09 679 BNE FI
FFE8: A5 1F 680 LDA HRS
FFE8: C5 25 681 CMP HRSP
FFE8: 00 03 682 BNE FI
FFD0: E4 20 00 683 JSR (IREL) ;EJECUTA FUNCION USUARIO
FFE3: AD 04 21 684 FI LDA CT100 ;RESTAURA FLAG INTER.
FFF6: 68 685 PLA ;RESTAURA REGISTROS
FFF7: AA 686 TAX
FFF8: 68 687 PLA
FFF9: 40 688 RTI
689 *
690 * - DIRECCIONES DE INTERRUPCIONES
691 *

FFFFA1: A6 692 NMII DFB NRES
FFFFB1: FC 693 DFB #RES/254
FFFFC1: A6 694 RESET DFB #RES
FFFFD1: FC 695 DFB #RES/254
FFFFE1: A6 696 IRO DFB #IRO0/254
FFFFF1: FE 697 DFB #IRO0/254

```

--End assembly--

2048 bytes

Errors: 0

Symbol table = alphabetical order:

AUX	=#0%	ACRV	=#2108	AUX	=#12	AUX1	=#10
CENIN	=#FE00	CENDOUT	=#FECC	CENBUFP	=#16	CENFLAG	=#15
CONTI	=#FEAB	CTIAU	=#FC90	CODE	=#FC90	CONT	=#FE50
DATI	=#FD0F	DATZ	=#FD29	DATFIN	=#FD04	DAT	=#F14
DATAI	=#FD06	DATAZ	=#FD02	DATI	=#FD00	DATD	=#FD00
DIRINT	=#FD62	DIREC	=#FD47	DIRH	=#08	DIRL	=#07
DIREC	=#FD103	DRBU	=#2102	ENTI	=#FCC0	DIREADA	=#FC80
ESCI	=#FD0E	ESCIRBE	=#FD07	FI	=#FFF9	FINTR	=#FFD6
FUNC	=#16	FUNCIONH	=#FC0C	HRS	=#1F	HRS10	=#20
HRSP	=#25	IERW	=#210E	IFRU	=#210D	INCENT	=#FE30
? INREL	=#FF2F	INRS	=#FE76	INTDIT	=#210D	INTCEN	=#FE47
? IRO	=#FFFE	IRO0	=#FE84	IRO1	=#FEF9	IRO2	=#FFC4
? IREL	=#10	LDAT	=#FE15	LEI	=#FD7C	LEDAT	=#FD20
LEEFIN	=#FE11	LEEFZ	=#FDFA	LEEIS	=#FE0A	LEEXB	=#FD07
LEFIN	=#FE12	LEEZ	=#FD73	MAS	=#10	MASI	=#FCF1
MAS2	=#FCFC	HABFE	=#00	MASB	=#11	MENOS1	=#FCF1
MENOS2	=#26	MD00A	=#MIN	MIM0	=#27	MIMB	=#0B
MIMP	=#26	NMI	=#FFFA	OTRODIA	=#FCCE	OUTRF	=#DC
OUTCINT	=#FE47	OUTRS	=#FE93	PADCR	=#2048	PADR	=#2049
? PASV	=#210F	PASV	=#2101	PBC06	=#2069	PBR	=#2082
? PBR	=#2100	PCRV	=#210C	PFNC1	=#2088	? PFNS1	=#2084
? PFRCP1	=#2107	PFRCP1	=#2085	RES	=#FD46	RESET	=#FFFC
RSBUFF	=#19	RSFLAO	=#19	RSIN	=#FF19	RSOUT	=#FF05

? RTIA	=#2107	RTIB	=#2106	RT2A	=#2109	RT2B	=#2108
? RTIC	=#2108	RUNI	=#E0F1	? SEC0	=#225	SEC10	=#24
SECP	=#27	SECF1	=#2097	? T1024BI	=#2097	T1024SI	=#2097
STIP	=#04	T1024CI	=#209F	? T1024BI	=#2097	T1024SI	=#2097
? T1G1	=#2094	T64CI	=#2099	? T48SI	=#2096	T48CI	=#2090
