

DISEÑO DE UN SISTEMA COMPUTADOR CON EL MICROPROCESADOR Z80

Junio de 2006

Autor: Ing. Alfredo Segura Celedón

El proyecto consiste en diseñar y armar un sistema computador utilizando el microprocesador Z80, memoria RAM, memoria ROM, Puertos Periféricos para entradas y salidas, reloj de tiempo real, interface serial RS-232, display LCD, display de segmentos de LED, teclado de Funciones y fuente de alimentación, que sea capaz de ejecutar programas introducidos por el usuario a través de comandos y datos hexadecimales correspondientes a los mnemónicos propios del microprocesador Z80, así como ejecutar subrutinas o subprogramas residentes en la ROM.

Con el teclado podrán efectuarse diversas Funciones, tales como:

- Introducir manualmente una dirección de memoria (2 bytes) y su correspondiente contenido (un byte)
- Ejecutar la función Intro o Enter
- Introducir manualmente una dirección de puerto de entrada o salida, así como capturar un dato o enviar un dato a dicho puerto.
- Avanzar o retroceder en las localidades de memoria de toda la RAM o la ROM, a través de dos botones con flecha adelante y atrás.
- Romper una secuencia de comandos o instrucciones a través de una tecla Break
- Introducir cualquiera de los caracteres alfanuméricos de la base hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.
- Ejecutar un programa introducido por el usuario o una rutina grabada en la ROM a través de la tecla GO.
- Resetear el computador mediante una tecla o mediante un comando.
- Provocar una interrupción no enmascarable NMI y ejecutar la acción llamada con su vector.
- Provocar una interrupción externa INT y ejecutar el programa al que apunta su vector.

En la siguiente figura se muestra un diagrama a bloques de todo el sistema computador con Z80.

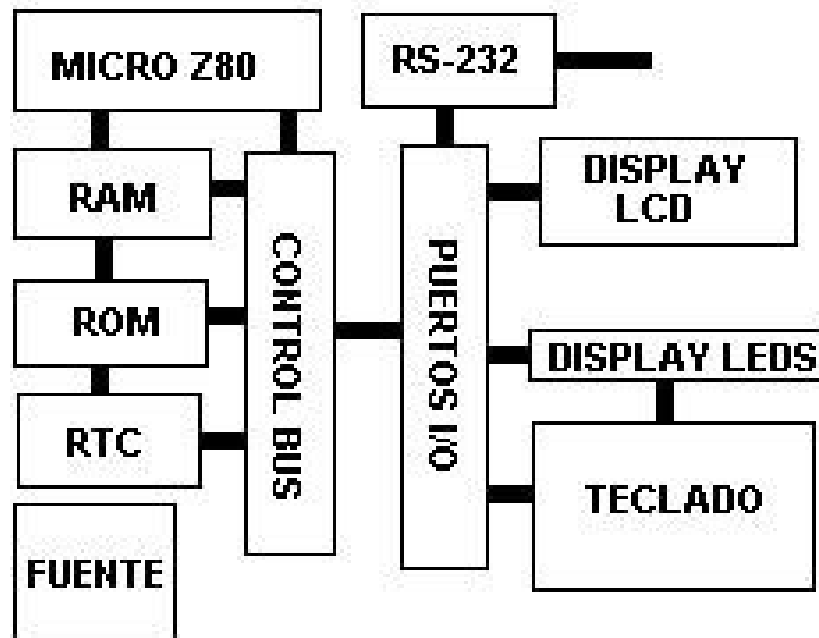
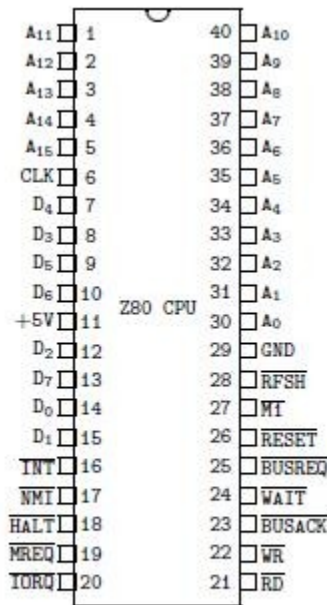


DIAGRAMA A BLOQUES DEL SISTEMA

PARTE I.

EL MICROPROCESADOR Z80

Primeramente se muestra una figura con los pines del microprocesador Z80 y en adelante se explicarán sus Funcionalidades, lo cual es básico para iniciar el diseño de todo periférico conectado al mismo y su interacción con el mundo externo y el usuario.



Microcontrolador Z80 CPU

PINS	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1-5, 30-40	A0 - 15	SALIDAS UNICAMENTE, ALTO ACTIVO, PROPORCIONAN LA DIRECCIÓN DE LOCALIDADES RAM, ROM Y PUERTOS DE ENTRADA O SALIDA, SELECCIONAN O HABILITAN MULTIPLEXORES O DECODIFICADORES DE MAPAS DE MEMORIA. CONSTITUYE EL BUS DE DIRECCIONES DE 16 BITS
6	CLK	ENTRADA DE PULSOS DE RELOJ, EN ESTE CASO DE 4 MHz, PROVENIENTES DE UN GENERADOR EXTERNO, DE ALTA ESTABILIDAD, PREFERENTEMENTE A CRISTAL DE CUARZO. UNA SOLA FASE.
7-10 12-15	D0 - D7	SALIDAS O ENTRADAS DE DATOS, NIVEL ACTIVO ALTO. ADMITEN O ENTREGAN LOS DATOS DESDE/A MEMORIA, PERIFERICOS, PUERTOS, RTC, Y CONSTITUYEN UN BUS COMPLETO DE 8 BITS.
11	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA EL MICROCONTROLADOR, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.
16	/INT	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE UN PERIFERICO ESTÁ SOLICITANDO UNA INTERRUPCIÓN, LA CUAL SERÁ IGNORADA O ATENDIDA POR EL MICROPROCESADOR DEPENDIENDO DEL SOFTWARE BAJO EL CUAL ESTÉ FUNCIONANDO.
17	/NMI	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE UN PERIFERICO ESTÁ EXIGIENDO UNA INTERRUPCIÓN NO ENMASCARABLE, LA CUAL ES ATENDIDA DE INMEDIATO POR EL MICROPROCESADOR, EJECUTANDO LA ACTIVIDAD QUE EL SOFTWARE HAYA DETERMINADO.
18	/HALT	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE DEBE DETENER TODA EJECUCIÓN DE SOFTWARE.
19	/MREQ	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE EL

		BUS DE DIRECCIONES ESTÁ APUNTANDO A UNA LOCALIDAD DE MEMORIA DETERMINADA Y ESTÁ LISTO PARA LEER O ESCRIBIR UN DATO EN DICHA LOCALIDAD. /MREQ SE USA PARA ACTIVAR LA LOGICA DECODIFICADORA DE MAPAS DE MEMORIA.
20	/IORQ	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE EL BUS DE DIRECCIONES ESTÁ APUNTANDO A UN PUERTO EN PARTICULAR, DE TAL MANERA QUE PUEDAN INGRESARSE O EXHIBIR DATOS HACIA O DESDE EL MISMO. /IORQ SE USA PARA ACTIVAR LA LOGICA DECODIFICADORA DE MAPAS DE PUERTOS .
21	/RD	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE LA INSTRUCCIÓN QUE SE ESTA EJECUTANDO ES DE LECTURA DE MEMORIA RAM O ROM O PERIFERICOS, DE TAL MANERA QUE PUEDAN INGRESARSE DATOS AL Z80 CPU. /RD SE USA PARA ACTIVAR LA LOGICA DECODIFICADORA DE MAPAS DE PUERTOS Y ACTIVAR DIRECTAMENTE A LAS MEMORIAS RAM Y ROM U OTROS DISPOSITIVOS PERIFERICOS EN SUS PINS CORRESPONDIENTES.
22	/WR	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE LA INSTRUCCIÓN QUE SE ESTA EJECUTANDO ES DE ESCRITURA DE MEMORIA RAM O ROM O PERIFERICOS, DE TAL MANERA QUE PUEDAN ENVIARSE DATOS A LOS MISMOS DESDE EL Z80 CPU. /WR SE USA PARA ACTIVAR LA LOGICA DECODIFICADORA DE MAPAS DE PUERTOS Y ACTIVAR DIRECTAMENTE A LAS MEMORIAS RAM Y OTROS DISPOSITIVOS PERIFERICOS EN SUS PINS CORRESPONDIENTES.
23	/BUSACK	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE EL BUS DE DIRECCIONES Y EL BUS DE DATOS ESTÁN LIBERADOS PARA QUE OTRO SISTEMA CONTROLADOR HAGA USO DE LOS MISMOS POR UN TIEMPO DETERMINADO, HASTA QUE LA SEÑAL /BUSREQ SEA LLEVADA NUEVAMENTE A NIVEL ALTO.
24	/WAIT	SALIDA ACTIVADA A NIVEL BAJO QUE INDICA QUE EL Z80 CPU HA INCRUSTADO PULSOS DE ESPERA SINCRONIZADOS CON CLK, PARA ESPERAR A AQUELLOS DISPOSITIVOS QUE CORREN MAS LENTOS QUE EL MICROPROCESADOR.
25	/BUSREQ	ENTRADA QUE SE ACTIVA A NIVEL BAJO QUE LE INDICA AL Z80 CPU QUE OTRO SISTEMA COMPUTADOR ESTÁ SOLICITANDO LA LIBERACIÓN DEL BUS DE DATOS Y BUS DE DIRECCIONES, PARA TOMAR CONTROL DE LOS MISMOS DURANTE UNA TRANSFERENCIA DE INFORMACIÓN. SU EFECTO CESA CUANDO /BUSREQ VA A NIVEL ALTO.
26	/RESET	ENTRADA QUE SE ACTIVA A NIVEL BAJO QUE LE INDICA AL Z80 CPU QUE DEBE DETENER TODA EJECUCIÓN DE INSTRUCCIONES Y RECOMENZAR SU OPERACIÓN DESDE 0000H, TAL Y COMO SE HACE CUANDO SE ENERGIZA POR PRIMERA VEZ EL SISTEMA.
27	/M1	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE SE ESTÁ EJECUTANDO UN CICLO M1 ACTUALMENTE.
28	/RFSH	SALIDA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA AL Z80 CPU QUE SE ESTÁ ENVIANDO UN PULSO QUE SE PUEDE USAR PARA REFRESCAR EL CONTENIDO DE MEMORIAS DINAMICAS, PARA QUE CONSERVAR LA INFORMACIÓN CONTENIDA EN TODAS SUS LOCALIDADES.

29	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
----	-----	---

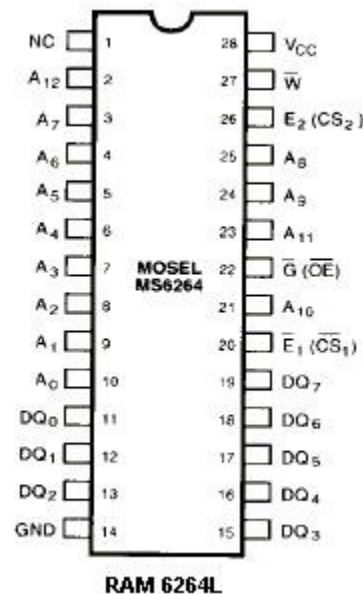
PARTE II

LA MEMORIA DE ACCESO ALEATORIO (RAM)

Aunque no es el alma propiamente del micro controlador, la RAM es muy importante en orden, ya que el primer lugar lo ocupa la memoria de solo lectura (ROM).

A la memoria RAM se puede acceder directamente con los pines de direcciones y de datos del Z80 CPU, pero hace falta controlar sus líneas de lectura (/RD), de escritura (/WR) e incluso de su habilitación de salida de datos (/OE), todo esto debidamente decodificado por mapas de memoria haciendo uso de las líneas de control del Z80 CPU. Así que primeramente daremos un vistazo a los pins de una memoria RAM y luego definiremos el mapeo de memoria correspondiente.

Memoria RAM MOSEL M6264L



Descripción de pins de la RAM MS6264L

Esta es una memoria RAM de 8K x 8 bits (8,192 bytes de 0000h a 2000h) ideal para este proyecto de baja escala.

PINS	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1	NC	PIN NO CONECTADO. SE RECOMIENDA NO CONECTAR NADA AQUÍ.
2-10,	A0 - A12	ENTRADAS QUE SE ACTIVAN A NIVEL ALTO Y QUE INDICAN A LA RAM

21, 23-25		QUE DEBE ACCEDER A UNA LOCALIDAD DE MEMORIA EN PARTICULAR, YA SEA PARA GRABAR UN DATO O PARA LEERLO.
11-13, 15-19	D0 – D7	ENTRADAS O SALIDAS QUE SE ACTIVAN A NIVEL ALTO Y QUE EXHIBEN LOS DATOS EXTRAIDOS DE LA RAM O PROVENIENTES DEL BUS DE DATOS PARA SU ALMACENAMIENTO.
14	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINADA TIERRA O MASA.
20	/E1 (/CS1)	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA A LA RAM QUE DEBE HABILITARSE PARCIALMENTE PARA OPERACIONES DE LECTURA O ESCRITURA EN SUS LOCALIDADES INTERNAS.
22	/G (/OE)	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE LE INDICA A LA RAM QUE DEBE HABILITARSE PARCIALMENTE PARA EXHIBIR DATOS DE SALIDA EN EL BUS DE DATOS. DURANTE OPERACIONES DE ESCRITURA, /G DEBE LLEVARSE A NIVEL ALTO.
26	E (CS2)	ENTRADA QUE SE ACTIVA A NIVEL ALTO Y QUE LE INDICA A LA RAM QUE DEBE HABILITARSE PARCIALMENTE PARA OPERACIONES DE LECTURA O ESCRITURA EN SUS LOCALIDADES INTERNAS. LA CONJUNCIÓN DE /E1 Y DE E, HACEN QUE LA RAM SE HABILITE TOTALMENTE.
27	/W	ENTRADA QUE SI SE ACTIVA A NIVEL BAJO, LE INDICA A LA RAM QUE LA OPERACIÓN ACTUAL SERÁ DE ESCRITURA. SI SE ACTIVA A NIVEL ALTO, LA OPERACIÓN SERÁ DE LECTURA.
28	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA. PUEDE CONECTARSE UNA BATERIA DE LITIO DEBIDAMENTE POLARIZADA CON DIODOS PARA SEGUIR MANTENIENDO DATOS, SI SE DESEA, AÚN CUANDO SE APAGUE EL SISTEMA.

Debido a que el microprocesador Z80 CPU “despierta” apuntando a la dirección 0000h, el mapeo de memoria debe comenzar con la ROM, y en sus primeras 3 localidades deberán estar los datos:

0000h C3 00 01 JP 0100h

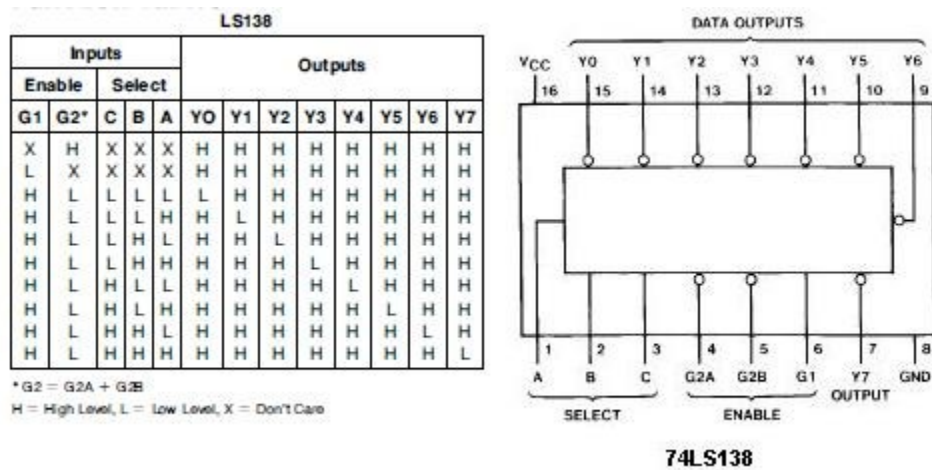
0003h FF FF FF ¿? ¿? ¿?

Es decir que el BIOS de este sistema micro controlador estará en la ROM desde la dirección 0000h hasta la 1FFFh, por lo tanto el mapeo de la memoria RAM arrancará en 2000h y terminará en 3FFFh.

Examinemos la siguiente tabla para determinar los rangos de accesos, teniendo como señales de control los pins A13, A14 y A15.

A15	A14	A13	A12-A0	INFERIOR	SUPERIOR
0	0	0	X	0000h	1FFFh
0	0	1	X	2000h	3FFFh
0	1	0	X	4000h	5FFFh
0	1	1	X	6000h	7FFFh
1	0	0	X	8000h	9FFFh
1	0	1	X	A000h	BFFFh
1	1	0	X	C000h	DFFFh
1	1	1	X	E000h	FFFFh

Para seleccionar adecuadamente a cada memoria, RAM y ROM en tiempos diferentes, es necesario echar mano de un decoder / demultiplexer tal como el 74LS138, cuya configuración se muestra a continuación:



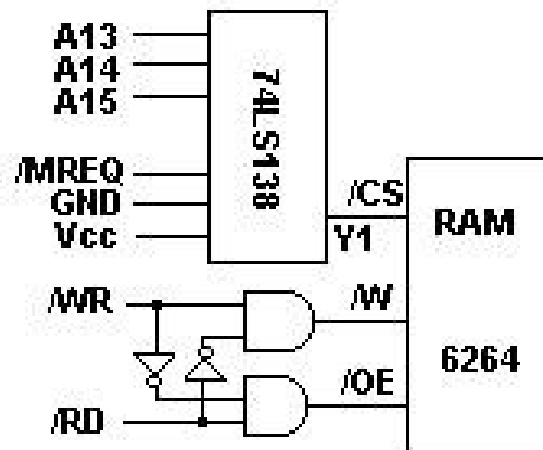
Usaremos A = A13, B = A14 y C = A15, G2A = /MREQ, G2B = GND, G1 = Vcc, y con esto decodificamos por Y0 salida para la ROM (de 0000h – 1FFFh) y Y1 salida para la RAM (de 2000h – 3FFFh).

Por otro lado para activar adecuadamente los pins /W y /OE de la RAM, usaremos:

$$/W = (/WR) \text{ AND } (/RD)$$

$$/OE = (/WR) \text{ AND } (/RD)$$

Circuito final de RAM y Decoder:



MAPEO RAM

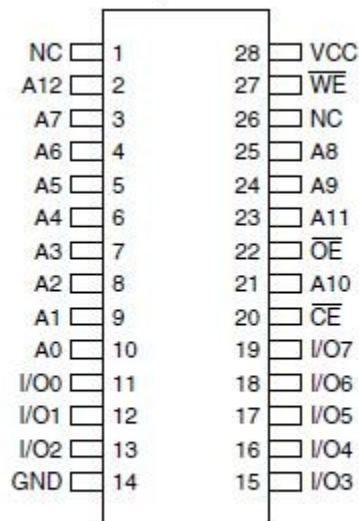
PARTE III

LA MEMORIA DE SOLO LECTURA (ROM)

Como se estableció en párrafos anteriores, la memoria ROM es el alma del sistema computador, mientras que el Z80 CPU es el cerebro.

Elegiremos la ROM AT28C64 que tiene una capacidad de 8,192 bytes, por lo que debe mapearse adecuadamente para acceder de la localidad 0000h a la 1FFFh. Tomaremos del mismo circuito anterior de la RAM, la señal Y0 del decoder 74LS138 para activar el pin /CS de la ROM.

Analicemos los pins de la memoria ROM elegida, cuya figura es la siguiente:

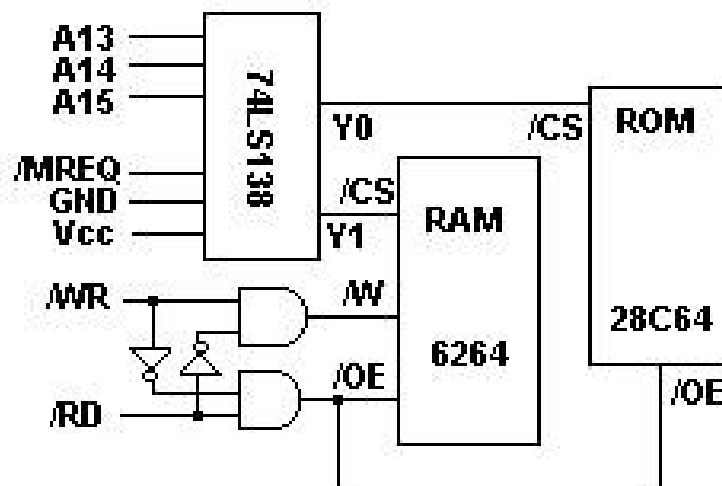


EPROM AT28C64

Descripción de pins de la EPROM (ROM) AT28C64:

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1,26	NC	NO CONEXIÓN. SE RECOMIENDA NO CONECTAR NADA A ESTE PIN
2 -10, 21, 23-25	A0 - A12	ENTRADAS QUE SE ACTIVAN A NIVEL ALTO Y QUE INDICAN A LA ROM QUE DEBE ACCEDER A UNA LOCALIDAD DE MEMORIA EN PARTICULAR, PARA LEER SU CONTENIDO.
11-13, 15-19	I/O0-7	SALIDAS QUE SE ACTIVAN A NIVEL ALTO Y QUE EXHIBEN LOS DATOS EXTRAIDOS DE LA ROM. EN ESTE DISPOSITIVO, POR TRATARSE DE EPROM ESTE BUS PUEDE SER DE ENTRADA, CUANDO SE GRABA INFORMACION QUE SERÁ SEMI-PERMANENTE, EL BIOS DEL SISTEMA.
14	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
20	/CE	ENTRADA QUE SE ACTIVA A NIVEL BAJO, SIGNIFICA CHIP ENABLE Y HABILITA TODA LA MEMORIA ROM PARA SER ACCESADA Y LEIDA.
22	/OE	ENTRADA QUE SE ACTIVA A NIVEL BAJO, UTILIZADA PARA ACTIVAR LAS SALIDAS DE DATOS LEIDOS. ES CONVENIENTE PONER /OE A NIVEL ALTO CUANDO NO SE ESTÁN REALIZANDO LECTURAS DE LA ROM.
27	/WE	ENRADA QUE SE ACTIVA A NIVEL BAJO Y HABILITA LA ESCRITURA DE LA EPROM (ROM) CUANDO SE PROGRAMA POR PRIMERA VEZ Y SUSBSECUENTES. SIN EMBARGO EN EL SISTEMA QUE SE ESTÁ DISEÑANDO PERMANECERÁ FIJA A NIVEL ALTO.
28	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.

Finalmente el circuito de la RAM estará ampliado para controlar la ROM, de la siguiente forma:



MAPEO DE RAM Y ROM

PARTE IV

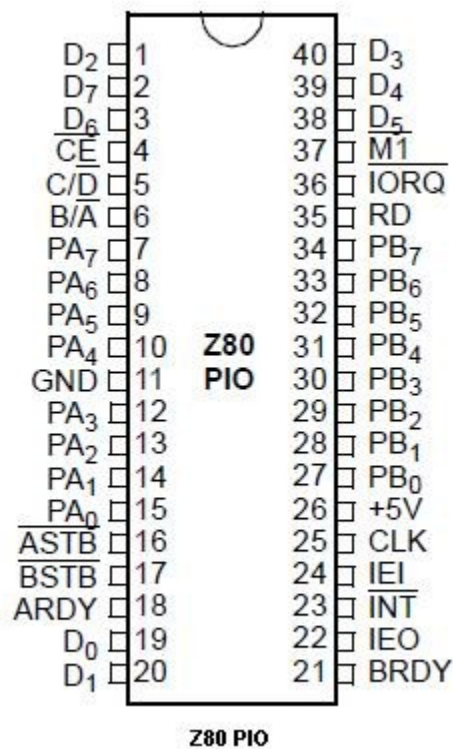
PUERTOS DE ENTRADA Y SALIDA

Hasta ahora la configuración del Z80 CPU con su RAM y ROM, aunque podrían Funcionar, realmente no proporcionan ninguna utilidad. Por lo tanto debemos proporcionarle canales de comunicación de datos que puedan ingresar o extraerse del sistema para darles un uso adecuado.

De esto se encargan los puertos, los cuales pueden ser de entrada o de salida. Los puertos pueden ser simples circuitos integrados TTL o CMOS, tales como el 74LS245, el 75LS373 o bien pueden usarse dispositivos más elaborados como son los PIO o PIA, los cuales pueden ser direccionados adecuadamente, programados sobre la marcha, como entradas o salidas, a veces con Funciones especiales en el manejo del bus de datos .

En este diseño usaremos el Z80-PIO, el 8255 PIA y el 74LS373 en diferentes actividades de entrada y salida del sistema computador.

La siguiente figura muestra los pines del Z80-PIO.



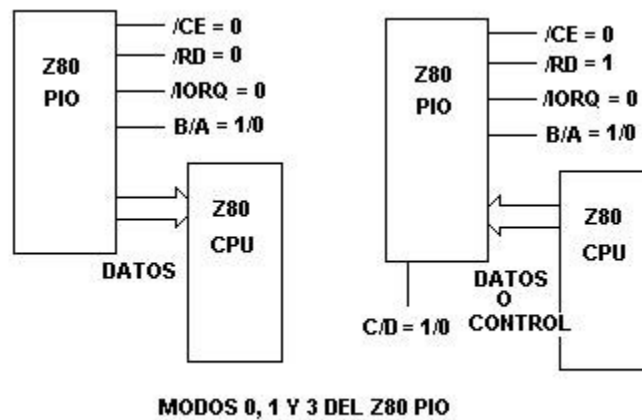
Descripción de pins del Z80 PIO.

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1-3, 19-20, 38-40	D0 – D7	ENTRADAS Y SALIDAS QUE SE ACTIVAN A NIVEL ALTO Y PUEDEN ENTRAR EN MODO TERCER ESTADO

4	/CE	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICA AL Z80 PIO QUE DEBE HABILITARSE PARA UNA OPERACIÓN DE ENTRADA O SALIDA DE DATOS O COMANDOS.
5	C-D	ENTRADA QUE PUEDE SER ACTIVA A NIVEL ALTO O BAJO. SI ES ALTO EL Z80 PIO SE DISPONE A RECIBIR UN COMANDO DESDE EL Z80 CPU. SI EL NIVEL ES BAJO, EL Z80 PIO RECIBIRÁ UN DATO. SE RECOMIENDA USAR A1 DE LA LINEA DE DIRECCIONES PARA CONTROLAR ESTE PIN.
6	B-A	ENTRADA QUE PUEDE SER ACTIVA A NIVEL ALTO O BAJO. SI ES ALTO EL Z80 PIO SELECCIONARÁ EL PUERTO B PARA OPERACIONES DE ENTRADA O SALIDA DE DATOS. SI EL NIVEL ES BAJO, SERÁ SELECCIONADO EL PUERTO A. SE RECOMIENDA USAR A0 DE LA LINEA DE DIRECCIONES PARA CONTROLAR ESTE PIN.
7-10, 12-15	PA0 – PA7	ENTRADAS Y SALIDAS QUE SE ACTIVAN A NIVEL ALTO Y PUEDEN ENTRAR EN MODO TERCER ESTADO. REPRESENTAN EL BUS DEL PUERTO A DEL Z80 PIO.
11	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
16	/ASTB	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y QUE PROVIENE DE UN PERIFERICO. DEPENDIENDO DEL MODO DE OPERACIÓN DEL Z80 PIO, ESTA SEÑAL REPRESENTA: MODO SALIDA: EL FILO POSITIVO DE ESTA SEÑAL DE ESTROBO ES EMITIDO POR EL PERIFERICO PARA RECONOCER QUE HA RECIBIDO LOS DATOS DISPONIBLES EN EL Z80 PIO. MODO ENTRADA: LA SEÑAL /ASTB DE NIVEL BAJO ES EMITIDA POR EL PERIFERICO PARA CARGAR DATOS DENTRO DEL REGISTRO DE ENTRADA DEL PUERTO A DEL Z80 PIO. MODO BIDIRECCIONAL: CUANDO /ASTB ESTA A NIVEL BAJO, LOS DATOS DESDE EL REGISTRO DE SALIDA DEL PUERTO A, SON PASADOS AL BUS DE DATOS BIDIRECCIONAL DEL MISMO PUERTO A. MODO DE CONTROL: LA SEÑAL QUE PUDIERA ESTAR EN /ASTB ES INHIBIDA INTERNAMENTE POR EL Z80 PIO.
17	/BSTB	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y PROVIENE DE UN PERIFERICO CONECTADO AL PUERTO B DEL Z80 PIO. FUNCIONA SIMILAR A /ASTB CON LA DIFERENCIA DE QUE EN EL MODO BIDIRECCIONAL DEL PUERTO A, ESTA SEÑAL PRESENTA LOS DATOS DEL PERIFERICO DENTRO DEL REGISTRO DE ENTRADA DEL PUERTO A.
18	ARDY	SALIDA QUE SE ACTIVA A NIVEL ALTO E INDICA QUE EL Z80 PIO TIENE LISTO SU REGISTRO A. EL SIGNIFICADO DE ESTA SEÑAL DEPENDE DEL MODO DE OPERACIÓN SELECCIONADO PARA EL PUERTO A, COMO SIGUE: MODO SALIDA: ARDY SE ACTIVA PARA INDICAR QUE EL REGISTRO DE SALIDA DEL PUERTO A, HA SIDO CARGADO Y EL BUS DE DATOS DEL PERIFERICO ES ESTABLE Y ESTA LISTO PARA TRANSFERIRLE DATOS. MODO ENTRADA: ARDY ES ACTIVA CUANDO EL REGISTRO DE ENTRADA DEL PUERTO A, ESTÁ VACÍO Y ESTÁ LISTO PARA ACEPTAR DATOS DESDE EL PERIFERICO CONECTADO AL PUERTO A. MODO BIDIRECCIONAL: ARDY ES ACTIVA CUANDO UN DATO ESTA DISPONIBLE EN EL REGISTRO DE SALIDA DEL PUERTO A PARA

		TRANSFERIRLO AL PERIFERICO. EN ESTE MODO EL DATO NO ES PUESTO EN EL BUS DE DATOS DEL PUERTO A, A MENOS QUE /ASTB SEA ACTIVO. MODO DE CONTROL: ARDY ES DESHABILITADA Y FORZADA A UN NIVEL BAJO.
21	BRDY	SALIDA QUE SE ACTIVA A NIVEL ALTO Y SU FUNCIONALIDAD ES SIMILAR A ARDY, CON LA DIFERENCIA QUE EN EL MODO BIDIRECCIONAL DEL PUERTO A, ESTA SEÑAL ES DE NIVEL ALTO CUANDO EL REGISTRO DE ENTRADA DEL PUERTO A ESTÁ VACÍO Y LISTO PARA ACEPTAR DATOS DESDE EL PERIFERICO.
22	IEO	SALIDA QUE SE ACTIVA A NIVEL ALTO E INDICA QUE EL Z80 PIO ESTA FORMANDO UNA CADENA DE PRIORIDADES. ESTA SALIDA SERÁ DE NIVEL ALTO SOLO SI IEI ES ALTO TAMBIEN Y EL Z80 CPU NO ESTÁ ATENDIENDO UNA INTERRUPCIÓN DESDE EL MISMO Z80 PIO. ENTONCES ESTA SEÑAL BLOQUEA LAS INTERRUPCIONES DE LOS DISPOSITIVOS PERIFERICOS DE MENOR PRIORIDAD, MIENTRAS OTRO CON UNA PRIORIDAD MAYOR ESTÉ SIENDO ATENDIDO.
23	/INT	SALIDA QUE SE ACTIVA A NIVEL BAJO, TIPO OPEN DRAIN E INDICA QUE EL Z80 PIO ESTÁ SOLICITANDO UNA INTERRUPCIÓN AL Z80 CPU.
24	IEI	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICA QUE EL Z80 PIO ESTA RECIBIENDO UNA SOLICITUD PARA FORMAR UNA CADENA DE PRIORIDADES, CUANDO VARIOS DISPOSITIVOS PERIFERICOS ESTÁN SIENDO USADOS. CUANDO IEI ESTA A NIVEL ALTO INDICA QUE NINGÚN OTRO DISPOSITIVO DE MAYOR PRIORIDAD ESTÁ SIENDO ATENDIDO POR EL Z80 CPU.
25	CLK	ENTRADA DE SEÑAL DE PULSOS DE RELOJ, QUE DEBE SER LA MISMA QUE UTILIZA EL Z80 CPU, EN NUESTRO CASO DE 4 MHz.
26	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.
27-34	PB0 – PB7	ENTRADAS Y SALIDAS QUE SE ACTIVAN A NIVEL ALTO Y PUEDEN ENTRAR EN MODO TERCER ESTADO. REPRESENTAN EL BUS DEL PUERTO B DEL Z80 PIO.
35	/RD	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y PROVIENE DEL Z80 CPU PARA INDICAR QUE SE LEERÁ UNO DE LOS PUERTOS DEL Z80 PIO, PASANDO LA INFORMACIÓN DESDE PUERTOS AL BUS DE DATOS DEL SISTEMA.
36	/IORQ	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICA AL Z80 PIO QUE ESTA EN PROCESO UNA ACCION DESDE EL Z80 CPU, TAL COMO LECTURA, ESCRITURA, DATOS O COMANDOS.
37	/M1	ENTRADA QUE SE ACTIVA A NIVEL BAJO QUE PROVIENE DEL Z80 CPU, E INDICA AL Z80 PIO QUE ES LA SEÑAL DE SINCRONIZACIÓN PARA VARIAS OPERACIONES. SI /M1 OCURRE SIN QUE /RD O /IORQ ESTÉN ACTIVAS, EL Z80 PIO ENTRA EN UN ESTADO DE RESET.

En las siguientes 2 figuras se muestran el Modo 0 y el Modo 1 para el Z80 PIO de nuestro diseño, lo que nos dará idea de las señales individuales y compuestas que deben llegar a sus entradas para un adecuado Funcionamiento.



Para la elección de números de puertos, seguiremos la base de que sean de 00h a FFh, más que suficientes para cualquier sistema computador pequeño.

Propondremos la creación del puerto 08h controlado por el Z80 PIO Puerto A y el 09h para el Puerto B. Por supuesto usaremos otro decoder / demultiplexer 74LS138 para obtener salidas que habiliten a dicho Z80 PIO en su pin /CE.

En la siguiente tabla se desglosa el diseño de su mapeo y configuración:

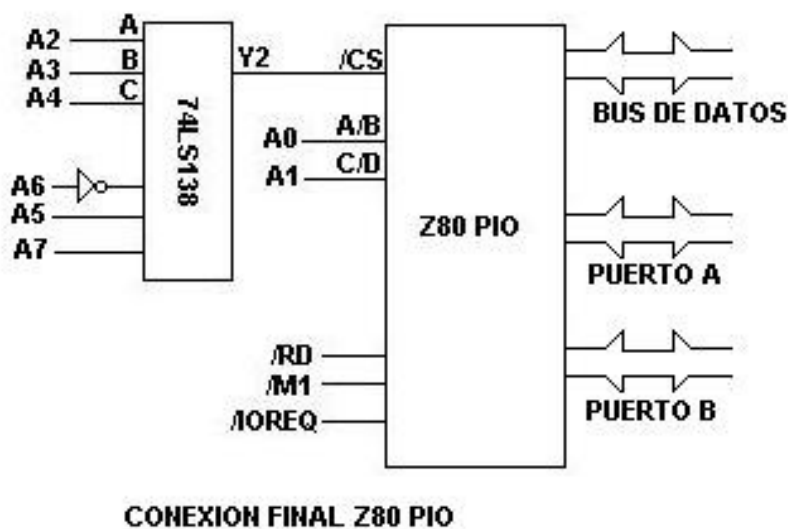
	74LS138						PIO		
74LS138	A7	A6	A5	A4	A3	A2	A1	A0	PUERTO
Y0	0	0	0	0	0	0	0	0	00h
Y0	0	0	0	0	0	0	0	1	01h
Y0	0	0	0	0	0	0	1	0	02h
Y0	0	0	0	0	0	0	1	1	03h
Y1	0	0	0	0	0	1	0	0	04h
Y1	0	0	0	0	0	1	0	1	05h
Y1	0	0	0	0	0	1	1	0	06h
Y1	0	0	0	0	0	1	1	1	07h
Y2	0	0	0	0	1	0	0	0	08h
Y2	0	0	0	0	1	0	0	1	09h
Y2	0	0	0	0	1	0	1	0	0Ah
Y2	0	0	0	0	1	0	1	1	0Bh
							C/D	B/A	

MAPEO DEL Z80 PIO

Dado que el 74LS138 tiene salidas hasta Y7, esta tabla crece, proporcionando los números de puertos correspondientes, para futuras expansiones.

Observamos que el decoder 74LS138 será controlado con las líneas de dirección A2, A3 y A4, en sus entradas A, B y C respectivamente, mientras que para habilitar los puertos 08h y 09h como se tiene planeado llevaremos Y2 al pin /CS del Z80 PIO. Para seleccionar el Puerto A o el Puerto B y datos o comandos, se llevarán las líneas del bus de direcciones A0 y A1 a los pines A/B y C/D respectivamente.

Finalmente las conexiones para el Z80 PIO y su correspondiente 74LS138 quedan como se indica en la siguiente figura:



El puerto 08h será usado como salida para mostrar en una barra de LEDS diferentes resultados del procesamiento del sistema computador en general, quedando a criterio del programador el uso que se le desee dar.

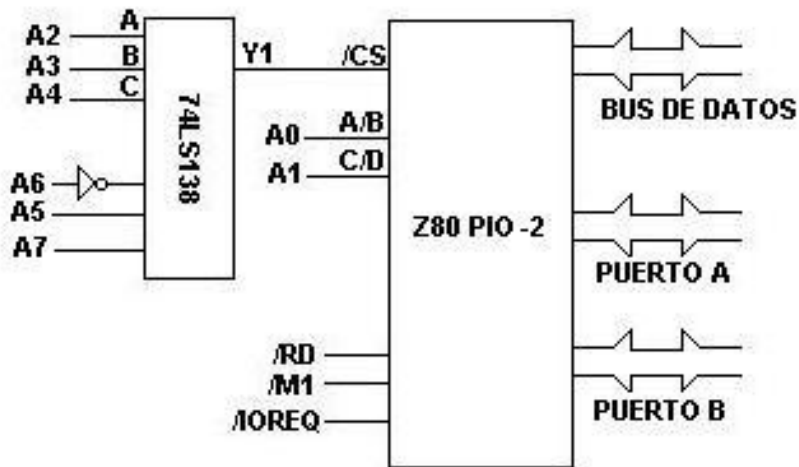
El puerto 09h será de entrada para obtener datos captados por la sección de comunicaciones seriales del sistema computador. Esta parte se diseñó por anticipado para tener acceso vía el puerto serie de una PC, para comunicarse rápidamente con el sistema computador y así ver los progresos de diseño, así como tener una manera a mano de introducir datos. En etapas posteriores se desarrollará el diseño completo de esta sección de comunicaciones seriales.

Hay que notar que de acuerdo a la tabla de mapeo del Z80 PIO, y bajo control de Y2 tenemos los puertos 0Ah y 0Bh, los cuales corresponden a Comandos-Puerto A y Comandos-Puerto B, respectivamente, que también serán usados en la creación del BIOS de este sistema computador, para definir la Funcionalidad completa del Z80 PIO.

UN SEGUNDO Z80 PIO EN EL SISTEMA

Adicionaremos un segundo Z80 PIO-2 a nuestro sistema computador y le asignaremos el rango de mapeo de 04h a 07h, y lo utilizaremos para realizar la interface con un teclado de matriz y un display de dígitos con segmentos de LEDS.

De la tabla de mapeo del Z80 PIO, simplemente tomamos la señal Y1 del 74LS138 y la llevamos a /CS del Z80 PIO-2 y tomamos nota que se seleccionará cuando A4-A3-A2 sea 0-0-1.



CONEXION FINAL Z80 PIO-2

PARTE V

DISPLAY DE SEGMENTOS DE LED Y TECLADO DE MATRIZ

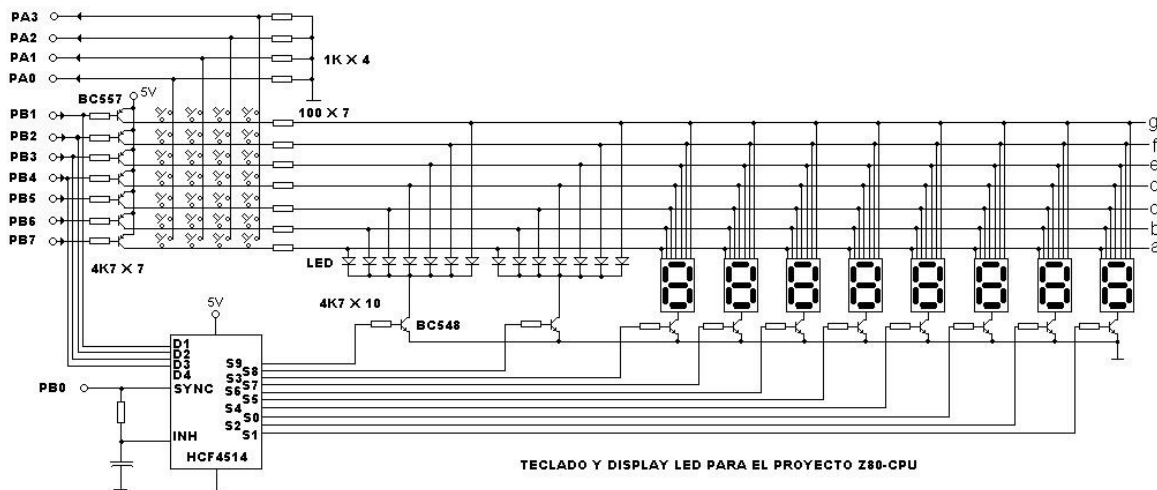
La interfaz principal de un sistema computador con el humano es lógicamente un teclado y un display. Tomaremos un modelo de display y teclado del libro Z80 Design de E. H Rony, el cual utiliza el MC4514B, el cual es un decodificador de 4 a 16 líneas con latch. Se utilizan esas 4 líneas de entrada de este decoder provenientes del Puerto B del Z80 PIO-2 (salidas), usando PB1 - PB4, mientras que PB0 se usa para activar el Estrobo y por consiguiente el Latch, dejando fija a la salida del decoder la selección efectuada. Por su parte PB1 - PB7 ingresarán a la matriz de teclas y al mismo tiempo a los segmentos comunes de los 7 dígitos del display de LEDS, en el orden g - a, respectivamente. El pulsar un botón del teclado matricial, se transferirá un voltaje a una de las salidas del teclado y dependiendo de cual botón se oprima se activará la columna correspondiente, 4 en total, para constituir salidas hacia el Puerto A del Z80 PIO-2.