? T8SI	=#2095	TABLA	=#FC88	UP	=#FDCA	USIR0	=#14
USREL	=#20	REG	=#01	YREG	=#02		

Symbol table = numerical order:

XREG	=#01	YREG	=#02	STATUS	=#00	STKP	=#04
ACC	=#05	DIRL	=#07	DIRH	=#00	PAW	=#0A
MINBF	=#08	OUTBF	=#0C	? MAS	=#10	MENOS	=#11
ADU	=#12	? RUN	=#12	AUX1	=#13	? DIR	=#13
USITRO	=#14	? T1024CI	=#14	CENFLAG	=#15	CENBUFF	=#16
RSFLAO	=#10	RSBUFP	=#17	PLAR	=#18	IRUSR	=#10
HRS	=#20	HRS10	=#20	HIN	=#21	MILIO	=#20
SECS	=#23	SECS10	=#24	HRSP	=#25	MIMP	=#26
SECP	=#27	USREL	=#28	INTONT	=#29	PASV	=#2080
PADCR	=#2081	PBR	=#2082	PBDCR	=#2083	? PTNS1	=#2084
? PFPSC1	=#2085	? PFNC1	=#2086	? PFPCL	=#2087	TISI	=#2094
? T1024CI	=#2089	T1024SI	=#2097	? T1024CI	=#2097	T1024SI	=#2097
PW1	=#2101	DRBV	=#2102	DRBV	=#2103	CT10V	=#2103
CT10V	=#2105	? RTIB	=#2106	? RTIA	=#2107	? RT2B	=#2103
? RT2A	=#2109	? SRU	=#2104	ACRU	=#2108	PCR0	=#210C
? IFRU	=#2100	IERV	=#210E	? PASV	=#210F	TABLA	=#FC96
CODE	=#FC90	RES	=#FC06	ENTRAD	=#FCBC	DIR1	=#FC33
? FCB0	=#FC07	? FC08	=#FC07	? FC09	=#FC07	? FC0A	=#FC07
MD00S2	=#FD64	DATO	=#FD00	DIRT1	=#FD0F	DIRT2	=#FD09
DATAFH	=#FD20	AUX1	=#F030	DIR1	=#FD34	DIRC4	=#FD47
DIRINT	=#FD62	DIRT1	=#FD4C	DIRT2	=#FD92	LCDAT	=#FD9A
LE1	=#FD94	BK	=#FDAC	ESCRIBE	=#FD87	E8CI	=#FD8E
UP	=#FD06	LCB#0	=#FD07	LEE1	=#FDE1	SI	=#FD01
LEE2	=#FDFA	LEE3	=#FE04	LEEFH	=#FE12	LDAT	=#FE15
DIRDW	=#FE2C	? INCENT	=#FE30	? OUTCEN	=#FE47	CONT	=#FE50
? INREL	=#FF2F	? T1024CI	=#2097	CONT1	=#FEAB	IR00	=#FEB4
? CENOUT	=#FECC	CENIN	=#FE09	? FEAD	=#FED0	? PHROUT	=#F057
RSIN	=#FF19	IRO2	=#FF2C	? INEL	=#FF2F	? LEHORA	=#FF53
IREL	=#FFBA	OTRODIA	=#FFCE	FINTR	=#FFDE	FI	=#FFF3
? NMII	=#FFFA	? RESET	=#FFFC	? IR0	=#FFFF		

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
23F8- FF FF 00 00 FF FF 57 06
2400- FF FF 00 00 FF FF 00 00
240C- 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 00 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 DB 78 20 15 FE
24C0- 20 D7 FD C9 10 30 F6 C9
24C8- 14 10 F2 38 E9 10 0A AA
24D0- BD 86 FC 85 1B 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 A3 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 0B 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 D0 F5 F0 1B A2