La programación de este teclado matricial con display consistirá en elaborar una rutina que escanee constantemente este periférico, por una parte y en un primer tiempo activar de uno en

uno los pins PB1 – PB4 y luego PB0 para elegir consecutivamente las salidas del MC4514B. En cada activación de PB0, colocar en PB1 a PB7 los segmentos que correspondan al dígito seleccionado, por lo que encenderán los leds correspondientes, visualizándose para el usuario, ese dígito y así sucesivamente.

En un segundo tiempo, y como parte del escaneo de teclas se activarán uno a uno los pins PB1 a PB7 y desactivando PB0, por lo que si se oprime un botón, se produzca un dato en PA0 – PA3, que es el Puerto A del Z80 PIO-2.

En la siguiente figura se muestra el diagrama del teclado de matriz con display de segmentos de LEDS.



En este punto, el BIOS del proyecto Z80 ya debe tener avance, pues ya contamos con memoria RAM, memoria ROM, 2 interfaces de entrada y salida para introducir datos y mostrar resultados en un display. Al final de este documento, en el Apéndice 1, aparece todo el listado del archivo z801bij.lst completo, por lo que a medida que avancemos en la estructuración de este sistema computarizado, iremos poniendo tramos de dicho programa.

En el arranque del Z80 CPU, la primera dirección que aparece en el bus de direcciones del propio microprocesador, es la 0000h, donde debe existir un código C3 00 01 (JP 0100h):

```
0070 0000 .org 0000h ;rutina que se invoca con RST 00h --> C7
```

```
0071 0000 C3 00 01 jp 0100h ;o reseteando el sistema
```

```
.
```

```
.
```

```
0129 0100 org 0100h
```

0130	0100 31 F0 20	ld sp,20F0h	;define el area stack EN 20F0h
0131	0103 21 00 21	ld hl,2100h	
0132	0106 ED 56	im 1	
0133	0108 FB	ei	
0134	0109 01 8F 8B	ld bc,8b8fh	;PIA-1 Puertos A,B,C entradas Modo 0
0135	010C ED 41	out (c),b	
0136	010E 01 93 8B	ld bc,8b93h	;PIA-2 Puertos A,B,C entradas Modo 0
0137	0111 ED 41	out (c),b	
0138	0113 01 07 0F	ld bc,0f07h	;PIO1-B salidas
0139	0116 ED 41	out (c),b	
0140	0118 01 06 4F	ld bc,4f06h	;PIO1-A entradas
0141	011B ED 41	out (c),b	
0142	011D 01 0B 4F	ld bc,4f0Bh	;PIO2-B entradas
0143	0120 ED 41	out (c),b	
0144	0122 01 0A 0F	ld bc,0f0Ah	;PIO2-A salidas
0145	0125 ED 41	out (c),b	
0146	0127 01 09 00	ld bc,0009h	;saca 00h en PIO2-B
0147	012A ED 41	out (c),b	
0148	012C 01 08 00	ld bc,0008h	;saca 00h en PIO2-A
0149	012F ED 41	out (c),b	
0150	0131 3E 00	ld a,00h	;banderas de leds apagadas
0151	0133 32 10 20	ld (ledsl),a	
0152	0136 CD 30 0F	call flags	;rutina que sensa los flags y los
0153	0139 00	nop	;guarda en ledsl y ledsh

En este tramo del código el BIOS, se notan los preparativos iniciales del sistema computador que el Z80 CPU va a controlar de ahora en adelante.

Línea 0071 es un salto a la línea 0100h

Línea 0130 Se define el inicio del Stack, es decir el área de memoria RAM que será utilizada por el Z80 CPU para ir almacenando temporalmente (PUSH) el contenido de los registros internos del CPU, cuando se hacen saltos, llamadas a rutinas, que en algún momento deben regresar al punto de llamada y en ese momento se rescatan (POP) en el orden inverso al que fueron almacenados, para restaurar las condiciones iniciales antes del salto. Así que este espacio de memoria se accede “hacia abajo” en la RAM, siendo la primera localidad la 20F0h, la segunda la 20EFh, la tercera sería la 20EEh, y así sucesivamente hasta la 2000h. A este modo de almacenamiento se le denomina FILO, es decir, primero en entrar-último en salir, lo cual quiere decir que si hacemos las siguientes instrucciones:

PUSH AF

PUSH DE

PUSH HL

Entonces sucede lo siguiente: Los registros AF se introducen en las localidades 20F0h y 20EFh. Cuando los registros DE se ingresan al STACK, A y F se desplazan a las 20EEh y 20EDh, mientras que D y E ocupan las localidades 20F0h y 20EFh. Finalmente cuando HL se ingresan al STACK, A y F se desplazan a las localidades 20ECh y 20EBh, D y se desplazan a los registros 20EDh y 20EEh y finalmente H y L se posicionan en 20F0h y 20EFh. Todo este proceso fue un efecto PUSH, es decir “empujar” dentro del STACK.

Veamos ahora las siguientes instrucciones:

POP HL

POP DE

POP AF

Se extraen H y L de las localidades 20F0h y 20EFh, mientras que D y L saltan hacia arriba a las localidades 20F0h y 20EFh y A y F saltan a las 20EEh y 20EDh. En seguida se extraen D y E, por lo que A y F saltan hacia arriba las localidades 20F0h y 20EFh. Finalmente se extraen A y F, por lo que el STACK queda vacío. Todo este proceso fue un efecto POP, es decir “jalar” hacia afuera del STACK.

Todo programador sabe éste proceso de sobra, sin embargo se pone en este documento para darnos una idea si el STACK es suficiente o no, al momento de elegirlo, pues debe ser un área intocable ya que de otra manera, el sistema se trabaría constantemente, dependiendo de la complejidad del software introducido en un área permitida.

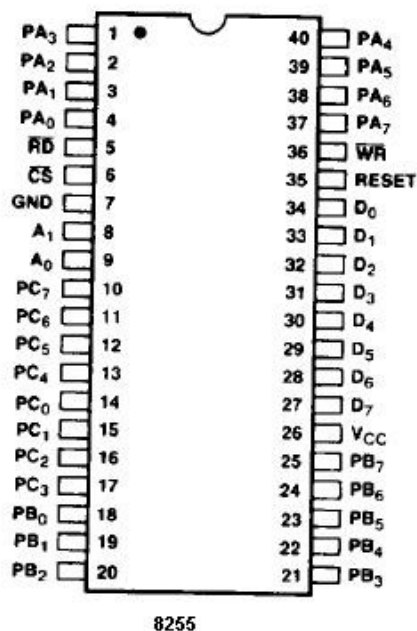
PARTE VI

MÁS PUERTOS DE ENTRADA Y SALIDA

Los ingenieros nos lamentamos de que muchos sistemas de cómputo tienen limitantes de puertos para introducir y obtener datos ya procesados, por lo que este diseño no quedará escaso de puertos, sino que haremos lo posible por adicionarle otros 4, para lo cual dedicaremos esta parte del documento.

Para variar, usaremos el circuito integrado 8255 denominado Programmable Peripheral Interface o PPI. También es posible utilizar el R6520 PIA (Peripheral Interface Adapter), que puede estar descontinuado y difícil de conseguir, por lo que usaremos el primero mencionado.

La siguiente figura muestra la configuración de pines del 8255.



Recorrido por los pines de este circuito integrado:

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1-4, 37-40	PA0 – PA7	8 BITS PARA SALIDAS O ENTRADAS A O DESDE PERIFÉRICOS. SI SON SALIDAS, SON CON BUFFER Y CON LATCH. SI SON ENTRADAS, SON CON LATCH.
5	/RD	ENTRADA QUE SE ACTIVA A NIVEL BAJO Y LE INDICA AL PPI QUE DEBE

		HABILITARSE PARA ENVIARLE AL Z80 CPU DATOS O INFORMACION DE ESTADO, VIA EL BUS DE DATOS DEL SISTEMA.
6	/CS	ENTRAD QUE SE ACTIVA A NIVEL BAJO Y LE INDICA AL PPI QUE ESTA DISPONIBLE PAA INICIAR LA COMUNICACIÓN CON EL Z80 CPU.
7	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
8	A1	ENTRADA QUE SE ACTIVA A NIVEL ALTO PARA SELECCIONAR UN PUERTO ESPECIFICO DEL PPI, EN CONJUNCION CON LAS ENTRADAS /RD Y /WR, CONTROLA LA SELECCIÓN DE UNO DE LOS 3 PUERTOS DEL PPI O SUS REGISTROS DE LA PALABRA DE CONTROL. SE RECOMIENDA CONECTARLO AL BUS DE DIRECCIONES DEL Z80 CPU, PIN A1.
9	A0	ENTRADA QUE SE ACTIVA A NIVEL ALTO PARA SELECCIONAR UN PUERTO ESPECIFICO DEL PPI, EN CONJUNCION CON LAS ENTRADAS /RD Y /WR, CONTROLA LA SELECCIÓN DE UNO DE LOS 3 PUERTOS DEL PPI O SUS REGISTROS DE LA PALABRA DE CONTROL. SE RECOMIENDA CONECTARLO AL BUS DE DIRECCIONES DEL Z80 CPU, PIN A0.
10-17	PC0 - PC7	8 BITS PARA SALIDAS O ENTRADAS A O DESDE PERIFÉRICOS. SI SON SALIDAS, SON CON BUFFER Y CON LATCH. LAS ENTRADAS NO SON CON LATCH. ESTE PUERTO PUEDE DIVIDIRSE EN 2 PUERTOS DE 4 BITS BAJO EL MODO DE CONTROL. CADA CANAL DE 4 BITS CONTINENE UN LATCH DE 4 BITS Y PUEDE USARSE PARA CONTROLAR LAS SEÑALES DE SALID A Y SEÑAL DE ESTADO DE ENTRADA EN CONJUNTO CON LOS PUERTOS A Y B.
18-25	PB0 -PB7	8 BITS PARA SALIDAS O ENTRADAS A O DESDE PERIFÉRICOS. SI SON SALIDAS, SON CON BUFFER Y CON LATCH. SI SON ENTRADAS, SON CON LATCH.
26	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.
27-34	D0 - D7	ESTOS PINES DE ENTRADA, SALIDA O TERCER ESTADO SE CONECTAN AL BUS DE DATOS DEL Z80 CPU PARA LA TRANSFERENCIA DE INFORMACION DESDE O A LOS PERIFERICOS CONECTADOS AL PPI. LAS PALABRAS DE CONTROL E INFORMACION DE ESTADO TAMBIEN SON TRANSFERIDAS A TRAVES DE ESTOS PINES.
35	RESET	ENTRADA ACTIVA A NIVEL ALTO E INDICA AL PPI QUE DEBE ENTRAR EN UN ESTADO DE REINICIO, BORRANDO LOS REGISTROS DE CONTROL DE TODOS LOS PUERTOS (A, B Y C),LOS CUALES SON AJUSTADOS EN EL MODO DE ENTRADA.
36	/WR	ENTRADA ACTIVA A NIVEL BAJO E INDICA AL PPI QUE EL Z80 CPU ESCRIBIRA DATOS O PALABRAS DE CONTROL DENTRO DEL MISMO.

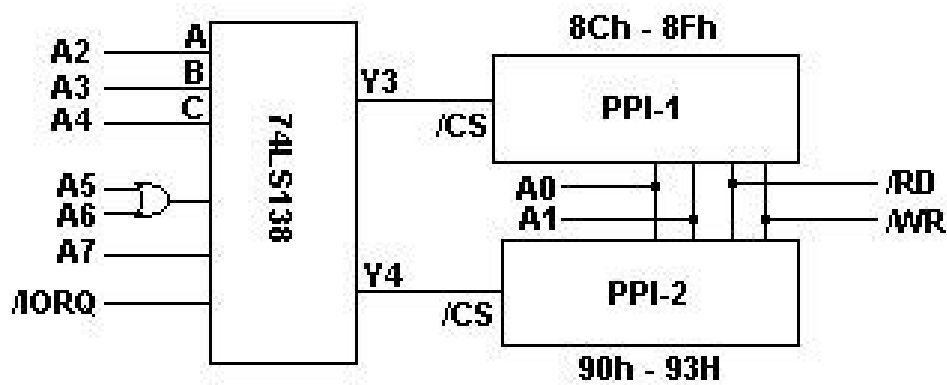
Incorporaremos 2 unidades PPI a este proyecto, el primero para controlar un display LCD de 16x2 líneas por el Puerto A, quedando configurados y en reserva los puertos B y C. El segundo PPI será destinado a controlar un motor stepper, por lo que se usarán solamente 4 bits de su Puerto A, quedando en reserva los puertos B y C, así como los otros 4 bits del Puerto A.

El primer PPI tendrá el rango de direcciones 8Ch a 8Fh y el segundo PPI, el rango de 90h a 93h, direcciones que deberán ser mapeadas en forma similar a como se hizo con los PIO, alcanzando cada dirección de puerto con la conjugación de las señales A0 y A1.

Entra en juego un nuevo decoder / demultiplexer 74LS138 para estos PPI y otros puertos adicionales que se incorporarán más adelante. Veamos primero un mapa de direccionamiento que puede manejar puertos arriba de 80h.

	74LS138						PPI		
74LS138	A7	A6	A5	A4	A3	A2	A1	A0	PUERTO
Y0	1	0	0	0	0	0	X	X	80h-83h
Y1	1	0	0	0	0	1	X	X	84h-87h
Y2	1	0	0	0	1	0	X	X	88h-8Bh
Y3	1	0	0	0	1	1	X	X	8Ch-8Fh
Y4	1	0	0	1	0	0	X	X	90h-93h
Y5	1	0	0	1	0	1	X	X	94h-97h
Y6	1	0	0	1	1	0	X	X	98h-9Bh
Y7	1	0	0	1	1	1	X	X	9Ch-9Fh
							A1	A0	

MAPEO PARA 8255 PPI



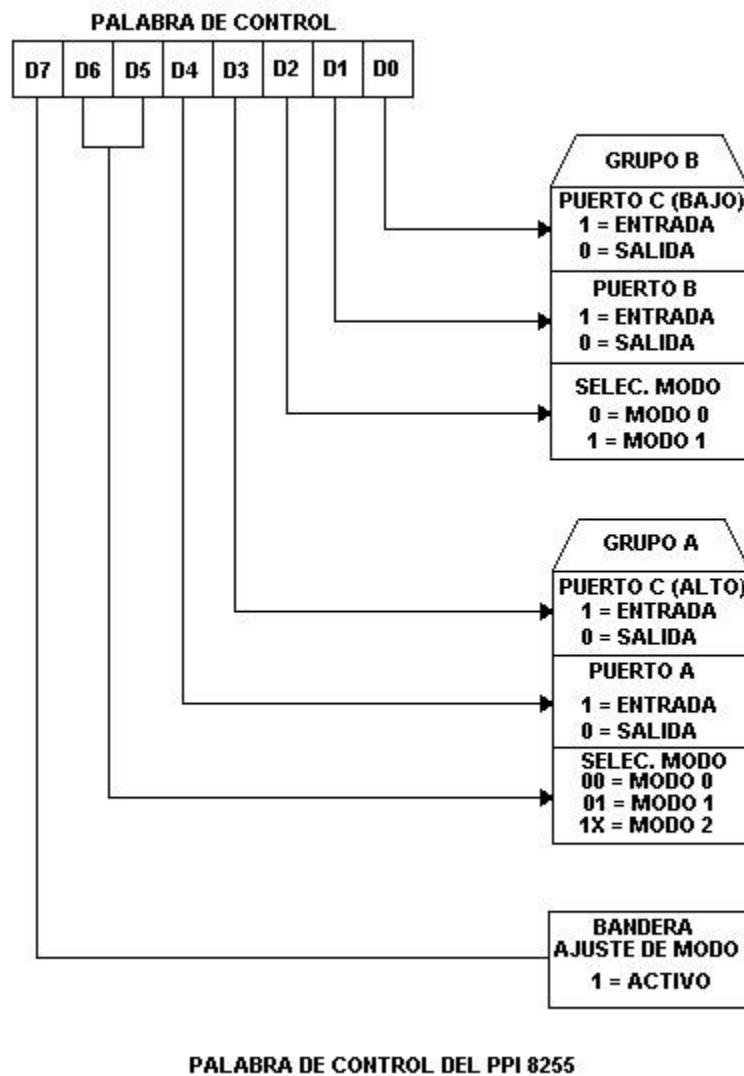
CONEXION FINAL 8255 PPI-1 PP-2

El 8255 PPI tiene tres registros de datos y un registro de control, que se seleccionan con las líneas A0 y A1, de la siguiente manera:

PUERTOS	A1	A0	/RD	/WR	/CS	OPERACIONES DE ENTRADA (READ)
8C, 90	0	0	0	1	0	PUERTO A → DATA BUS
8D, 91	0	1	0	1	0	PUERTO B → DATA BUS

8E, 92	1	0	0	1	0	PUERTO C → DATA BUS
						OPERACIONES DE SALIDA (WRITE)
8C, 90	0	0	1	0	0	DATA BUS → PUERTO A
8D, 91	0	1	1	0	0	DATA BUS → PUERTO B
8E, 92	1	0	1	0	0	DATA BUS → PUERTO C
8F, 93	1	1	1	0	0	DATA BUS → CONTROL
						FUNCION DESHABILITADA
	X	X	X	X	1	DATA BUS → TRI ESTADO
	1	1	0	1	0	CONDICION ILEGAL
	X	X	1	1	0	DATA BIS → TRI ESTADO

Palabras de configuración del PPI 8255:



DISPLAY LCD 16X2

Se introduce aquí el tema del display LCD ya que en el proyecto está conectado a uno de los PPI 8255, y en ese caso, esta interface se mantiene exclusiva para dicho display.

El display LCD 16x2 consta de 2 líneas horizontales de 16 caracteres cada una y consta de la circuitería propia para decodificar los datos que ingresan por sus terminales. Consta de 16 terminales, las cuales en orden son las siguientes:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
GND	VCC	VEE	RS	R/ W	E	D0	D1	D2	D3	D4	D5	D6	D7	LED-	LED +

En algunos diseños del LCD 16x2 únicamente se cuenta con los primeros 14 pines, porque no llevan un LED de iluminación de la carátula incluido.

Se alimenta con 5VCD y la terminal VEE, se puede ajustar dentro ese rango para controlar el contraste. Sin embargo si se conecta a GND, se simplifica el diseño de hardware y el display queda con un contraste muy aceptable.

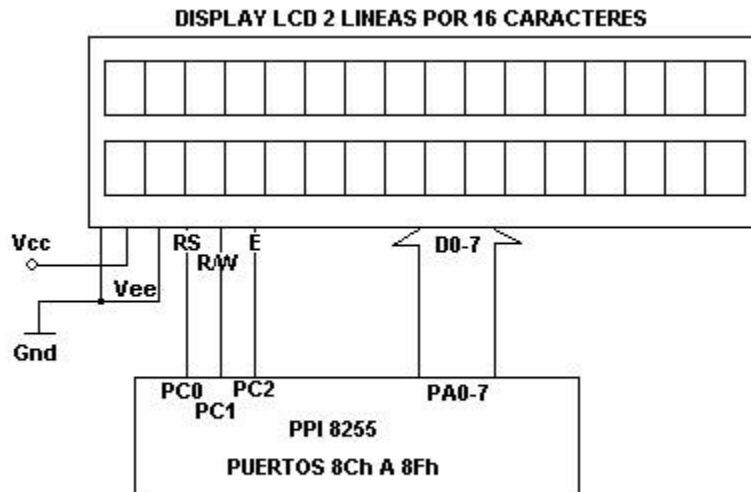
Aunque el LCD 16x2 que usamos en este proyecto tiene la capacidad de funcionar con los 8 bits de datos y también con solo 4 de ellos (D4-D7), se usará el estilo de 8 bits.

Las señales de control RS, R/W y E, cuyos nombres respectivamente son: Register Selection, Read/Write y Enable, juegan importante papel en la interface controlada por software de este proyecto. Aunque R/W no juega un papel definitivo, porque siempre escribiremos sobre el display, y W será 0 siempre, lo hemos dejado cableado con el PPI 8255 al pin 5 del display.

RS: SELECCIÓN DE REGISTRO	R/W: LECTURA/ESCRITURA	E: ACTIVACION
0 = CONTROL	0 = ESCRITURA	0 = DESACTIVADO
1 = DATOS	1 = LECTURA	1 = ACTIVADO

Se usa el puerto A (PA0-PA7) del PPI 8255 para los 8 bits de datos hacia el LCD y el puerto C, únicamente con sus pines PC0-PC2, para las señales de control del LCD, quedando libre y disponible el puerto B completo, así como los pines PC3-PC7.

En la siguiente figura se muestra la conexión entre el LCD 16x2 y el PPI 8255:



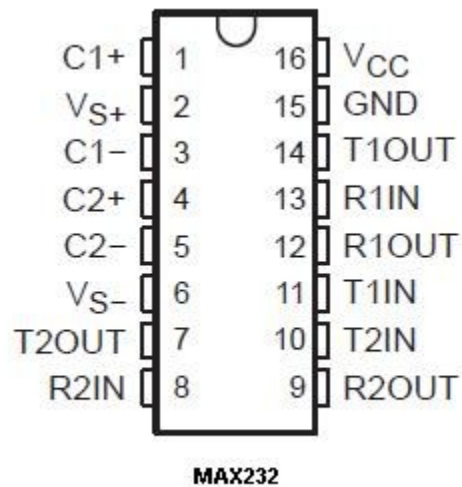
CONEXION FINAL DEL PPI 8255 Y DISPLAY LCD 16X2

PARTE VII

PUERTO DE COMUNICACIONES SERIALES

Tal como se mencionó en apartados anteriores, este circuito computador estará dotado de un puerto de comunicaciones seriales tipo RS-232, capaz de comunicarse con una PC con 2 hilos únicamente, Tx, Rx y por supuesto GND.

El puerto hardware se implementará con un circuito integrado MAX232, del cual vemos la configuración de sus pines en la siguiente figura:



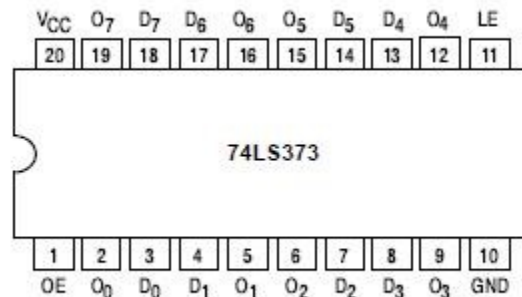
Descripción de los pines del MAX232.

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1	C1+	EN ESTE PIN SE CONECTA EL BORNE POSITIVO DE UN CAPACITOR ELECTROLITICO CON VALOR DE 1 A 10 Uf.
2	Vs+	EN ESTE PIN APARECERÁ UN VOLTAJE POSITIVO MAYOR DE 8 VCD, USADO EN AJUSTAR EL NIVEL TTL A +12 V DE LA NORMA RS-232. Y DEBE CONECTARSE EL BORNE POSITIVO DE UN SEGUNDO CAPACITOR Y SU OTRO BORNE A GND, CON VALOR DE 4.7 A 10Uf.
3	C1-	EN ESTE PIN SE CONECTA EL BORNE NEGATIVO DEL CAPACITOR USADO EN EL PIN 1, CON UN VALOR DE 1 A 10 Uf.
4	C2+	EN ESTE PIN SE CONECTA EL BORNE POSITIVO DE UN TERCER CAPACITOR ELECTROLITICO CON VALOR DE 1 A 10Uf.
5	C2-	EN ESTE PIN SE CONECTA EL BORNE NEGATIVO DEL TERCER CAPACITOR.
6	Vs-	EN ESTE PIN APARECERÁ UN VOLTAJE NEGATIVO MAYOR DE 8 VCD, USADO EN AJUSTAR EL NIVEL TTL A -12 v DE LA NORMA RS-232 Y DEBE CONECTARSE EL BORNE NEGATIVO DE UN CUARTO CAPACITOR Y SU OTRO BORNE A GND, CON VALOR DE 4.7 A 10Uf.
7	T2OUT	POR ESTE PIN SALE LA SEÑAL Tx HACIA LA PC. POR LO TANTO DEBE CONECTARSE A Rx DEL PUERTO SERIAL DE LA PC.
8	R2IN	POR ESTE PIN INGRESA LA SEÑAL Tx DESDE LA PC. POR LO TANTO DEBE CONECTARSE A Tx DEL PUERTO SERIAL DE LA PC.
9	R2OUT	POR ESTE PIN SALE EN NIVEL TTL, LA SEÑAL Rx YA AJUSTADA QUE INGRESÓ POR R2IN. SU NIVEL DE 0 A 5 VOLTS PERMITE USARLA EN EL SISTEMA COMPUTADOR.
10	T2IN	POR ESTE PIN INGRESA LA SEÑAL TX A NIVEL TTL PROVENIENTE DEL SISTEMA COMPUTADOR DE 0 A 5 VOLTS, PARA AJUSTARSE A NIVEL RS-232 Y QUE SALDRÁ POR EL PIN T2OUT.
11	T1IN	POR ESTE PIN INGRESA OTRA SEÑAL TX A NIVEL TTL PROVENIENTE DEL SISTEMA COMPUTADOR DE 0 A 5 VOLTS, PARA AJUSTARSE A NIVEL RS-232 Y QUE SALDRÁ POR EL PIN T2OUT. (EN ESTE DISEÑO NO SE USARÁ).
12	R1OUT	POR ESTE PIN SALE EN NIVEL TTL, OTRA SEÑAL Rx YA AJUSTADA QUE INGRESÓ POR R2IN. SU NIVEL DE 0 A 5 VOLTS PERMITE USARLA EN EL SISTEMA COMPUTADOR. (EN ESTE DISEÑO NO SE USARÁ).
13	R1IN	POR ESTE PIN INGRESA OTRA SEÑAL Tx DESDE LA PC. POR LO TANTO DEBE CONECTARSE A Tx DEL PUERTO SERIAL DE LA PC. (EN ESTE DISEÑO NO SE USARÁ).
14	T1OUT	POR ESTE PIN SALE OTRA SEÑAL Tx HACIA LA PC. POR LO TANTO DEBE CONECTARSE A Rx DEL PUERTO SERIAL DE LA PC. (EN ESTE DISEÑO NO SE USARÁ).
15	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
16	Vcc	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE

		PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.
--	--	---

En este ambiente de diseño lo más adecuado sería utilizar el Z80 SIO para las comunicaciones seriales, sin embargo dicho circuito integrado es difícil de conseguir hoy en día y por lo tanto se ha decidido implementar un sistema de comunicaciones con el MAX232, un PIC16F628A y un 74LS373.

La siguiente figura muestra los pines del 74LS373.

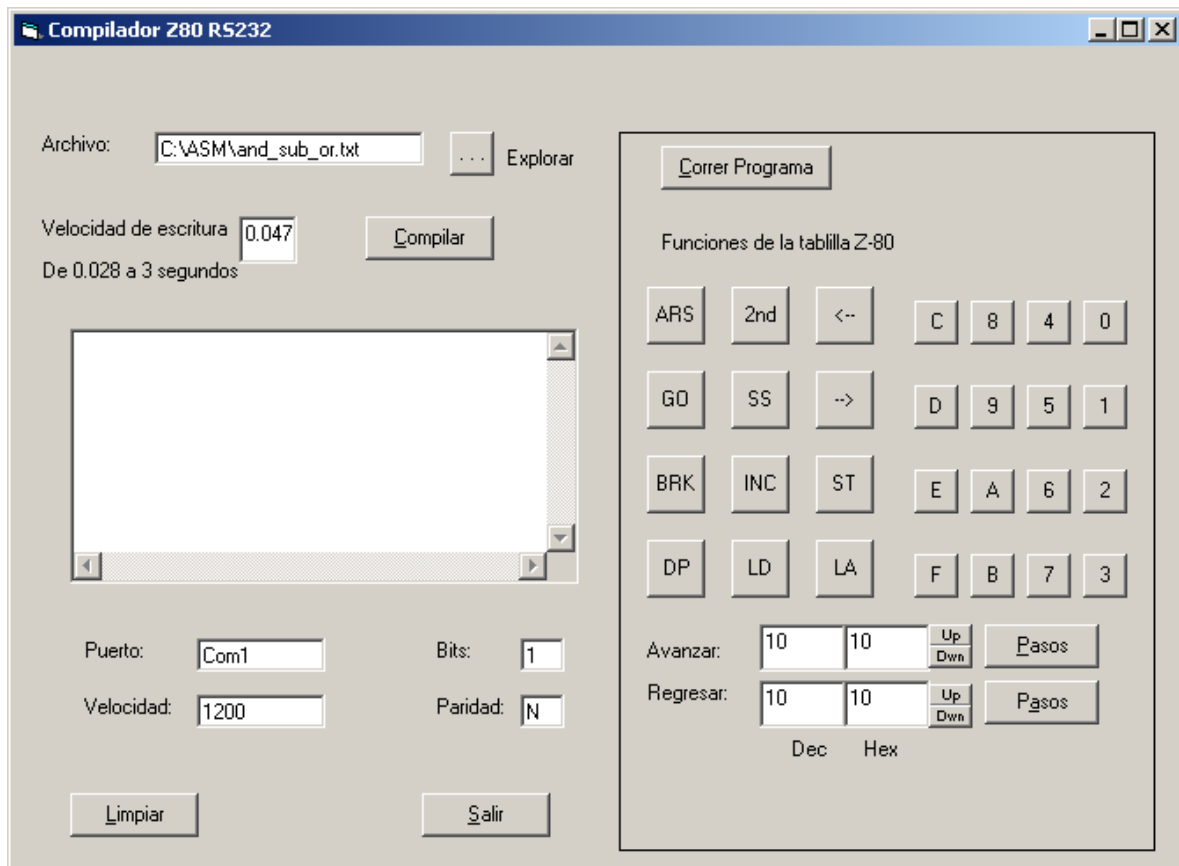


El PIC2628A entregará y recibirá datos al Z80 PIO-1 por el puerto 09h, tal como se estableció en un apartado anterior de este documento, y tiene su propio software de funcionalidad, mismo que se proporciona en el Apéndice 2. Para esto, el PIC usará los pines RA0-RA3 y RB4-RB7 como puerto de 8 bits conectado directamente al Puerto B del Z80 PIO-2. Mientras tanto, RB1 será usado para recepción de los datos seriales que vienen del MAX232 y a su vez del puerto serial RS-232. Así mismo el pin RB0 se usará para pedir una interrupción (INT) al Z80 CPU, quien al atenderla dejará pasar un byte completo desde el Z80 PIO-1, para darle su adecuado tratamiento. RB2 es salida de Tx dirigida al MAX232.

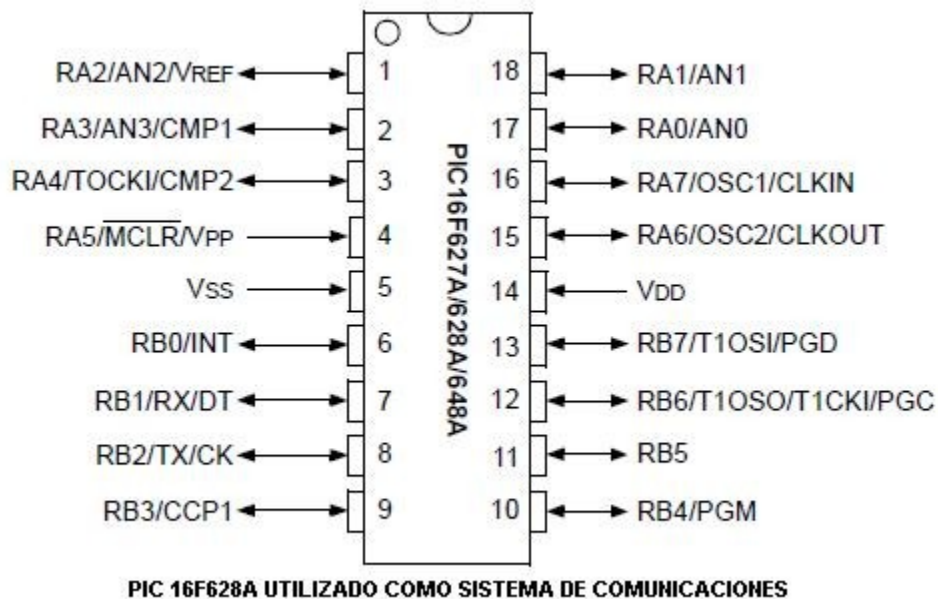
Tan pronto como se enciende el sistema computador, el PIC envía por el puerto serial hacia la PC, el siguiente mensaje:

“Bienvenido “

En la PC deberá correrse un programa que efectúe comunicaciones seriales, por ejemplo Hyperterminal por el puerto serie disponible. Para este diseño, también se realizó un programa en Visual Basic 6.0 para comunicarse con el sistema computador que se está diseñando, el cual se ha denominado Z80Conv_v5.exe y que es una interface muy similar al teclado de matriz del propio sistema computador, con algunas funciones adicionales, programa que se analizará más adelante en un apartado especial.



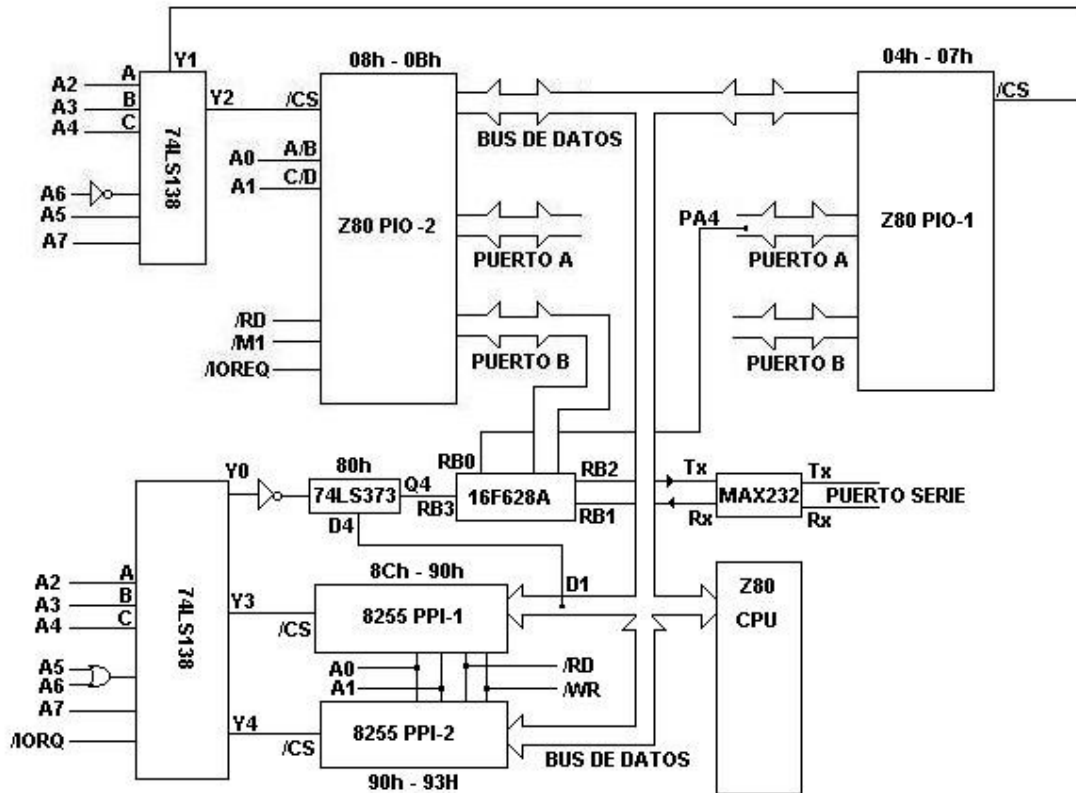
Se muestran en la siguiente figura los pines del PIC 16F628A:



Descripción de los pines del PIC 16F628A con su funcionalidad para este sistema computador bajo diseño:

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1-2, 17-18	PUERTO A, 4 bits	SALIDAS O ENTRADAS. NIBBLE ALTO QUE SE CONECTA A PB4-PB7 DEL Z80 PIO-2 (PUERTO 09h).
3	RA4	NO SE USA
4	MCLR	ENTRADA DE RESET QUE PROVIENE DEL RESET GENERAL DEL SISTEMA COMPUTADOR.
5	Vss	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINDA TIERRA O MASA.
6	RB0	SALIDA UTILIZADA PARA INFORMAR AL Z80 CPU QUE HAY UN DATO DISPONIBLE RECIBIDO DEL PUERTO DE COMUNICACIONES. SE CONECTA AL PIN 10 DEL PUERTO A DEL Z80 PIO-1. (04h).
7	RB1	ENTRADA Rx UTILIZADA PARA LA RECEPCIÓN DE DATOS SERIALES QUE PROVIENEN DEL MAX232 Y A SU VEZ DE LA PC POR EL PUERTO SERIAL.
8	RB2	SALIDA Tx UTILIZADA PARA ENVIAR DATOS AL SISTEMA DE COMUNICACIONES A TRAVES DEL MAX232 Y HACIA UNA PC POR EL PUERTO SERIAL.
9	RB3	ENTRADA UTILIZADA POR EL PIN Q4 DEL 74LS373 COMO SEÑAL ACK DESDE EL Z80 CPU, "DATO ACEPTADO".
10-13	PUERTO B, 4 BITS	SALIDAS O ENTRADAS. NIBBLE BAJO QUE SE CONECTA A PB0-PB3 DEL Z80 PIO-2 (PUERTO 09h)
14	VDD	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.
15-16	XTAL	PINES QUE SE CONECTAN A UN CRISTAL DE CUARZO DE 4 MHz PARA DAR COMPLETA EXACTITUD A LA VELOCIDAD DE TRANSMISION DEL SISTEMA DE COMUNICACIONES A 1200 bps.

A continuación definiremos el mapeo del circuito integrado 74LS373 utilizado para informar al PIC 16F628A que el dato enviado ha sido debidamente aceptado.



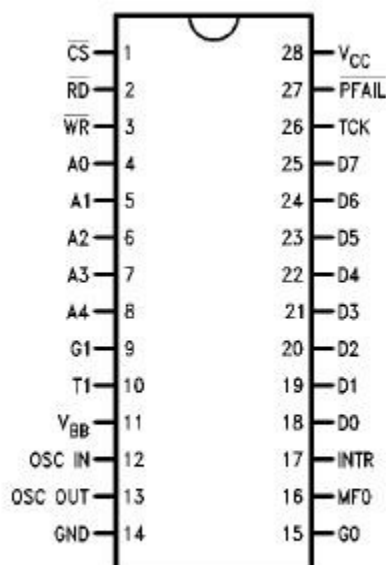
CONEXION FINAL 74LS373 - PIC16F628A -MAX232

PARTE VIII

RELOJ DE TIEMPO REAL DP8750A (TIMER CLOCK PERIPHERAL -TCP)

Con éste circuito integrado, pretendemos proporcionar al sistema de cómputo Z80 una manera de contar el tiempo real, ya sea en horas, minutos y segundos, de tal manera que cuando tenga interacción con el mundo exterior, esté a la par con el mundo que nos rodea.

En la siguiente figura se muestra la definición de pines del DP8750A



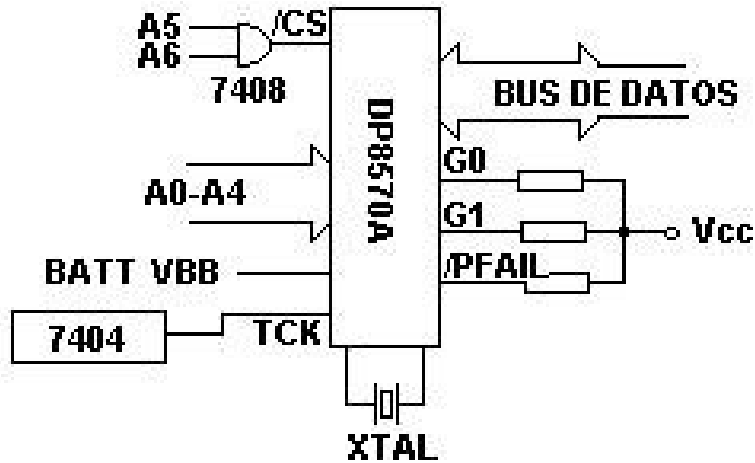
DP8750A RTC O TCP

En la siguiente tabla aparecen los números de pin y su función en este diseño.