2548- 00 B6 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 D0
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 0B 48 20 15 FE 6B 4C
2590- C3 FC A4 12 20 0F FD 4C
2598- BD FC A6 0B 20 D7 FD C5
25A0- 11 F0 09 95 0C 20 B7 FD
25AB- E0 E4 0A 60 A9 7F 95 0C
25B0- CA E4 0B F0 E5 D0 E5 A9
25B8- 7F 8D 81 20 A2 05 B4 0C
25C0- B9 90 FC 90 80 20 A0 7F
25CB- 8D 10 FB 8C 80 20 A9 06
25D0- BD 82 20 CA 10 E8 60 A9
25D8- 00 8D 81 20 A2 06 8E 82
25E0- 20 AD 80 20 00 0B E8 E0
25E8- 09 D0 F6 20 B7 FD 4C D7
25F0- FD 86 13 48 20 B7 FD 6B
25F8- A6 13 A0 7F 88 10 FB AD
2600- 80 20 85 12 8A 38 E9 06
2608- F0 0B 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 6B 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 BD 0C 21 AD
2660- 0B 21 29 FE 09 01 8D 0B
2668- 21 A9 83 00 0E 21 8D 0E
2670- 21 A9 00 8D 0D 21 58 60
2678- AD 0B 21 29 E1 09 0E 8D
2680- 0B 21 AD 0C 21 29 0F 09
2688- 80 8D 0C 21 A9 00 85 18
2690- 4C AB FE AD 0B 21 29 E1
2698- 09 1E 8D 0B 21 AD 0C 21
26A0- 29 0F 09 00 8D 0C 21 A9

APENDICE F CODIGOS PROGRAMACION

26A8- 01 85 18 A9 00 80 0D 21
26B0- AD 0E 21 09 84 8D 0E 21
26B8- 58 60 0D 01 21 29 80 F0
26C0- 32 AD 0D 21 22 02 F0 28
26C8- A5 15 F0 14 A5 16 F0 23
26D0- C6 14 8A 40 A2 00 A1 1C
26D8- 8D 01 21 68 8A 4C F3 F0
26E0- A5 15 C9 FF 00 0D E6 16
26E8- 8A 48 A2 00 AD 01 21 81
26F0- 14 68 AA AD 0D 21 29 80
26F8- F0 32 AD 0D 21 29 1C F0
2700- 28 A5 18 F0 14 A5 19 F0
2708- 23 C6 19 8A 48 A2 00 A1
2710- 19 80 00 21 68 AA 4C 2C
2718- FF A5 19 C9 FF F0 00 ED E6
2720- 19 8A 4C A2 00 AD 00 21
2728- 81 19 68 AA 6C 1D 00 A9
2730- FF 85 1E AD BA FF 85 1D
2738- AD 0E 21 09 C0 8D 0E 21
2740- AD 0B 21 09 40 80 0B 21
2748- 58 A9 22 80 04 21 A9 F4
2750- 8D 05 21 A2 00 86 BB AB
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 24 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 23 D0 DF 85
28A0- 23 E6 24 A2 36 E4 24 D0
27AB- 35 85 24 E6 21 A2 3A E4
2780- 21 D0 28 85 21 E6 22 A2