PIN	NOMBRE	DESCRIPCION, FUNCIONES Y NIVELES
1	/CS	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICA AL RTC QUE DEBE HABILITARSE PARA OPERACIONES DE LECTURA Y ESCRITURA.
2	/RD	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICAL AL RTC QUE LA OPERACIÓN ACTUAL ES DE LECTURA.
3	/WR	ENTRADA QUE SE ACTIVA A NIVEL BAJO E INDICA AL RTC QUE LA OPERACIÓN ACTUAL ES DE ESCRITURA.
4-8	A0 - A4	ENTRADAS QUE SE ACTIVAN A NIVEL ALTO E INDICAN ALA RTC QUE LA COMBINACIÓN PRESENTE DE ESTAS LINEAS SELECCIONAN UNO DE LOS REGISTROS INTERNOS.
9	G1	ENTRADA QUE SE ACTIVA A NIVEL BAJO EN LOS MODOS 0, 1 Y 2 PARA ACTIVAR EL TIMER 1. EN EL MODO 3, G1 SE DISPARA CON EL FILO POSITIVO DE UN PULSO DE ENTRADA PARA INCIAR EL TIMER 1.
10	T1	SALIDA ACTIVA A NIVEL ALTO O BAJO, SEGÚN LO PROGRAME EL USUARIO, ASÍ COMO PUSH-PULL U OPEN DRAIN. SI SE CONECTA UNA BATERÍA DE RESPALDO $V_{BB} > V_{CC}$, T1 DEBERÁ PROGRAMARSE COMO OPEN DRAIN.
11	VBB	ENTRADA QUE ACEPTA UN VOLTAJE DE BATERÍA EXTERNA PARA RESPALDO DEL TIEMPO (HORA, MINUTOS Y SEGUNDOS) DEL RTC. INTERNAMENTE VBB ENTRA EN FUNCIONAMIENTO CUANDO V_{CC} ES MENOR QUE VBB. SIN NO SE VA A USAR VBB, ESTE PIN DEBERÁ CONECTARSE A GND.
12	OSC IN	ENTRADA DEL OSCILADOR INTERNO DEL RTC. SE CONECTA UN BORNE DE UN CRISTAL DE CUARZO.

13	OSC OUT	SALIDA DEL OSCILADOR INTERNO DEL RTC. SE CONECTA AL OTRO BORNE DEL CRISTAL DE CUARZO.
14	GND	NIVEL BAJO O NEGATIVO DE LA FUENTE DE PODER, DENOMINADA TIERRA O MASA.
15	G0	ENTRADA QUE SE ACTIVA A NIVEL BAJO EN LOS MODOS 0, 1 Y 2 PARA ACTIVAR EL TIMER 0. EN EL MODO 3, G0 SE DISPARA CON EL FILO POSITIVO DE UN PULSO DE ENTRADA PARA INICIAR EL TIMER 0.
16	MFO	SALIDA ACTIVA A NIVEL ALTO O BAJO, SEGÚN LA PROGRAMACION DEL USUARIO. SIGNIFICA SALIDA DE MULTI-FUNCIÓN Y SE PUEDE USAR COMO UNA SEGUNDA SEÑAL PARA PRODUCIR UNA INTERRUPCION (INT) AL Z80 CPU. MFO TAMBIEN PUEDE SER UNA SALIDA DEL OSCILADOR INTERNO, O EL TIMER 0. PUEDE SER PROGRAMADO EN PUSH-PULL U OPEN DRAIN. SI SE USA $V_{BB} > V_{cc}$, MFO DEBE PROGRAMARSE EN OPEN DRAIN.
17	INTR	SALIDA ACTIVA A NIVEL ALTO O BAJO SEGÚN LA PROGRAMACION DEL USUARIO, USADA PARA PRODUCIR UNA INTERRUPCION AL Z80 CPU CUANDO UN EVENTO DE TIEMPO O FALLA DE ENERGÍA OCURRAN. PUEDE PROGRAMARSE EN PUSH-PULL U OPEN DRAIN. SI SE USA CON $V_{BB} > V_{cc}$, INTR DEBE PROGRAMARSE EN OPEN DRAIN. PARA BORRAR INTR, DEBERÁ ESCRIBIRSE UN 1 EN LOS BITS APROPIADOS DEL REGISTRO PRINCIPAL DE ESTADO.
18-25	D0 - D7	ENTRADAS/SALIDAS QUE CONSTITUYEN EL BUS DE DATOS DEL RTC.
26	TCK	ENTRADA QUE SE ENCARGA DE RECIBIR PULSOS DE RELOJ PARA AMBOS TIMERS 0 Y 1 CUANDO TIENEN SELECCIONADO ACTIVARSE POR RELOJ EXTERNO.
27	/PFAIL	ENTRADA QUE SE ACTIVA A NIVEL BAJO. PUEDE TENER UNA SEÑAL DIGITAL EXTERNA QUE PROVENGA DE UN SISTEMA DE DETECCIÓN DE ENERGIA. CUANDO /PFAIL SE ACTIVA CON UN NIVEL BAJO, EL TCP VA AL MODO "CERRADO" EN UN MINIMO DE 20 useg Y UN MAXIMO DE 63 useg. EN EL MODO DE SIMPLE FUENTE DE PODER, ESTE PIN NO SE USA COMO ENTRADA Y DEBERÁ CONECTARSE A V_{cc} .
28	V_{cc}	VOLTAJE POSITIVO DE ALIMENTACIÓN PARA LA MEMORIA RAM, QUE PUEDE ESTAR ENTRE 3 Y 5.25 VOLTS DE CORRIENTE DIRECTA, PREFERENTEMENTE BIEN FILTRADA.

En la siguiente figura se muestra la conexión final del RTC DP8570A en la motherboard.



CONEXION FINAL RTC DP8570A

PARTE IX

LAS FUNCIONES DEL TECLADO

Tal como vimos en la Parte V y en la Parte VII, el teclado consta de varios botones que tienen asociada una función en especial. Los botones con los dígitos 0-9 y de A a F, sólo realizan la función de introducir ese número.

Las teclas con función son las siguientes:

ARS, 2ND, ←, GO, SS, →, BRK, IN, ST, DP, LD y LA. Vamos a comentar cada una de ellas.

ARS

Inicia en la localidad 0A90h del BIOS y se encarga de cargar en la RAM, comenzando desde la localidad 2100h, datos que provengan desde una PC vía el puerto RS-232. De hecho, ARS es (A)synchronous RS(232). Este es un DMA muy rudimentario, pues si se hace participar al Z80 CPU para el manejo de la información que va llegando al depositarla en la RAM. Sirve para cargar un programa en código HEX desde la PC al sistema computador, sin necesidad de teclearlo todo, sino que desde un archivo de texto y haciendo uso del programa mencionado en la Parte VII, Z80Conv_v5.exe se hace esta función.

2ND

Actualmente no tiene ninguna programación, pero está reservada para el futuro, o lo que un nuevo programador quiera añadir.



Flecha izquierda, función que inicia en la localidad 0820h del BIOS y se encarga de retroceder ya sea una localidad completa, o bien un dígito durante una introducción de dirección o dato.

GO

Esta tecla permite ejecutar el programa que está residente en la RAM a partir de la localidad 2100h. La secuencia será GO y enseguida DP para realmente ejecutar el programa. Su función inicia en la localidad 09A0h del BIOS.

SS

Actualmente no tiene ninguna programación, pero está reservada para el futuro, o lo que un nuevo programador quiera añadir.



Flecha derecha, función que inicia en la localidad 0800h del BIOS y se encarga de avanzar ya sea una localidad completa o bien, un dígito durante una introducción de dirección o dato.

BRK

Función Break, que interrumpe cualquier función en proceso, haciendo salir al Z80 de la rutina en que se encuentre para ir al inicio, excepto cuando está atendiendo una rutina invocada por una INT o NMI.

IN

Función que inicia en 1290h del BIOS y permite seleccionar un puerto en el rango de 00h a FFh, para entrada de datos. Después de oprimir el botón IN, se tecleará el número de puerto, seguido del botón DP. En ese instante, el puerto seleccionado será leído por el Z80 CPU y el dato será almacenado en la localidad 2032h.

ST

Este botón invoca un STOP, que en mnemónicos del Z80 CPU provoca un HALT. Inicia en la localidad 06C0h de BIOS.

DP

Este botón es similar a INTRO o ENTER en un teclado convencional.

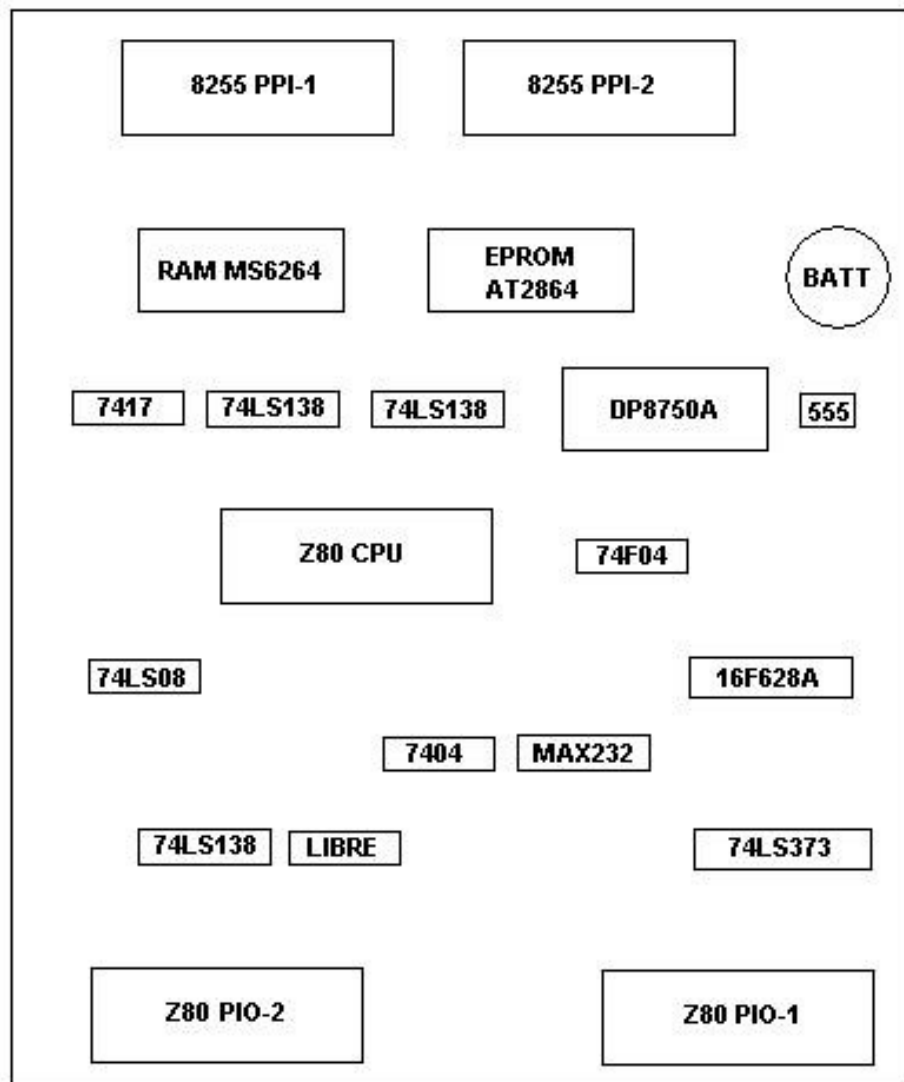
LD

La función inicia en la localidad 0840h del BIOS y permite cargar una localidad de la memoria RAM con un dato. Puede tratarse de cualquier localidad de la RAM, pero no de la ROM. Incluso puede tratarse de la RAM de 32 bytes que contiene el RTC DP8750A. Es la manera manual de introducir un programa a partir de la localidad 2100h. Puede introducirse una rutina en cualquiera otra localidad, siempre y cuando en la 2100H se ponga un JP XYYY, para hacer un salto (C3 YY XX).

LA

Su función inicia en la localidad 0700h del BIOS y permite seleccionar un puerto para salida de datos. Después de oprimir el botón LA, se tecleará el número de puerto válido, de 00h a FFh y enseguida el dato deseado. Al final se deberá oprimir DP para que tenga efecto esta función.

En la siguiente figura se muestra un diagrama de bloques de la motherboard de este sistema computador, con la disposición real de los circuitos integrados que la componen. Los puertos de entrada y salida están justo donde los pines de estos circuitos se encuentran.



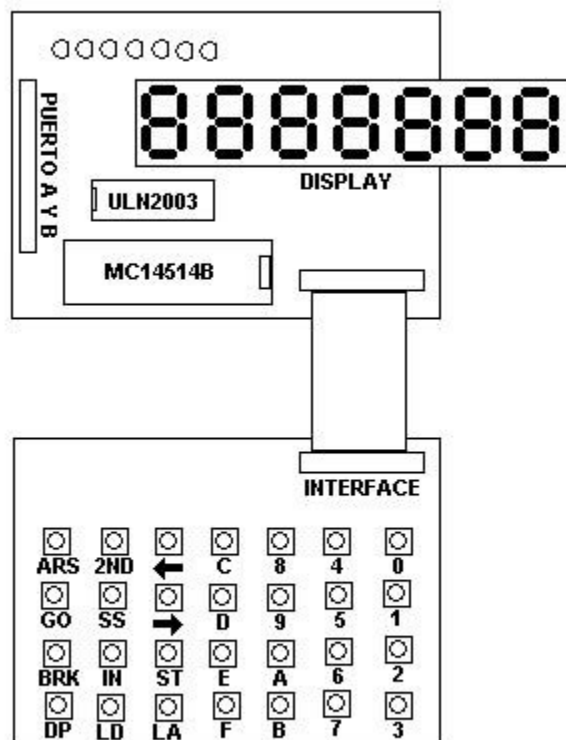
MOTHERBOARD Z80

En esta otra figura se muestra la tablilla donde se ha montado el Display LCD de 16 caracteres por 2 líneas, con 5 botones de función, RST, NMI e INT y 2 deshabilitados reservados para futuros diseños.



DISPLAY LCD 16 X 2

En la siguiente figura se muestra la tablilla del teclado y del Display de segmentos de LEDs.



DIPLAY Y TECLADO Z80

PARTE X

EJEMPLOS DE USO DEL SISTEMA COMPUTARIZADO Y PROGRAMACIÓN

Ejemplo 1: INTRODUCIR DATOS EN LOCALIDADES DE LA RAM

Supongamos que deseamos insertar en las localidades de memoria RAM algunos datos iniciando en la 2200h y terminando en la 2205h (6 datos). Los datos serán número del 01 al 06.

1.- Encender el sistema: Aparece en el display de leds

2100 F7

2.- Oprimir el botón LD

El Display se borra totalmente

3.- Introducir los números 2200 uno a uno...

Van apareciendo los dígitos.

4.- Debido a que el dígito 3 no enciende, esto es para separar direcciones de datos, insertar un 0.

Display 2200_

5.- insertar 01, primer dato.

Display; 2200_01

6.- Oprimir DP (Enter)

Display: 2201_11 (este 11 es un dato que ya estaba en la 2201h). Se notará que el display cambió a la siguiente localidad.

7.- De aquí en adelante se introducen solo los datos y se oprime DT.

Display: 2205_06

8.- Oprimir la tecla BRK para terminar la introducción de datos en RAM.

Display; Parpadea

9.- Con flechas ← o → retroceder o avanzar, para verificar los datos introducidos.

Ejemplo 2: CORRECCIÓN DE UNO O VARIOS DIGITOS CUANDO SE ESTÁ INTRODUCIENDO UNA DIRECCIÓN O UN DATO.

Supongamos que en el ejemplo anterior, equivocamos la introducción de la localidad 2200h y hemos escrito 2300_. Entonces procedemos como sigue:

1.- Oprimimos la tecla ← tantas veces como queramos regresar a la posición del dígito a corregir.

Display: 2__

2.- Tecleamos lo correcto: 200

Display: 2200_F1. Hemos tecleado F1 en vez de 01

3.- Presionamos el botón ← 2 veces:

Display: 2200_ __

4.- Escribimos el dato correcto:

Display 2200_01

5.- Finalmente oprimimos DP y luego BRK.

**Ejemplo 3: SACAR UN DATO POR UN PUERTO PROPORCIONADO POR LAS INTERFACES
INSTALADAS EN EL SISTEMA COMPUTADOR: 04h - 07h, 08h - 0Bh, 8Ch - 8Fh, 90h - 93h.**

Tenemos a la mano el puerto 08h en el cual hay instalado un bloque de 8 leds en color rojo. Por este bloque sacaremos el número A5h, y procedemos en la forma siguiente:

1.- Damos un RST al sistema con el botón correspondiente.

Display: Parpadea y se posiciona en 2100_F7

2.- Oprimir la tecla LA

Display: Se borra totalmente

3.- teclear lo siguiente: 0, 8, 0, A, 5

Display: 08_A5

5.- Oprimir DP

Display: 08_A5

Paquete LEDS: 1010 0101

6.- Oprimir BRK

**Ejemplo 4: INTRODUCIR UN DATO POR ALGUNO DE LOS PUERTOS PROPORCIONADO POR LAS
INTERFACES INSTALADAS EN EL SISTEMA COMPUTADOR: 04h - 07h, 08h - 0Bh, 8Ch - 8Fh, 90h -
93h.**

Recordaremos que por sistema, los datos leídos de algún puerto se depositan en la localidad 2032h de la RAM. Una vez leído y almacenado podemos navegar esa localidad para verificar el dato y darle alguna utilidad. Aprovecharemos el puerto de entrada 04h acoplado al teclado matricial para leerlo.

1.- Damos un RST al sistema con el botón correspondiente.

Display: Parpadea y se posiciona en 2100_F7

2.- Oprimir el botón IN

Display: Se borra totalmente

3.- Oprimir las siguientes teclas: 0, 4

Display: __ 0 4 _ _ _

4.- Oprimir DP

Display: __ 0 4 _ 0 0 Se ha leído un 00h

5.- Oprimir BRK, para salir de esta rutina.

Display: __ 0 4 _ 0 0

6.- Oprimir la tecla LD.

Display: se borra totalmente

7.- Teclear la siguiente dirección: 2 0 3 2

Display: 2032_ _ _

8.- Oprimir BRK

Display: 2032_ _ _

9.- Con flechas ← y luego → retroceder una posición y luego avanzar a la 2032h

Display: 2031_09

Display: 2032_00 Que es el valor leído por el puerto 04h

Ejemplo 5: AJUSTAR LA HORA EN EL RELOJ DE TIEMPO REAL

El objetivo es poner la hora y los minutos y si es posible, los segundos al tiempo local de la zona horaria donde se use el sistema computador.

El RTC DP8570A consta de los registros 06h para los segundos, 07h para los minutos y 08h para las horas. Debido a que dicho circuito integrado está mapeado en el puerto A0h, las direcciones reales son: A6h para segundos, A7h para minutos y A8h para horas.

Supongamos que la hora local en este momento es de 4:55:07 pm, así que procedemos de la siguiente manera:

1.- Oprimir **RST** para reiniciar el sistema.

Display: Se borra y aparece 2100_1F

2.- Oprimir el botón **LA** del teclado.

Display : Se borra totalmente

3.- Teclear A, 6, 0, 0, 7 y al final **DP**

Display: __ A6_07 Se ha introducido el valor 07 en el registro de los segundos del RTC

4.- Oprimir **BRK**

Display: __ A6_07

5.- Oprimir nuevamente el botón **LA** del teclado.

Display: Se borra totalmente

6.- Teclear A, 7, 0, 5, 5 y al final **DP**

Display: __ A7_55 Se ha introducido el valor 55 en el registro de los minutos del RTC

7.- Oprimir **BRK**

Display: __ A7_55

8.- Oprimir por tercera vez el botón **LA** del teclado

Display: Se borra totalmente

9.- Teclear A, 8, 0, 0, 4 y al final **DP**

Display: __ A8_04 Se ha introducido el valor 04 en el registro de las horas del RTC

10.- Oprimir **BRK** en el teclado y luego **RST** y **NMI** en la tablilla del display LCD

Display: 04 : 55

Ejemplo 6: REALIZAR UNA OPERACIÓN MATEMATICA SENCILLA, A TRAVES DE CODIGO HEX INTRODUCIDO MANUALMENTE.

En este ejemplo introduciremos el código en HEX de las líneas adecuadas para sumar el contenido de 2 localidades consecutivas y poner el resultado en la tercera localidad. La suma será de 8 bits y no deberá rebasar FFh.

Introducimos el valor 2Ah en la localidad de memoria RAM 2200h y el valor 3Bh en la localidad 2201h. Sumaremos ambos valores y el resultado lo colocaremos en la localidad 2202h. Posteriormente consultaremos ese resultado, de la manera que ya se ha descrito en este documento. El valor esperado será 65h.