27B8- 36 E4 22 D0 21 95 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 B2 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 00 AD 04 21 20 AA
27F8- 48 40 04 FC A4 EC BC FF

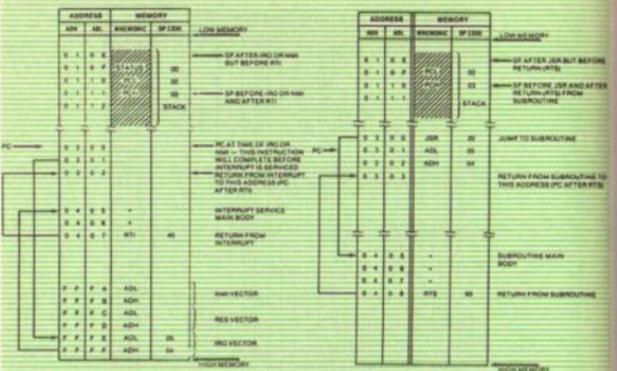
OPERATION CODE TABLE

	A	B	C	D	E	F
INT	INT-A	INT-B	INT-C	INT-D	INT-E	INT-F
INT-A	INT-A-A	INT-A-B	INT-A-C	INT-A-D	INT-A-E	INT-A-F
INT-B	INT-B-A	INT-B-B	INT-B-C	INT-B-D	INT-B-E	INT-B-F
INT-C	INT-C-A	INT-C-B	INT-C-C	INT-C-D	INT-C-E	INT-C-F
INT-D	INT-D-A	INT-D-B	INT-D-C	INT-D-D	INT-D-E	INT-D-F
INT-E	INT-E-A	INT-E-B	INT-E-C	INT-E-D	INT-E-E	INT-E-F
INT-F	INT-F-A	INT-F-B	INT-F-C	INT-F-D	INT-F-E	INT-F-F
INT-G	INT-G-A	INT-G-B	INT-G-C	INT-G-D	INT-G-E	INT-G-F
INT-H	INT-H-A	INT-H-B	INT-H-C	INT-H-D	INT-H-E	INT-H-F
INT-I	INT-I-A	INT-I-B	INT-I-C	INT-I-D	INT-I-E	INT-I-F
INT-J	INT-J-A	INT-J-B	INT-J-C	INT-J-D	INT-J-E	INT-J-F
INT-K	INT-K-A	INT-K-B	INT-K-C	INT-K-D	INT-K-E	INT-K-F
INT-L	INT-L-A	INT-L-B	INT-L-C	INT-L-D	INT-L-E	INT-L-F
INT-M	INT-M-A	INT-M-B	INT-M-C	INT-M-D	INT-M-E	INT-M-F
INT-N	INT-N-A	INT-N-B	INT-N-C	INT-N-D	INT-N-E	INT-N-F
INT-O	INT-O-A	INT-O-B	INT-O-C	INT-O-D	INT-O-E	INT-O-F
INT-P	INT-P-A	INT-P-B	INT-P-C	INT-P-D	INT-P-E	INT-P-F
INT-Q	INT-Q-A	INT-Q-B	INT-Q-C	INT-Q-D	INT-Q-E	INT-Q-F
INT-R	INT-R-A	INT-R-B	INT-R-C	INT-R-D	INT-R-E	INT-R-F
INT-S	INT-S-A	INT-S-B	INT-S-C	INT-S-D	INT-S-E	INT-S-F
INT-T	INT-T-A	INT-T-B	INT-T-C	INT-T-D	INT-T-E	INT-T-F
INT-U	INT-U-A	INT-U-B	INT-U-C	INT-U-D	INT-U-E	INT-U-F
INT-V	INT-V-A	INT-V-B	INT-V-C	INT-V-D	INT-V-E	INT-V-F
INT-W	INT-W-A	INT-W-B	INT-W-C	INT-W-D	INT-W-E	INT-W-F
INT-X	INT-X-A	INT-X-B	INT-X-C	INT-X-D	INT-X-E	INT-X-F
INT-Y	INT-Y-A	INT-Y-B	INT-Y-C	INT-Y-D	INT-Y-E	INT-Y-F
INT-Z	INT-Z-A	INT-Z-B	INT-Z-C	INT-Z-D	INT-Z-E	INT-Z-F

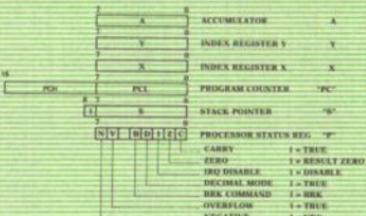
- (1) ADD 1 TO "N" IF PAGE BOUNDARY IS CROSSED
- (2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE
- (3) BRANCH NOT = BRANCH
- (4) IF IN DECIMAL MODE, SET UP DECIMAL POINT
- (5) ADD ALIGNED ALU RESULT BY CHECKED SIGN ZERO RESULT

Nota: Continúa en la página siguiente.

FIG. 1 IRQ, NMI, RTI, BRK OPERATION



PROCESSOR PROGRAMMING MODEL



BACKWARD RELATIVE BRANCH TABLE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S	128	127	126	125	124	123	122	121	120	119	118	117	115	115	114	113
R	112	111	110	109	108	107	106	105	104	103	102	101	101	99	98	97
A	95	95	94	93	92	91	90	85	85	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	40	39	38	37	36	35	34	33	32
E	33	31	30	29	28	27	26	25	24	23	21	20	19	18	17	16
F	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

FIG. 2 JSR, RTS OPERATION



FORWARD RELATIVE BRANCH TABLE

LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
2	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7
3	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
4	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
5	80	71	62	53	44	35	26	17	8	1	125	124	123	122	121	120
6	82	73	64	55	46	37	28	19	10	1	123	122	121	120	119	118
7	112	112	114	115	116	117	118	119	120	121	122	123	124	125	126	127

COMPARE INSTRUCTION RESULTS

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

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

HEXADECIMAL AND DECIMAL CONVERSION

HEXADECIMAL COLUMNS																	
8	6	4	3	2	1	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1	1	1	1	1	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2	2	2	2	2	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3	3	3	3	3	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4	4	4	4	4	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5	5	5	5	5	5	5
6	6,291,456	6	393,316	6	24,576	6	1,536	6	96	6	6	6	6	6	6	6	6
7	7,340,032	7	459,052	7	29,368	7	1,792	7	112	7	7	7	7	7	7	7	7
8	8,398,608	8	524,288	8	32,768	8	2,048	8	128	8	8	8	8	8	8	8	8
9	9,457,184	9	589,824	9	36,894	9	2,304	9	144	9	9	9	9	9	9	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10	A	10	A	10	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11	B	11	B	11	B	11
C	12,582,912	C	787,520	C	50,016	C	3,272	C	128	C	12	C	12	C	12	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13	D	13	D	13	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14	E	14	E	14	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15	F	15	F	15	F	15
	7654		3210		7654		3210		7654		3210		3210		3210		3210

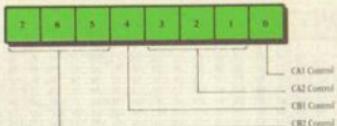
POWERS OF 2

2^n	n	2^6	= 64
256	8	2^4	= 16
512	9	2^5	= 32
1,024	10	2^6	= 64
2,048	11	2^7	= 128
4,096	12	2^8	= 256
8,192	13	2^9	= 512
16,384	14	2^10	= 1,024
32,768	15	2^11	= 2,048
65,536	16	2^12	= 4,096
131,072	17	2^13	= 8,192
262,144	18	2^14	= 16,384
524,288	19	2^15	= 32,768
1,048,576	20	2^16	= 65,536
2,097,152	21	2^17	= 131,072
4,194,304	22	2^18	= 262,144
8,388,608	23	2^19	= 524,288
16,777,216	24	2^20	= 1,048,576

POWERS OF 16

16^n	n
16	0
256	1
4,096	2
65,536	3
1,048,576	4
2,097,152	5
4,194,304	6
8,388,608	7
16,777,216	8

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.

CBI CONTROL

PCR4 = 0 - The CBI Interrupt Flag (IFR3) will be set by a negative transition (high to low) on the CBI pin.

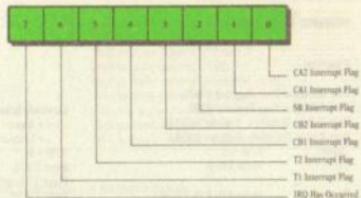
= 1 - The CBI Interrupt Flag (IFR3) will be set by a positive transition (low to high) on the CBI pin.

CB2 CONTROL

PCR7 PCR6 PCR5 Mode

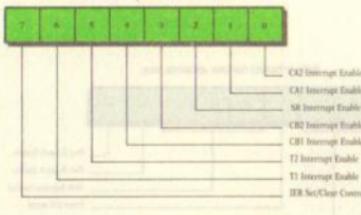
0	0	0	CB2 negative edge interrupt (IFR3/ORB clear) mode - Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the ORB or by writing logic 1 into IFR3.