LOCALIDAD	CODIGO MAQUINA	CODIGO HEX
2100 - 2101	LD A,2Ah	3E 2A
2102 - 2104	LD (2200h),A	32 00 22
2105	LD B,A	47
2106 - 2107	LD A, 3Bh	3E 3B
2108 - 210A	LD (2201h),A	32 01 22
210B	ADD B	80
210C -210E	LD (2202h),A	32 02 22
210F - 2110	OUT (08h),A	D3 08
2111 - 2113	JP 2100h	C3 00 21

Display: Apagado totalmente

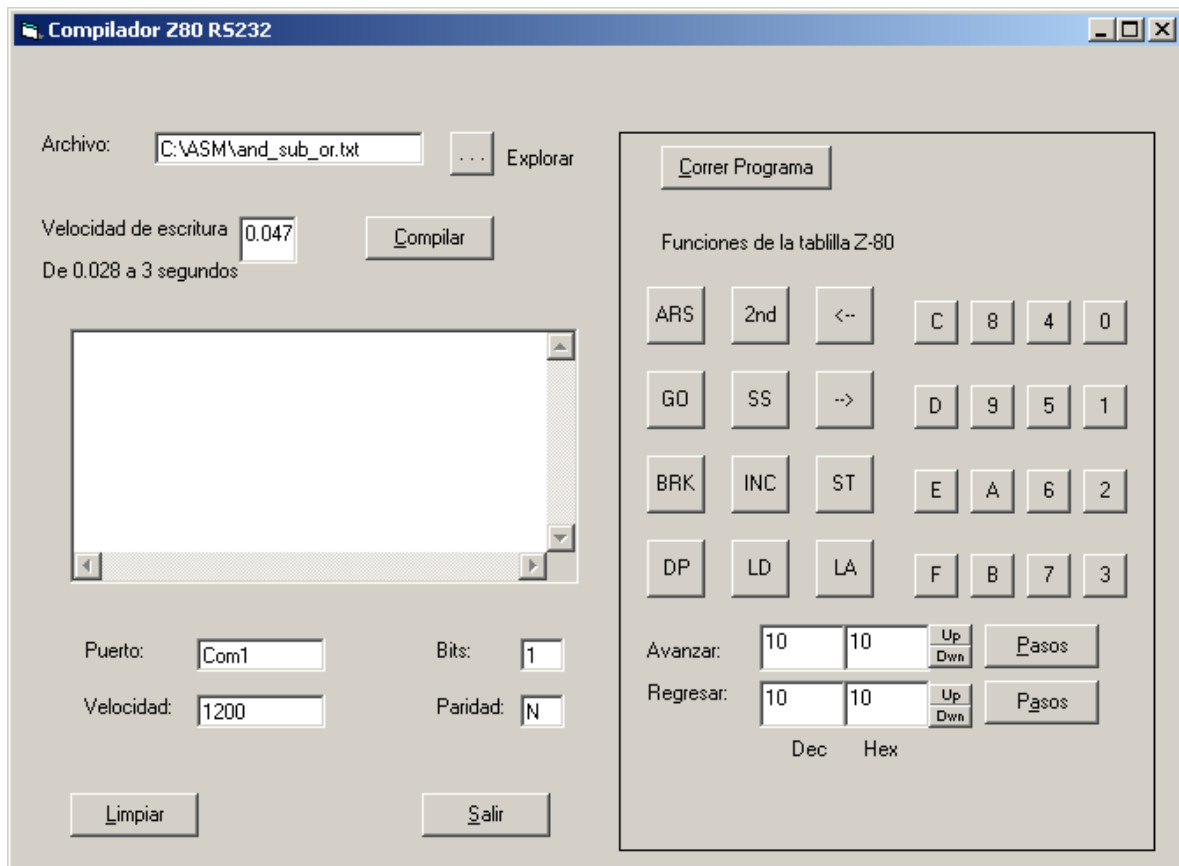
Paquete de leds puerto 08h: 0110-0101 65h resultado

PARTE XI

UTILIZACIÓN DEL SOFTWARE COMO INTERFACE ENTRE UNA PC Y EL SISTEMA COMPUTADOR

En la PARTE VII de este documentos e mencionó la existencia de un software, diseñado por el autor, para comunicar una PC con el sistema computador vía el puerto serial de ambos dispositivos.

Se muestra a continuación la pantalla principal de dicho software:



Este software se puede invocar con el nombre de Z80Conv_v5.exe en la PC. Está diseñado con VB6.0 y hace uso del puerto Com1 a 1200 bps, 8 bits, Ninguna paridad y 1 bit de parada.

Se notará que puede cargarse un archivo de texto que contenga un programa en lenguaje ensamblador, mismo que al seleccionar el botón Compilar se iniciará una transmisión desde la PC hacia el sistema computador, vaciando directamente el código HEX en las localidades de memoria que el propio programa tenga señalados en su código.

Veamos un segmento de un programa en código ensamblador:

```
ORG 2100

LD BC,0f0a

OUT (C),B

LD D,07

LD BC,8008

OUT (C),B
```

RRC B

CALL 2150

CALL 0048

DEC D

JR NZ,f3

LD D,07

LD BC,0108

OUT (C),B

RLC B

CALL 2150

CALL 0048

DEC D

JR NZ,f3

JP 2105

NOP

ORG 2150

PUSH HL

PUSH DE

PUSH AF

LD D,07

LD IX,2200

LD IY,2020

LD A,(IX+00)

LD (IY+00),A

INC IY

INC IX

DEC D

JR NZ,f3

CALL 03c9

POP AF

POP DE

POP HL

RET

ORG 2200

DATA 6e08,1e00

DATA 0000,0000

El software tiene su propio convertidor de lenguaje ensamblador a código HEX, es decir tiene su propio compilador, y el resultado de este código se muestra a continuación:

;-----

L21000

01

0a

0f

ed

41

16

07

01

08

80

ed

41

cb

08

cd

50

21

cd

48

00

15

20

f3

16

07

01

08

01

ed

41

cb

00

cd

50

21

cd

48

00

15

20

f3

c3

05

21

00

L21500

e5

d5

f5

16

07

dd

21

00

22

fd

21

20

20

dd

7e

00

fd

77

00

fd

23

dd

23

15

20

f3

cd

c9

03

f1

d1

e1

c9

L22000

6e

08

1e

00

00

00

00

00

;-----

El código se ha impreso en estas hojas en forma vertical, porque el software en la PC envía un carácter CF+LF después de cada byte de código HEX, con la finalidad de que el sistema computador identifique cada uno de ellos.

Se notará que la secuencia L21000, L21500 y L2200, van en un solo envío y que terminan con un cero adicional, por la razón que en el sistema computador, así se exige introducir direcciones de RAM, por el Dígito 3 del Display que aparece apagado para separar direcciones de datos.

El sistema computador va recibiendo esta información por el puerto serial, pasando primeramente por el MAX232, entrando al PIC16F628A donde se recompone cada byte en paralelo, avisando al Z80 CPU que existe un dato disponible en el puerto 09h, aceptándolo y colocándolo en la localidad de memoria señalada por el comando 'L' y en forma consecutiva, colocando los bytes subsecuentes, hasta la llegada de otro comando 'L'. Mientras se van recibiendo los datos, el led blanco adscrito al PIC16F628A enciende y apaga y en el paquete de leds conectado al puerto 08h aparece en binario el código recibido.

En el cuadro de texto del mismo software aparece todo el código transferido en tres columnas: Las localidades, el código HEX y el lenguaje ensamblador. Una vez concluida la transferencia se puede señalar y oprimir con el puntero del mouse el botón **GO** y luego el botón **DP**, para ejecutar el programa enviado, en el sistema computador. Por supuesto también se pueden oprimir los botones reales en el teclado matricial del sistema computador para tener el mismo efecto.

En el ejemplo de código descrito en párrafos anteriores, el paquete de leds en el puerto 08h aparece un movimiento de luces de izquierda a derecha y viceversa.

Recordar que la memoria RAM tiene 8,192 bytes de capacidad, mapeados desde 2000h hasta 3FFFh y que está reservada una zona desde la localidad 2000h hasta la 2099h (donde se incluyen variables que usa el BIOS y el STACK), por lo que la capacidad útil para programas va de 2100h a 3FFFh, es decir 1EFFh o sea 7,935 bytes.

Aprovechando las facilidades de un software que simula un teclado físico, se han colocado botones y recuadros para avanzar o retroceder rápidamente de una dirección a otra, efectuando pasos largos, lo que en el teclado matricial involucra más tiempo:

Avanzar:	10	10	Up Down	Pasos
Regresar:	10	10	Up Down	Pasos
	Dec	Hex		

APÉNDICE 1

DATOS TECNICOS Y CARACTERÍSTICAS DEL SISTEMA COMPUTADOR BAJO DISEÑO DE ESTE PROYECTO

DESCRIPCION	RANGOS/NOMBRES	CONDICIONES/CANTIDADES
Voltaje de Alimentación	8 a 16 VCD	Alimentación General
Consumo Básico de Corriente	220 mA	Sin periféricos
Microprocesador	Z80 CPU @ 4 MHz	1 unidad
Parallel Input/Output	Z80 PIO	2 unidades
Peripheral Interface Adapter	8255 PIA	2 unidades
Reloj de Tiempo Real	DP8750A	1 unidad
Memoria RAM	MS6264	8,192 bytes (2000h), 1 unidad
Memoria ROM	AT2864B	8,192 bytes (2000h), 1 unidad
Microcontrolador	PIC 16F628A	Sistema de Comunicaciones, 1 unidad
Interface RS232	MAX232	Puerto de comunicaciones, 1 unidad
Decoder 4 a 16 líneas	MC14514B	Selector Teclado matricial, 1 unidad
Driver dígitos de display	ULN2003	Inversor emisor común, 1 unidad
Transistores PNP	BC557	Drivers segmentos leds 7 unidades
Driver Colector Abierto	7407	Sumador INT
Decoder	74LS138	Selector Memorias RAM, ROM, PIOs, PIAs, RTC, PIC16F628A, 3 unidades
Cuatro NAND de 2 entradas	74LS08	Lógica de selección, 1 unidad
Seis Inversores Shmith Trigger	74F04	Reloj 4 MHz, y drivers, 1 unidad
Temporizador	NE555	Monoestable RESET del sistema, 1 unidad
Seis Inversores Shmith Trigger	7404	Lógica de selección, 1 unidad
Ocho FF-Latch Tri Estado	74LS373	Puerto 80h, 1 unidad
Display LCD 16x2	1602 TMS	Display LCD
Displays de 8 segmentos	DA05 o similar	Juntos para formar un display longitudinal de 7 dígitos sin punto decimal. 7 unidades.
Batería de Litio	3.0 VCD	VBB respaldo para RTC, 1 unidad.
Botones de contacto por presión	RST, NMI, INT, Teclado	35 unidades
Bases para circuitos integrados	8, 14, 16, 18, 20, 24, 28 y	22 unidades

	40 pins	
--	---------	--

APÉNDICE 2

LISTADO EN CÓDIGO DE MÁQUINA Y ENSAMBLADOR, ASÍ COMO COMENTARIOS DEL BIOS DEL PROYECTO Z80.

```

0001 0000      ;Este programa es el bios de la tablilla del z80 Primer Proyecto y
0002 0000      ;contiene todas las Funciones necesarias para el manejo de datos y comandos
0003 0000      ;Puertos 04 - 07h PIO1 Teclado y display de dígitos de 7segmntos cátodo común
0004 0000      ;Puertos 07 - 0Bh PIO2 entrada datos del PIC y salida indicadora por leds
0005 0000      ;Puertos 8C - 8Fh PIA2 salidas para un display LCD 16x2
0006 0000      ;Puertos 90 - 93h PIA1 salidas para activar un motor stepper
0007 0000      ;puerto 80h salida de 1 bit (D0) para ACK del PIC16f628A -como RS-232 Com
Port
0008 0000      ;Utiliza un RTC DP8570A y una batería de Litio de 3V.
0009 0000      ;Se alimenta con una fuente de 12VCD proporcionando 5VCD para todo el
sistema
0010 0000      ;En la misma tablilla de la fuente se encuentra el driver del motor stepper
0011 0000      ;y el conector rs232 hacia una PC u otro dispositivo de comunicación serie.
0012 0000      ;El hardware de reset se implementa con un NE555 en modo monoestable
0013 0000      ;La memoria EPROM es del tipo AT68C64. La RAM es una MS6264.
0014 0000      ;El generador de reloj (CLK) es un 74F04 con cristal de 4 MHz.
0015 0000      ;La tablilla del Display LCD tiene además 5 botones para RST, NMI, INT y 2
0016 0000      ;de reserva. El teclado es una matriz de 4 x 7 botones y el conjunto es
0017 0000      ;controlado por el CI Cd4514B, auxiliado por transistores BC548 para controlar

```

```

0018 0000      ;el nivel de presión de teclas para alimentar los segmentos del display de
0019 0000      ;leds así como un ULN2003 para controlar los dígitos de dicho display.
0020 0000      ;La RAM está alimentada en forma auxiliar con un capacitor de 1 uF, para
0021 0000      ;tener la capacidad de retener la última información por al menos 1 semana.
0022 0000      ;Se tiene un paquete de 10 leds (8 usados) acoplado al Puerto 08h del PIA2
0023 0000      ;El botón de RST es doble, uno en la tablilla del LCD y otro en la mother
0024 0000      ;board cerca del NE555.
0025 0000      ;No está implementado el hardware para los leds indicadores IY, SP, PC, MEM
0026 0000      ;I/O, ERR, ARS, BRK, IR, AF, BC, DE, HL, IX.
0027 0000
0028 0000      ;Alfredo Segura.- Querétaro. Junio 2006.- Ok.-revisado junio 2008.
0029 0000      ;revisado ene 2009, julio 2012, julio 2017.
0030 0000
0031 0000      #define equ    .equ
0032 0000      #define end    .end
0033 0000      #define org     .org
0034 0000      #define byte .byte
0035 0000      ledslequ    2010h          ;juego de leds IY,SP,PC,MEM,I/O,ERR,ARS
0036 0000      ledsh      equ    2011h          ;juego de leds BRK,IR,AF,BC,DE,HL,IX
0037 0000      adhnh      equ    2012h          ;adnh.- address high nibble high
0038 0000      adhnl      equ    2013h          ;adnhl.- address high nibble low
0039 0000      adlnh      equ    2014h          ;adlnh.- address low nibble high
0040 0000      adlnl      equ    2015h          ;adlnl.- address low nibble low
0041 0000      vacío       equ    2016h
0042 0000      datnh      equ    2017h          ;datnh.- data nibble high
0043 0000      datnl      equ    2018h          ;datnl.- data nibble low

```

0044	0000	digte	equ	2019h	;es también la posición del dígito 1
0045	0000	addrl	equ	201Ah	;este espacio requiere de 2 bytes
0046	0000	addrh	equ	201Bh	;
0047	0000	data equ	201Ch		;dato introducido mediante el teclado
0048	0000	bande	equ	201Dh	;bandera de identificadores
0049	0000	datin	equ	201Eh	;dato de entrada por in a,(04h)
0050	0000	serdat	equ	201Fh	;dato que entra por RS232
0051	0000	digi10	equ	2020h	;Banderas1
0052	0000	digi9	equ	2021h	;Banderas2
0053	0000	digi8	equ	2022h	;Estos son los dígitos introducidos por
0054	0000	digi7	equ	2023h	;el usuario de 00h - 0Fh
0055	0000	digi6	equ	2024h	
0056	0000	digi5	equ	2025h	
0057	0000	digi4	equ	2026h	
0058	0000	digi3	equ	2027h	
0059	0000	digi2	equ	2028h	
0060	0000	digi1	equ	2029h	
0061	0000	tecl	equ	202Ah	;guarda el valor de la tecla presionada
0062	0000	fila	equ	202Bh	;guarda el valor de la fila 1-7
0063	0000	column	equ	202Ch	;guarda el valor de la columna 1-4
0064	0000	temp	equ	202Dh	;temporal para dato entre rutinas
0065	0000	pto08	equ	202Eh	;dato en el puerto PIO2-A (08h)
0066	0000	carac	equ	202Fh	;contador de caracteres de texto
0067	0000	inice	equ	2030h	;memoria donde inicia el texto de bienv
0068	0000	inice1	equ	2031h	;u otros textos de otros programas en RAM
0069	0000	;=====			

```

0070 0000      .org    0000h      ;rutina que se invoca con RST 00h --> C7
0071 0000 C3 00 01  jp 0100h      ;o reseteando el sistema
0072 0003      ;=====
0073 0008      org     0008h      ;rutina que se invoca con RST 08h --> CF
0074 0008 C3 00 01  jp 0100h
0075 000B      ;=====
0076 0010      org     0010h      ;rutina que se invoca con RST 10h --> D7
0077 0010 C3 00 01  jp 0100h
0078 0013      ;=====
0079 0018      org     0018h      ;rutina que se invoca con RST 18h --> DF
0080 0018 C3 00 01  jp 0100h
0081 001B      ;=====
0082 0020      org     0020h      ;rutina que se invoca con RST 20h --> E7
0083 0020 C3 70 10  jp motor
0084 0023      ;=====
0085 0028      org     0028h      ;rutina que se invoca con RST 28h --> EF
0086 0028 C3 00 01  jp 0100h
0087 002B      ;=====
0088 0030      org     0030h      ;rutina que se invoca con RST 30h --> F7
0089 0030 C3 00 01  jp 0100h
0090 0033      ;=====
0091 0038      org     0038h
0092 0038 F3      di
0093 0039 C3 70 10  jp motor      ;salta a la rutina 'motor' para activar el
0094 003C C9      ret            ;stepper motor (org 1070h)
0095 003D      ;=====

```

```

0096 0066      org    0066h
0097 0066 F3      di
0098 0067 CD B0 11  call reloj          ;salta a la rutina 'reloj' para mostrar
0099 006A C9      ret                  ;minutos y segundos del DP8570A RTC (org
0100 006B      ;=====
0101 0048      org    0048h
0102 0048 F5      push af          ;Rutina retar3 con un tiempo largo
0103 0049 C5      push bc
0104 004A D5      push de
0105 004B 06 49    ld b,049h
0106 004D 0E 29    aci3:    ld c,029h
0107 004F 16 15    aqi3:    ld d,015h
0108 0051 15      aki3:    dec d
0109 0052 20 FD    jr nz,aki3
0110 0054 0D      dec c
0111 0055 20 F8    jr nz,aqi3
0112 0057 05      dec b
0113 0058 20 F3    jr nz,aci3
0114 005A D1      pop de
0115 005B C1      pop bc
0116 005C F1      pop af
0117 005D C9      ret
0118 005E      ;=====
0119 0080      org    0080h
0120 0080 42 69 65 6E      byte  42h,69h,65h,6eh      ;B,i,e,n
0121 0084 76 65 6E 69      byte  76h,65h,6eh,69h      ;v,e,n,i

```

0122	0088 64 6F 20 61	byte	64h,6fh,20h,61h	;d,o ,a
0123	008C 6C 20 73 69	byte	6ch,20h,73h,69h	;l ,s,i
0124	0090 73 74 65 6D	byte	73h,74h,65h,6dh	;s,t,e,m
0125	0094 61 20 3E 3A	byte	61h,20h,3eh,3ah	;a, ,>;
0126	0098 20 20 20 20	byte	20h,20h,20h,20h	; , , ,
0127	009C 20 20 20 20	byte	20h,20h,20h,20h	; , , ,
0128	00A0		;=====	
0129	0100	org	0100h	
0130	0100 31 F0 20	ld sp,20F0h	;define el area stack EN 20F0h	
0131	0103 21 00 21	ld hl,2100h		
0132	0106 ED 56	im 1		
0133	0108 FB	ei		
0134	0109 01 8F 8B	ld bc,8b8fh	;PIA-1 Puertos A,B,C entradas Modo 0	
0135	010C ED 41	out (c),b		
0136	010E 01 93 8B	ld bc,8b93h	;PIA-2 Puertos A,B,C entradas Modo 0	
0137	0111 ED 41	out (c),b		
0138	0113 01 07 0F	ld bc,0f07h	;PIO1-B salidas	
0139	0116 ED 41	out (c),b		
0140	0118 01 06 4F	ld bc,4f06h	;PIO1-A entradas	
0141	011B ED 41	out (c),b		
0142	011D 01 0B 4F	ld bc,4f0Bh	;PIO2-B entradas	
0143	0120 ED 41	out (c),b		
0144	0122 01 0A 0F	ld bc,0f0Ah	;PIO2-A salidas	
0145	0125 ED 41	out (c),b		
0146	0127 01 09 00	ld bc,0009h	;saca 00h en PIO2-B	
0147	012A ED 41	out (c),b		