0	0	1	CB2 negative edge interrupt (IFR3 clear) mode - Set IFR3 on a negative transition of 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 for 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 SK (\$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 T2C-L (\$A008) or writing T2C-H (\$A009)
6	Time-out of Timer 1	Reading T1C-L (\$A004) or writing T1L-H (\$A005) or \$A007
7	Any IFR bit set with its corresponding IER bit also set	Clearing IFR0-IFR6 (\$A00D) or IER0-IER6 (\$A00E)

R6522 INTERRUPT ENABLE REGISTER (IER)



INTERRUPT ENABLE BITS (IER#4)

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.

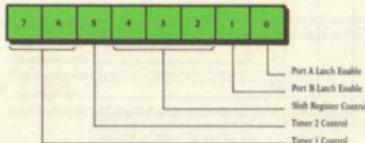
Note: IER7 is active only when R/W = L; when R/W = H, IER7 will read logic 1.

USER R 6522 VERSATILE INTERFACE ADAPTER (VIA)

R6522 M MEMORY ASSIGNEMENTS

Location	Function	
A000	Port B Output Data Register (ORB)	
A001	Port A Output Data Register (ORA)	
A002	Port B Data Direction Register (DDRB)	Controls handshake
A003	Port A Data Direction Register (DDRA)	0 = Input 1 = Output
Timer	R/W = L	R/W = H
A004	T1 Write T1L-L	Read T1C-L Clear T1 Interrupt Flag
A005	T1 Write T1L-H & T1C-H	Read T1C-H
	T1L-L → T1C-L	
	Clear T1 Interrupt Flag	
A006	T1 Write T1L-L	Read T1L-L
A007	T1 Write T1L-H	Read T1L-H
A008	T2 Write T2L-L	Read T2C-L
	T2L-L → T2C-L	Clear T2 Interrupt Flag
A009	T2 Write T2C-H	Read T2C-H
	T2L-L → T2C-L	Clear T2 Interrupt Flag
A00A	Shift Register (SR)	
A00B	Auxiliary Control Register (ACR)	
A00C	Peripheral Control Register (PCR)	
A00D	Interrupt Flag Register (IFR)	
A00E	Interrupt Enable Register (IER)	
A00F	Port A Output Data Register (ORA)	No effect on handshake

R6522 AUXILIARY CONTROL REGISTER (ACR)



PORT A LATCH ENABLE

ACR0 = 1 - Port A latch is enabled to latch input data when CA1 Interrupt Flag (IFR1) is set.
= 0 - Port A latch is disabled, reflects current data on PA pins.

PORT B LATCH ENABLE

ACR1 = 1 - Port B latch is enabled to latch the voltage on the pins for the input lines or the ORB contents for the output lines when CB1 Interrupt Flag (IFR4) is set.
= 0 - Port B latch is disabled, reflects current data on PB pins.

SHIFT REGISTER CONTROL

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of $\varnothing 2$.
0	1	1	Shift in under control of external clock.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of $\varnothing 2$.
1	1	1	Shift out under control of external clock.

TIMER 2 CONTROL

ACR5 = 0 - T2 acts as an interval timer in the one-shot mode.
= 1 - T2 counts a predetermined number of pulses on PB6.

TIMER 1 CONTROL

ACR7	ACR6	Mode
0	0	T1 one shot mode - Generate a single time-out interrupt each time T1 is loaded. Output to PB7 disabled.
0	1	T1 free-running mode - Generate continuous interrupts. Output to PB7 disabled.
1	0	T1 one shot mode - Generate a single time-out interrupt and an output pulse on PB7 each time T1 is loaded.
1	1	T1 free-running mode - Generate continuous interrupts and a square-wave output on PB7.

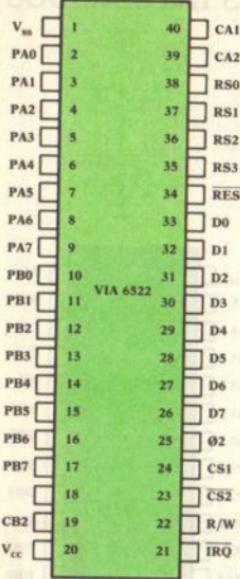
APÉNDICE G

PATILLAJE DE CIRCUITOS INTEGRADOS

V _{SS}	1	40	A6
RDY	2	39	Ω ₂
Ω ₁ (OUT)	3	38	CS1
IRQ	4	37	CS2
N.C.	5	36	RS
NMI	6	35	R/W
SYNC	7	34	RES
VCC	8	33	DB0
AB0	9	32	DB1
AB1	10	31	DB2
AB2	11	30	DB3
AB3	12	29	DB4
AB4	13	28	DB5
AB5	14	27	DB6
AB6	15	26	DB7
AB7	16	25	AB15
AB8	17	24	AB14
AB9	18	23	AB13
AB10	19	22	AB12
AB11	20	21	V _{SS}
μP 6502			
V _{SS}	1	40	A6
A5	2	39	Ω ₂
A4	3	38	CS1
A3	4	37	CS2
A2	5	36	RS
A1	6	35	R/W
A0	7	34	RES
PA0	8	33	DB0
PA1	9	32	DB1
PA2	10	31	DB2
PA3	11	30	DB3
PA4	12	29	DB4
PA5	13	28	DB5
PA6	14	27	DB6
PA7	15	26	DB7
PB7	16	25	IRQ
PB6	17	24	PB0
PB5	18	23	PB1
PB4	19	22	PB2
V _{DD}	20	21	PB3

Conexiónado de μP 6502.

R6522

 $V_{cc} = +5\text{ V}$ $A_9 \dots A_1$ = líneas de direcciones $D_0 \dots D_7$ = líneas de datos

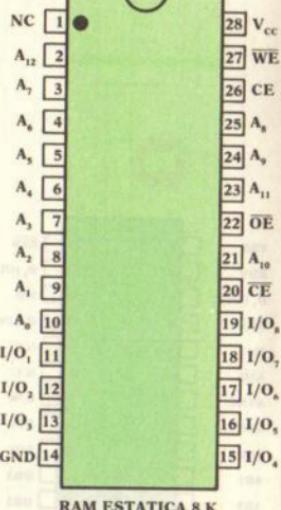
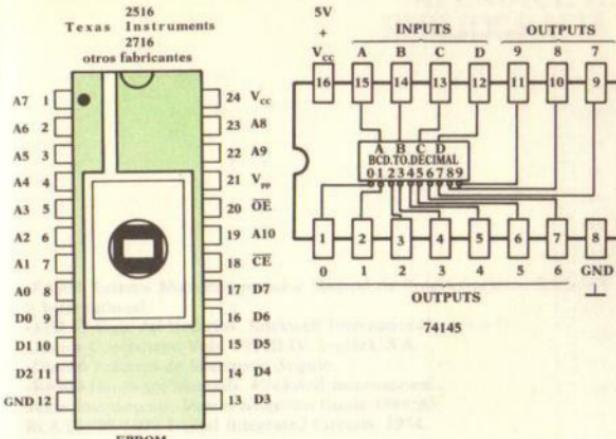
CS = Chip Select

CE = Chip Enable

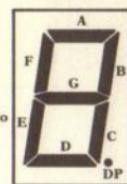
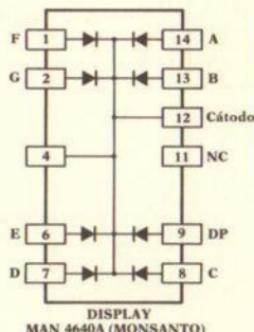
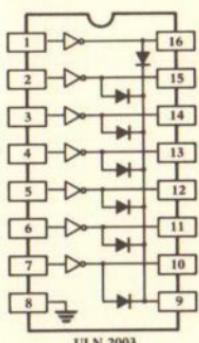
OE = Output Enable

PD = Power Down

5188

5564
5565Texas Instruments
2716
otros fabricantes

74145



APENDICE H. BIBLIOGRAFIA

«R6500 Sistema Microcomputador Manual de Programación». Rockwell International.

• «AIM 65 Guía del Usuario». Rockwell International.
• «Junior Computer». Vols. I-III-II-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.