0148	012C 01 08 00	ld bc,0008h	;saca 00h en PIO2-A
0149	012F ED 41	out (c),b	
0150	0131 3E 00	ld a,00h	;banderas de leds apagadas
0151	0133 32 10 20	ld (ledsl),a	
0152	0136 CD 30 0F	call flags	;rutina que sensa los flags y los
0153	0139 00	nop	;guarda en ledsl y ledsh
0154	013A 3E 00	ld a,00h	
0155	013C D3 80	out (80h),a	;borra puerto 80h en 74LS373
0156	013E 3E 21	ld a,21h	
0157	0140 32 1B 20	ld (addrh),a	;valor addrh
0158	0143 3E 00	ld a,00h	
0159	0145 32 1A 20	ld (addrl),a	
0160	0148 2A 1A 20	ld hl,(addrl)	;toma la RAM
0161	014B 7E	ld a,(hl)	;lee su valor
0162	014C 32 1C 20	ld (data),a	;y lo guarda en data
0163	014F CD 30 0E	call desag	;incluye 'convi'
0164	0152 21 2F 20	ld hl,carac	;apunta a la localidad que indica la
0165	0155 36 20	ld (hl),20h	;cantidad de caracteres de bienv
0166	0157 21 80 00	ld hl,0080h	;memoria donde inicia el texto de bienv
0167	015A 22 30 20	ld (inice),hl	;o cualquier otro texto de otro programa
0168	015D CD 40 0F	call bienv	;prepara el LCD para mostrar texto
0169	0160 CD A0 0F	call muestr	;muestra texto "Bienvenido al sistema"
0170	0163 21 1D 20	sigla: ld hl,bande	;en la rutina principal, bit1=0 de 'bande'
0171	0166 CB 8E	res 1,(hl)	;para que no se muestre el dígito 1
0172	0168 CD C0 01	call scan	;busca una tecla presionada
0173	016B 3A 2C 20	ld a,(colum)	;solo si se presionó tecla hace 'decod'

0174	016E FE 00	cp 00h	;verifica si es '0' (Z no se afecta en ld)
0175	0170 20 28	jr nz,sitecl	;z=0, se oprimió una tecla
0176	0172 CD A0 04	call displ	;z=1 no se oprimió, display como está
0177	0175 DB 04	in a,(04h)	;sensa presencia dato de desde PIC16F628A
0178	0177 CB 67	bit 4,a	;si z=1, el bit4=0 no hay dato, regresa
0180	017B 00	nop	;si z=0, el bit4=1 si hay dato en RS232
0181	017C DB 09	in a,(09h)	;introduce el dato en Acc PIO2-B
0182	017E D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
0183	0180 32 1F 20	ld (serdat),a	;guarda dato en 'serdat'
0184	0183 CD A0 0B	call hexa	;convierte de ASCII a hex: serdat es hex
0185	0186 3E 01	ld a,01h	
0186	0188 D3 80	out (80h),a	;avisa a PIC dato leído pto 80h del 74LS373.
0187	018A 00	nop	
0188	018B 00	nop	
0189	018C 00	nop	
0190	018D 3E 00	ld a,00h	
0191	018F D3 80	out (80h),a	
0192	0191 3A 1F 20	ld a,(serdat)	;toma el dato convertido a Hex
0193	0194 32 2A 20	ld (tecl),a	;y lo guarda en "tecl" para proceder
0194	0197 C3 9D 01	jp r232s	;regresa a la rutina
0195	019A CD 60 02	sitecl: call decod	;z=0, tecla oprimida, ver valor (tecl)
0196	019D 21 1D 20	r232s: ld hl,bande	;carga el valor de 'bande'
0197	01A0 CB 46	bit 0,(hl)	;prueba el bit0 de bande (dígito o comando)
0198	01A2 20 BF	jr nz,siga	;si z=0, bit=1,es un dígito, no hacer nada
0199	01A4 CD 50 06	call ejecut	;si z=1, bit=0, es un comando, ejecutarlo
0200	01A7 C3 63 01	jp siga	

```

0201 01AA      ;=====
0202 01C0      org      01c0h
0203 01C0 F5    scan:    push af
0204 01C1 C5      push bc
0205 01C2 D5      push de
0206 01C3 3A 10 20 ld a,(ledsl)      ;toma el valor de ledsl
0207 01C6 CB B7    res 6,a            ;Apaga LED indicador de tecla presionada
0208 01C8 32 10 20 ld (ledsl),a
0209 01CB 06 02    ld b,02h          ;valor 0000 0010 para PIO1-B filas
0210 01CD 78      otro1:  ld a,b      ;
0211 01CE 2F      cpl              ;complementa por inversores del teclado
0212 01CF CB 87    res 0,a          ;deja en 0 el Bit0 de PB
0213 01D1 D3 05    out (05h),a      ;y saca el dato
0214 01D3 E3      ex (sp),hl
0215 01D4 E3      ex (sp),hl        ;hace un retardo
0216 01D5 DB 04    in a,(04h)        ;lee datos del teclado (columnas)
0217 01D7 E6 0F    and 0fh          ;enmascara los bits. Nibble bajo util
0218 01D9 20 0B    jr nz,tecpre      ;si hubo tecla presionada
0219 01DB CB 10    rl b              ;recorre 'b' a la izquierda un bit.
0220 01DD 30 EE    jr nc,otro1        ;si Cy=1 se termina el scan. Si no, continuar
0221 01DF 3E 00    ld a,00h          ;Acc=0
0222 01E1 32 2C 20 ld (colum),a      ;no se oprimió tecla
0223 01E4 18 1D    jr salida          ; si no encontró tecla sale de aquí.
0224 01E6 32 2C 20 tecpre: ld (colum),a ;en 'a' está la palabra de COLUMNAS
0225 01E9 78      ld a,b
0226 01EA 32 2B 20 ld (fila),a       ;en 'b' está la palabra de FILAS

```

0227	01ED 06 FE	salt:	ld b,0feh	;espera a que se suelte la tecla, por
0228	01EF 78		ld a,b	;lo que habilita todas las filas
0229	01F0 2F		cpl	;las complementa para activar teclado
0230	01F1 CB 87		res 0,a	
0231	01F3 D3 05		out (05h),a	
0232	01F5 3A 10 20		ld a,(ledsl)	;Enciende LED indicador de tecla presionada
0233	01F8 CB F7		set 6,a	
0234	01FA 32 10 20		ld (ledsl),a	
0235	01FD DB 04		in a,(04h)	;vuelve a sensor teclado
0236	01FF E6 0F		and 0fh	;enmascara el dato
0237	0201 20 EA		jr nz,salt	;si tiene algún valor, vuelve a checar
0238	0203 D1	salida:	pop de	;si Acc=0, sale. Aquí garantizamos
0239	0204 C1		pop bc	;que el usuario ya soltó la tecla.
0240	0205 F1		pop af	
0241	0206 C9		ret	;si al regresar colum=0 no hubo tecla
0242	0207		;=====	
0243	0260		org 0260h	
0244	0260 21 1D 20	decod:	ld hl,bande	;toma coordenadas del teclado...
0245	0263 3A 2B 20		ld a,(fila)	;toma el valor de la 'fila'. No hay fila0
0246	0266 FE 02		cp 02h	;compara con 0000 0010
0247	0268 CA 8A 02		jp z,fil1	;si es igual, salta a fil2
0248	026B FE 04		cp 04h	;0000 0100
0249	026D CA A2 02		jp z,fil2	
0250	0270 FE 08		cp 08h	;0000 1000
0251	0272 CA BA 02		jp z,fil3	
0252	0275 FE 10		cp 10h	;0001 0000

0253	0277 CA D2 02	jp z,fil4	
0254	027A FE 20	cp 20h	;0010 0000
0255	027C CA EA 02	jp z,fil5	
0256	027F FE 40	cp 40h	;0100 0000
0257	0281 CA 02 03	jp z,fil6	
0258	0284 FE 80	cp 80h	;1000 0000
0259	0286 CA 1A 03	jp z,fil7	
0260	0289 C9	ret	
0261	028A		;
0262	028A 3A 2C 20	fil1: ld a,(column)	;toma el valor de 'column'
0263	028D FE 01	cp 01h	;compara con 0000 0001
0264	028F CA 32 03	jp z,cor10	;en formato (x,y) renglón, columna
0265	0292 FE 02	cp 02h	;compara con 0000 0010
0266	0294 CA 3A 03	jp z,cor11	;en formato (x,y) renglón, columna
0267	0297 FE 04	cp 04h	;compara con 0000 0100
0268	0299 CA 42 03	jp z,cor12	;en formato (x,y) renglón, columna
0269	029C FE 08	cp 08h	;compara con 0000 1000
0270	029E CA 4A 03	jp z,cor13	;en formato (x,y) renglón, columna
0271	02A1 C9	ret	
0272	02A2 3A 2C 20	fil2: ld a,(column)	;toma el valor de 'column'
0273	02A5 FE 01	cp 01h	;compara con 0000 0001
0274	02A7 CA 52 03	jp z,cor20	;en formato (x,y) renglón, columna
0275	02AA FE 02	cp 02h	;compara con 0000 0010
0276	02AC CA 5A 03	jp z,cor21	;en formato (x,y) renglón, columna
0277	02AF FE 04	cp 04h	;compara con 0000 0100
0278	02B1 CA 62 03	jp z,cor22	;en formato (x,y) renglón, columna

0279	02B4 FE 08	cp 08h	;compara con 0000 1000
0280	02B6 CA 6A 03	jp z,cor23	;en formato (x,y) renglón, columna
0281	02B9 C9	ret	
0282	02BA 3A 2C 20	fil3: ld a,(column)	;toma el valor de 'column'
0283	02BD FE 01	cp 01h	;compara con 0000 0010
0284	02BF CA 72 03	jp z,cor30	;en formato (x,y) renglón, columna
0285	02C2 FE 02	cp 02h	;compara con 0000 0010
0286	02C4 CA 7A 03	jp z,cor31	;en formato (x,y) renglón, columna
0287	02C7 FE 04	cp 04h	;compara con 0000 0010
0288	02C9 CA 82 03	jp z,cor32	;en formato (x,y) renglón, columna
0289	02CC FE 08	cp 08h	;compara con 0000 0010
0290	02CE CA 8A 03	jp z,cor33	;en formato (x,y) renglón, columna
0291	02D1 C9	ret	
0292	02D2 3A 2C 20	fil4: ld a,(column)	;toma el valor de 'column'
0293	02D5 FE 01	cp 01h	;compara con 0000 0001
0294	02D7 CA 92 03	jp z,cor40	;en formato (x,y) renglón, columna
0295	02DA FE 02	cp 02h	;compara con 0000 0010
0296	02DC CA 9A 03	jp z,cor41	;en formato (x,y) renglón, columna
0297	02DF FE 04	cp 04h	;compara con 0000 0100
0298	02E1 CA A2 03	jp z,cor42	;en formato (x,y) renglón, columna
0299	02E4 FE 08	cp 08h	;compara con 0000 1000
0300	02E6 CA AA 03	jp z,cor43	;en formato (x,y) renglón, columna
0301	02E9 C9	ret	
0302	02EA 3A 2C 20	fil5: ld a,(column)	;toma el valor de 'column'
0303	02ED FE 01	cp 01h	;compara con 0000 0001
0304	02EF CA B2 03	jp z,cor50	;en formato (x,y) renglón, columna

0305	02F2 FE 02	cp 02h	;compara con 0000 0010
0306	02F4 CA BA 03	jp z,cor51	;en formato (x,y) renglón, columna
0307	02F7 FE 04	cp 04h	;compara con 0000 0100
0308	02F9 CA C2 03	jp z,cor52	;en formato (x,y) renglón, columna
0309	02FC FE 08	cp 08h	;compara con 0000 1000
0310	02FE CA CA 03	jp z,cor53	;en formato (x,y) renglón, columna
0311	0301 C9	ret	
0312	0302 3A 2C 20	fil6: ld a,(column)	;toma el valor de 'column'
0313	0305 FE 01	cp 01h	;compara con 0000 0001
0314	0307 CA D2 03	jp z,cor60	;en formato (x,y) renglón, columna
0315	030A FE 02	cp 02h	compara con 0000 0010
0316	030C CA DA 03	jp z,cor61	;en formato (x,y) renglón, columna
0317	030F FE 04	cp 04h	;compara con 0000 0100
0318	0311 CA E2 03	jp z,cor62	;en formato (x,y) renglón, columna
0319	0314 FE 08	cp 08h	;compara con 0000 1000
0320	0316 CA EA 03	jp z,cor63	;en formato (x,y) renglón, columna
0321	0319 C9	ret	
0322	031A 3A 2C 20	fil7: ld a,(column)	;toma el valor de 'column'
0323	031D FE 01	cp 01h	;compara con 0000 0001
0324	031F CA F2 03	jp z,cor70	;en formato (x,y) renglón, columna
0325	0322 FE 02	cp 02h	;compara con 0000 0010
0326	0324 CA FA 03	jp z,cor71	;en formato (x,y) renglón, columna
0327	0327 FE 04	cp 04h	;compara con 0000 0100
0328	0329 CA 02 04	jp z,cor72	;en formato (x,y) renglón, columna
0329	032C FE 08	cp 08h	;compara con 0000 1000
0330	032E CA 0A 04	jp z,cor73	;en formato (x,y) renglón, columna

0331	0331 C9	ret	
0332	0332	;	
0333	0332 3E 00	cor10: ld a,00h	
0334	0334 32 2A 20	ld (tecl),a	
0335	0337 CB C6	set 0,(hl)	;(bande) = xxxx xxx1, es un dígito
0336	0339 C9	ret	
0337	033A 3E 01	cor11: ld a,01h	
0338	033C 32 2A 20	ld (tecl),a	
0339	033F CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0340	0341 C9	ret	
0341	0342 3E 02	cor12: ld a,02h	
0342	0344 32 2A 20	ld (tecl),a	
0343	0347 CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0344	0349 C9	ret	
0345	034A 3E 03	cor13: ld a,03h	
0346	034C 32 2A 20	ld (tecl),a	
0347	034F CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0348	0351 C9	ret	
0349	0352 3E 04	cor20: ld a,04h	
0350	0354 32 2A 20	ld (tecl),a	
0351	0357 CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0352	0359 C9	ret	
0353	035A 3E 05	cor21: ld a,05h	
0354	035C 32 2A 20	ld (tecl),a	
0355	035F CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0356	0361 C9	ret	

0357	0362 3E 06	cor22:	ld a,06h	
0358	0364 32 2A 20		ld (tecl),a	
0359	0367 CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0360	0369 C9	ret		
0361	036A 3E 07	cor23:	ld a,07h	
0362	036C 32 2A 20		ld (tecl),a	
0363	036F CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0364	0371 C9	ret		
0365	0372 3E 08	cor30:	ld a,08h	
0366	0374 32 2A 20		ld (tecl),a	
0367	0377 CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0368	0379 C9	ret		
0369	037A 3E 09	cor31:	ld a,09h	
0370	037C 32 2A 20		ld (tecl),a	
0371	037F CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0372	0381 C9	ret		
0373	0382 3E 0A	cor32:	ld a,0Ah	
0374	0384 32 2A 20		ld (tecl),a	
0375	0387 CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0376	0389 C9	ret		
0377	038A 3E 0B	cor33:	ld a,0Bh	
0378	038C 32 2A 20		ld (tecl),a	
0379	038F CB C6	set 0,(hl)		;(bande) = xxxx xxx1
0380	0391 C9	ret		
0381	0392 3E 0C	cor40:	ld a,0Ch	
0382	0394 32 2A 20		ld (tecl),a	

0383	0397 CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0384	0399 C9	ret	
0385	039A 3E 0D	cor41: ld a,0Dh	
0386	039C 32 2A 20	ld (tecl),a	
0387	039F CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0388	03A1 C9	ret	
0389	03A2 3E 0E	cor42: ld a,0Eh	
0390	03A4 32 2A 20	ld (tecl),a	
0391	03A7 CB C6	set 0,(hl)	;(bande) = xxxx xxx1
0392	03A9 C9	ret	
0393	03AA 3E 0F	cor43: ld a,0Fh	
0394	03AC 32 2A 20	ld (tecl),a	
0395	03AF CB C6	set 0,(hl)	;(bande) = xxxx xxx1, es un dígito
0396	03B1 C9	ret	
0397	03B2 3E 10	cor50: ld a,10h	
0398	03B4 32 2A 20	ld (tecl),a	
0399	03B7 CB 86	res 0,(hl)	;(bande) = xxxx xxx0, es un comando
0400	03B9 C9	ret	
0401	03BA 3E 11	cor51: ld a,11h	
0402	03BC 32 2A 20	ld (tecl),a	
0403	03BF CB 86	res 0,(hl)	;(bande) = xxxx xxx0
0404	03C1 C9	ret	
0405	03C2 3E 12	cor52: ld a,12h	
0406	03C4 32 2A 20	ld (tecl),a	
0407	03C7 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
0408	03C9 C9	ret	

0409	03CA 3E 13	cor53:	ld a,13h	
0410	03CC 32 2A 20		ld (tecl),a	
0411	03CF CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0412	03D1 C9		ret	
0413	03D2 3E 14	cor60:	ld a,14h	
0414	03D4 32 2A 20		ld (tecl),a	
0415	03D7 CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0416	03D9 C9		ret	
0417	03DA 3E 15	cor61:	ld a,15h	
0418	03DC 32 2A 20		ld (tecl),a	
0419	03DF CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0420	03E1 C9		ret	
0421	03E2 3E 16	cor62:	ld a,16h	
0422	03E4 32 2A 20		ld (tecl),a	
0423	03E7 CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0424	03E9 C9		ret	
0425	03EA 3E 17	cor63:	ld a,17h	
0426	03EC 32 2A 20		ld (tecl),a	
0427	03EF CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0428	03F1 C9		ret	
0429	03F2 3E 18	cor70:	ld a,18h	
0430	03F4 32 2A 20		ld (tecl),a	
0431	03F7 CB 86	res 0,(hl)		;(bande) = xxxx xxx0
0432	03F9 C9		ret	
0433	03FA 3E 19	cor71:	ld a,19h	
0434	03FC 32 2A 20		ld (tecl),a	

0435	03FF CB 86	res 0,(hl)	;(bande) = xxxx xxx0
0436	0401 C9	ret	
0437	0402 3E 1A	cor72: ld a,1Ah	
0438	0404 32 2A 20	ld (tecl),a	
0439	0407 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
0440	0409 C9	ret	
0441	040A 3E 1B	cor73: ld a,1Bh	
0442	040C 32 2A 20	ld (tecl),a	
0443	040F CB 86	res 0,(hl)	;(bande) = xxxx xxx0, es un comando
0444	0411 C9	ret	
0445	0412	;=====	
0446	04A0	org 04a0h	
0447	04A0 F5	displ: push af	;para llegar a esta rutina debes haber
0448	04A1 C5	push bc	;guardado en las memorias adnhn a digte
0449	04A2 D5	push de	;los datos correspondientes en formato
0450	04A3 E5	push hl	;de segmentos por la rutina: ' '
0451	04A4 DD E5	push ix	
0452	04A6 CD 50 09	call agrupa	;reordena datos: addrl, addrh y data
0453	04A9 CD 30 0F	call flags	;estado de los 'flags' de estado
0454	04AC DD 21 10 20	ld ix,ledsl	;apunta a los valores convertidos de RAM
0455	04B0 16 06	ld d,06h	;contador de dígitos de ADDRESS
0456	04B2 0E 05	ld c,05h	;puerto PIO1-B de salida
0457	04B4 06 12	ld b,12h	;selecciona dígito 10 (S9= 12h)
0458	04B6 DD 7E 00	otro: ld a,(ix)	;toma el contenido 'son segmentos'
0459	04B9 CB 80	res 0,b	;bit0=0 latch del 4514
0460	04BB ED 41	out (c),b	;habilita el dígito N PIO1-B HCF4514

0461	04BD CD 10 0E	call delay1	
0462	04C0 CB C0	set 0,b	;bit0=1 pulso en latch del 4514
0463	04C2 ED 41	out (c),b	
0464	04C4 CD 10 0E	call delay1	
0465	04C7 CB 80	res 0,b	;bit0=0 latch del 4514
0466	04C9 ED 41	out (c),b	
0467	04CB CD 10 0E	call delay1	
0468	04CE 2F	cpl	;complementa a 1's segmentos
0469	04CF CB 87	res 0,a	;garantiza que el Bit0 de 'a' sea 0
0470	04D1 ED 79	out (c),a	;saca el código en segmentos del valor
0471	04D3 CD D0 0D	call retar1	;exhibe el dígito por un momento
0472	04D6 DD 23	inc ix	;nueva dirección: adhn1,adlnh y adln1=4 dig
0473	04D8 05	dec b	;hacia abajo, selecciona el siguiente dígito
0474	04D9 05	dec b	;dos veces ya que están de dos en dos.
0475	04DA 15	dec d	;cuenta los dígitos de ADDRESS
0476	04DB 20 D9	jr nz,otro	;continua mostrando RAM ADDRESSs
0477	04DD	;	
0478	04DD DD 21 17 20	ld ix,datnh	;ahora comienza con datnh
0479	04E1 06 04	ld b,04h	;desde D3(04h), D2(02h) y D1(00h)
0480	04E3 0E 05	ld c,05h	;puerto de salida PIO1-B
0481	04E5 21 1D 20	ld hl,bande	;prueba bit1 si es '0' o '1' para mostrarlo
0482	04E8 CB 4E	bit 1,(hl)	;el bit1 de bande, determina si muestra dig1
0483	04EA 20 05	jr nz,most	;si z=0, el bit=1, mostrar dig1 con d=03h
0484	04EC 16 02	ld d,02h	;si z=1, el bit=0, no mostrar con d=02h
0485	04EE C3 F3 04	jp otra	
0486	04F1 16 03	most: ld d,03h	;contador de dígitos DATA

0487	04F3 DD 7E 00	otra: ld a,(ix)	;y toma su contenido
0488	04F6 CB 80	res 0,b	;bit0=0 latch
0489	04F8 ED 41	out (c),b	;selecciona el Dígito a encender
0490	04FA CD 10 0E	call delay1	
0491	04FD CB C0	set 0,b	;bit0=1 latch abierto
0492	04FF ED 41	out (c),b	;sincroniza selección del dígito en 4514
0493	0501 CD 10 0E	call delay1	
0494	0504 CB 80	res 0,b	;bit0=0 latch cerrado
0495	0506 ED 41	out (c),b	
0496	0508 CD 10 0E	call delay1	
0497	050B 2F	cpl	;complementa Acc (son los segmentos)
0498	050C CB 87	res 0,a	;garantiza que el Bit0 de 'a' sea 0
0499	050E ED 79	out (c),a	;saca el valor de DATA
0500	0510 CD D0 0D	call retar1	
0501	0513 DD 23	inc ix	;siguiente dirección: datnl y digte
0502	0515 05	dec b	;disminuye la posición de D3,D2,D1 de
0503	0516 05	dec b	;dos en dos.
0504	0517 15	dec d	
0505	0518 20 D9	jr nz,otra	;aquí termina displ que muestra RAM y DATA
0506	051A DD E1	pop ix	
0507	051C E1	pop hl	
0508	051D D1	pop de	
0509	051E C1	pop bc	
0510	051F F1	pop af	
0511	0520 C9	ret	

0512 0521 ;===== Esta rutina convierte los valores HEX a
SEGMENTOS

0513 0560 org 0560h ;del display rojo.

0514 0560 3A 2D 20 convi: ld a,(temp) ;retoma el valor transportado

0515 0563 FE 00 cp 00h

0516 0565 CA B6 05 jp z,cero ;fue 0

0517 0568 FE 01 cp 01h

0518 056A CA BC 05 jp z,uno

0519 056D FE 02 cp 02h

0520 056F CA C2 05 jp z,dos

0521 0572 FE 03 cp 03h

0522 0574 CA C8 05 jp z,tres

0523 0577 FE 04 cp 04h

0524 0579 CA CE 05 jp z,cuat

0525 057C FE 05 cp 05h

0526 057E CA D4 05 jp z,cinc

0527 0581 FE 06 cp 06h

0528 0583 CA DA 05 jp z,seis

0529 0586 FE 07 cp 07h

0530 0588 CA E0 05 jp z,siet

0531 058B FE 08 cp 08h

0532 058D CA E6 05 jp z,ocho

0533 0590 FE 09 cp 09h

0534 0592 CA EC 05 jp z,nuev

0535 0595 FE 0A cp 0Ah

0536 0597 CA F2 05 jp z,esa

0537	059A FE 0B	cp 0Bh
0538	059C CA F8 05	jp z,esb
0539	059F FE 0C	cp 0Ch
0540	05A1 CA FE 05	jp z,esc
0541	05A4 FE 0D	cp 0Dh
0542	05A6 CA 04 06	jp z,esd
0543	05A9 FE 0E	cp 0Eh
0544	05AB CA 0A 06	jp z,ese
0545	05AE FE 0F	cp 0fh
0546	05B0 CA 10 06	jp z,esf
0547	05B3 C3 16 06	jp ningun
0548	05B6 3E FC	cero: ld a,0fch
0549	05B8 32 2D 20	ld (temp),a
0550	05BB C9	ret
0551	05BC 3E 60	uno: ld a,60h
0552	05BE 32 2D 20	ld (temp),a
0553	05C1 C9	ret
0554	05C2 3E DA	dos: ld a,0dah
0555	05C4 32 2D 20	ld (temp),a
0556	05C7 C9	ret
0557	05C8 3E F2	tres: ld a,0f2h
0558	05CA 32 2D 20	ld (temp),a
0559	05CD C9	ret
0560	05CE 3E 66	cuat: ld a,66h
0561	05D0 32 2D 20	ld (temp),a
0562	05D3 C9	ret

0563 05D4 3E B6 cinc: ld a,0b6h
0564 05D6 32 2D 20 ld (temp),a
0565 05D9 C9 ret
0566 05DA 3E BE seis: ld a,0beh
0567 05DC 32 2D 20 ld (temp),a
0568 05DF C9 ret
0569 05E0 3E E0 siet: ld a,0e0h
0570 05E2 32 2D 20 ld (temp),a
0571 05E5 C9 ret
0572 05E6 3E FE ocho: ld a,0feh
0573 05E8 32 2D 20 ld (temp),a
0574 05EB C9 ret
0575 05EC 3E E6 nuev: ld a,0e6h
0576 05EE 32 2D 20 ld (temp),a
0577 05F1 C9 ret
0578 05F2 3E EE esa: ld a,0eeh
0579 05F4 32 2D 20 ld (temp),a
0580 05F7 C9 ret
0581 05F8 3E 3E esb: ld a,3eh
0582 05FA 32 2D 20 ld (temp),a
0583 05FD C9 ret
0584 05FE 3E 9C esc: ld a,9ch
0585 0600 32 2D 20 ld (temp),a
0586 0603 C9 ret
0587 0604 3E 7A esd: ld a,7ah
0588 0606 32 2D 20 ld (temp),a

```

0589 0609 C9      ret
0590 060A 3E 9E   ese:      ld a,9eh
0591 060C 32 2D 20 ld (temp),a
0592 060F C9      ret
0593 0610 3E 8E   esf:      ld a,8eh
0594 0612 32 2D 20 ld (temp),a
0595 0615 C9      ret
0596 0616 3E 00   ningún: ld a,00h
0597 0618 32 2D 20 ld (temp),a
0598 061B C9      ret
0599 061C      ;=====
0600 0650      org      0650h
0601 0650 3A 2A 20 ejecut: ld a,(tecl)      ;verifica de que tecla se trata
0602 0653 FE 10   cp 10h      ;para determinar solamente comandos.
0603 0655 28 31   jr z,esizq      ;<---- Retrocede RAM
0604 0657 FE 11   cp 11h
0605 0659 28 29   jr z,esder      ;----> Avanza RAM
0606 065B FE 12   cp 12h
0607 065D 28 2D   jr z,esstp      ;STP, HLT
0608 065F FE 13   cp 13h
0609 0661 28 2D   jr z,esla      ;LA, Load Address o Puerto
0610 0663 FE 14   cp 14h
0611 0665 28 34   jr z,es2nd      ;Segunda Función
0612 0667 FE 15   cp 15h
0613 0669 28 31   jr z,esss
0614 066B FE 16   cp 16h

```

0615	066D 28 2E	jr z,esin	;IN, lee dato desde puerto
0616	066F FE 17	cp 17h	
0617	0671 28 2E	jr z,esld	;LD, Load
0618	0673 FE 18	cp 18h	
0619	0675 28 35	jr z,esars	;ARS
0620	0677 FE 19	cp 19h	
0621	0679 28 35	jr z,esgo	;GO, ejecuta desde dirección
0622	067B FE 1A	cp 1ah	
0623	067D 28 3C	jr z,esbrk	;BRK, /NMI
0624	067F FE 1B	cp 1bh	
0625	0681 28 39	jr z,esdp	;DP, Enter
0626	0683 C9	ret	;no es ningún comando, regresa.
0627	0684 CD 00 08	esder: call der	;llama a la rutina der que incrementa RAM
0628	0687 C9	ret	
0629	0688 CD 20 08	esizq: call izq	
0630	068B C9	ret	
0631	068C CD C0 06	esstp: call stp	;llama a la rutina que envía un HALT
0632	068F C9	ret	
0633	0690 2A 1D 20	esla: ld hl,(bande)	;prepara bande con XXXX 11XX
0634	0693 CB D6	set 2,(hl)	;pone bit2=1
0635	0695 CB DE	set 3,(hl)	;pone bit3=1
0636	0697 CD 00 07	call la	;rutina para cargar una sola dirección o
0637	069A C9	ret	;puerto de salida para datos 0000 DBB DP
0638	069B C9	es2nd: ret	
0639	069C C9	esss: ret	
0640	069D CD 90 12	esin: call input	

```

0641 06A0 C9      ret

0642 06A1 2A 1D 20 esld:  ld hl,(bande)  ;prepara bande con XXXX 01XX

0643 06A4 CB D6    set 2,(hl)              ;pone bit2=1

0644 06A6 CB 9E    res 3,(hl)              ;pone bit3=0

0645 06A8 CD 40 08 call ld              ;rutina para cargar direcciones seguidas

0646 06AB C9      ret

0647 06AC CD 90 0A esars:  call ars              ;rutina que introduce datos al Z80 y RAM

0648 06AF C9      ret                    ;por (A)sincronous (RS)232

0649 06B0 2A 1D 20 esgo:  ld hl,(bande)  ;prepara bande con XXXX 10XX

0650 06B3 CB 96    res 2,(hl)              ;pone bit2=0

0651 06B5 CB DE    set 3,(hl)              ;pone bit3=1

0652 06B7 CD A0 09 call goir            ;ejecuta lo que hay en 2100h

0653 06BA C9      ret

0654 06BB C9      esbrk:  ret

0655 06BC C9      esdp:   ret

0656 06BD        ;=====

0657 06C0        org     06c0h

0658 06C0 76      stp:    halt              ;provoca un HALT y espera una
interrupcion o

0659 06C1 C9      ret                    ;un reset.

0660 06C2        ;=====

0661 0700        org     0700h

0662 0700 F5      la:  push af              ;esta rutina permite seleccionar un puerto

0663 0701 C5      push bc              ;para salida de datos, por lo que se tecleara

0664 0702 D5      push de              ;el puerto valido de 00-ff y el dato DD + DP

0665 0703 E5      push hl

```

0666	0704 01 0A 0F	ld bc,0f0ah	;prepara el puerto PIO2-A para salidas
0667	0707 ED 41	out (c),b	
0668	0709 21 1D 20	ld hl,bande	;en esta rutina se muestra el dig1 por eso
0669	070C CB CE	set 1,(hl)	;se pone bit1=1 de bande
0670	070E 3E 00	ld a,00h	;con LA: desaparece todo el display y solo
0671	0710 32 12 20	ld (adnh),a	;se muestra un '0' en el dígito 1.
0672	0713 32 13 20	ld (adhl),a	
0673	0716 32 14 20	ld (adlh),a	
0674	0719 32 15 20	ld (adlnl),a	
0675	071C 32 17 20	ld (datnh),a	
0676	071F 32 18 20	ld (datnl),a	
0677	0722 3E 00	ld a,00h	;carga con '0' digi8 y digi7
0678	0724 32 22 20	ld (digi8),a	
0679	0727 32 23 20	ld (digi7),a	
0680	072A 3E FC	ld a,0fch	;FCh son los segmentos de '0'
0681	072C 32 19 20	ld (digte),a	;que será mostrado en el dígito1
0682	072F DD 21 14 20	ld ix,adlnh	;primera dirección de código de segmentos
0683	0733 FD 21 24 20	ld iy,digi6	;primera dirección de código de dígitos
0684	0737 16 05	ld d,05h	;se esperan 5 dígitos inclusive DP (Enter)
0685	0739 CD C0 01	busk: call scan	;sensa la presión de una tecla
0686	073C 3A 2C 20	ld a,(colum)	;verifica si se presionó una tecla
0687	073F FE 00	cp 00h	;compara con 0.
0688	0741 20 06	jr nz,sitek	;z=0, si hay tecla.La introduce en memoria
0689	0743 CD A0 04	call displ	;no tecla. Muestra el display sin cambios
0690	0746 C3 B8 07	jp bscla	;y regresa a sensar tecla
0691	0749 7A	sitek: ld a,d	

0692	074A 28 ED	jr z,busk	;verifica si d=0, es decir 'terminado'
0693	074C CD 60 02	call decod	;primero decodifica tecla numérica (00h-0Fh)
0694	074F 21 1D 20	r232a: ld hl,bande	;toma el valor de 'bande'
0695	0752 CB 46	bit 0,(hl)	;prueba bit0 de 'bande'
0696	0754 28 20	jr z,comma	;z=1 (bit=0), es comando
0697	0756 3A 2A 20	ld a,(tecl)	;z=0 (bit=1), es dígito
0698	0759 FD 77 00	ld (iy),a	;guarda dígito en su posición
0699	075C 32 29 20	ld (digi1),a	;y en dig1 siempre
0700	075F 32 2D 20	ld (temp),a	;para traspasar a rutina convertidora de
0701	0762 CD 60 05	call convi	;'segmentos'
0702	0765 3A 2D 20	ld a,(temp)	;
0703	0768 DD 77 00	ld (ix),a	;y guarda segmentos en adnhh-datnl
0704	076B 32 19 20	ld (digte),a	;y en el dígito1 como segmento
0705	076E DD 23	inc ix	;apunta al siguiente valor de segmentos
0706	0770 FD 23	inc iy	;apunta al siguiente valor de dígitos
0707	0772 15	dec d	;contador=contador-1
0708	0773 C3 39 07	jp busk	;espera el siguiente dígito del teclado
0709	0776 21 1D 20	comma: ld hl,bande	;toma el valor de 'bande'
0710	0779 CB 46	bit 0,(hl)	;prueba si es dígito o comando
0711	077B 28 06	jr z,sicmd	;z=1, es comando
0712	077D CD A0 04	call displ	;Z=0 no es dígito ni comando
0713	0780 C3 39 07	jp busk	
0714	0783 3A 2A 20	sicmd: ld a,(tecl)	;toma el valor del comando y verifica si
0715	0786 FE 1B	cp 1Bh	;es un comando y es 'Enter' o DP
0716	0788 28 0E	jr z,hacer	;si es 1Bh ejecuta 'Enter'
0717	078A FE 1A	cp 1ah	;es BRK

0718	078C 28 52	jr z,acab	;sale de esta rutina
0719	078E FE 10	cp 10h	;es <--
0720	0790 28 17	jr z,retr	;retrocede un caracter
0721	0792 CD A0 04	call displ	;si no es 'Enter' muestra todo como está
0722	0795 C3 39 07	jp busk	;y espera nueva tecla
0723	0798 CD 50 09	hacer: call agrupa	;agrupa los datos de ADD introducidos
0724	079B 3A 1C 20	ld a,(data)	
0725	079E 47	ld b,a	;prepara DATA
0726	079F 3A 1A 20	ld a,(addrl)	
0727	07A2 4F	ld c,a	;prepara el PUERTO
0728	07A3 ED 41	out (c),b	;saca el DATA por el PUERTO
0729	07A5 00	nop	;regresa de la subrutina del usuario
0730	07A6 C3 E0 07	jp acab	
0731	07A9 DD 2B	retr: dec ix	
0732	07AB FD 2B	dec iy	
0733	07AD 3E 00	ld a,00h	
0734	07AF DD 77 00	ld (ix),a	
0735	07B2 CD A0 04	call displ	
0736	07B5 C3 39 07	jp busk	
0737	07B8 DB 04	bscla: in a,(04h)	;lee dato de entrada
0738	07BA CB 67	bit 4,a	;prueba el bit4 de Acc (1 = Dato en Rs232)
0739	07BC 28 1F	jr z,nobat	;si z=1, el bit4=0 no hay dato, regresa
0740	07BE 00	nop	;si z=0, el bit4=1 si hay dato en RS232
0741	07BF DB 09	in a,(09h)	;introduce el dato en Acc
0742	07C1 D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
0743	07C3 32 1F 20	ld (serdat),a	;guarda dato en 'serdat'

0744	07C6 CD A0 0B	call hexa	;convierte de ASCII a hex: serdat es hex
0745	07C9 3E 01	ld a,01h	
0746	07CB D3 80	out (80h),a	;avisa a PIC dato leído
0747	07CD CD D0 0D	call retar1	
0748	07D0 3E 00	ld a,00h	
0749	07D2 D3 80	out (80h),a	
0750	07D4 3A 1F 20	ld a,(serdat)	;toma el dato convertido a Hex
0751	07D7 32 2A 20	ld (tecl),a	;y lo guarda en "tecl" para proceder
0752	07DA C3 4F 07	jp r232a	;regresa a la rutina
0753	07DD C3 39 07	nobat: jp busk	
0754	07E0 E1	acab: pop hl	
0755	07E1 D1	pop de	
0756	07E2 C1	pop bc	
0757	07E3 F1	pop af	
0758	07E4 C9	ret	
0759	07E5	;=====	
0760	0800	org	0800h
0761	0800 F5	der: push af	
0762	0801 C5	push bc	
0763	0802 D5	push de	
0764	0803 E5	push hl	
0765	0804 CD 30 0E	call desag	;desagrupa para tener addrh,addrl y data
0766	0807 2A 1A 20	ld hl,(addrl)	;incrementa la RAM
0767	080A 23	inc hl	
0768	080B 22 1A 20	ld (addrl),hl	;devuelve el valor incrementado
0769	080E CD 30 0E	call desag	

0770	0811 E1	pop hl	
0771	0812 D1	pop de	
0772	0813 C1	pop bc	
0773	0814 F1	pop af	
0774	0815 C9	ret	
0775	0816	;=====	
0776	0820	org	0820h
0777	0820 F5	izq: push af	
0778	0821 C5	push bc	
0779	0822 D5	push de	
0780	0823 E5	push hl	
0781	0824 CD 30 0E	call desag	;desagrupa para tener addrh,addrl y data
0782	0827 2A 1A 20	ld hl,(addrl)	;decrementa la RAM
0783	082A 2B	dec hl	
0784	082B 22 1A 20	ld (addrl),hl	;devuelve el valor decrementado
0785	082E CD 30 0E	call desag	
0786	0831 E1	pop hl	
0787	0832 D1	pop de	
0788	0833 C1	pop bc	
0789	0834 F1	pop af	
0790	0835 C9	ret	
0791	0836	;=====	
0792	0840	org	0840h
0793	0840 F5	ld: push af	;rutina que carga una localidad con un valor
0794	0841 C5	push bc	;el usuario introduce: LD 0000 D00 DP
0795	0842 D5	push de	;para introducir correctamente el dato

0796	0843 E5	push hl	
0797	0844 21 1D 20	ld hl,bande	;en esta rutina se muestra el dig1 por eso
0798	0847 CB CE	set 1,(hl)	;se pone bit1=1 de bande
0799	0849 3E 00	ld a,00h	;con LD: desaparece todo el display y solo
0800	084B 32 12 20	ld (adnh),a	;se muestra un '0' en el dígito 1.
0801	084E 32 13 20	ld (adhl),a	
0802	0851 32 14 20	ld (adlh),a	
0803	0854 32 15 20	ld (adhl),a	
0804	0857 32 17 20	ld (datnh),a	
0805	085A 32 18 20	ld (datnl),a	
0806	085D 3E FC	ld a,0fch	;0Fh son los segmentos de '0'
0807	085F 32 19 20	ld (digte),a ;	que será mostrado en el dígito1
0808	0862 DD 21 12 20	ld ix,adnh	;primera dirección de código de segmentos
0809	0866 FD 21 22 20	ld iy,digi8	;primera dirección de código de dígitos
0810	086A 16 08	ld d,08h	;cuenta dígitos introducidos inclusive ENTER
0811	086C C3 79 08	jp busca	
0812	086F DD 21 17 20	poste: ld ix,datnh	;primer dato en código de segmentos
0813	0873 FD 21 27 20	ld iy,digi3	;primer dato en código de dígitos
0814	0877 16 03	ld d,03h	;cuenta dígitos introducidos inclusive ENTER
0815	0879 CD C0 01	busca: call scan	;sensa la presión de una tecla
0816	087C 3A 2C 20	ld a,(colum)	;verifica si se presionó una tecla
0817	087F FE 00	cp 00h	;compara con 0.
0818	0881 20 06	jr nz,sitec	z=0, si hay tecla.La introduce en memoria
0819	0883 CD A0 04	call displ	;no tecla. Muestra el display sin cambios
0820	0886 C3 FE 08	jp bscb	;y regresa a sensar tecla
0821	0889 7A	sitec: ld a,d	

0822	088A 28 ED	jr z,busca	
0823	088C CD 60 02	call decod	;primero decodifica tecla numérica (00h-0Fh)
0824	088F 21 1D 20	r232b: ld hl,bande	;toma el valor de 'bande'
0825	0892 CB 46	bit 0,(hl)	;prueba bit0 de 'bande'
0826	0894 28 20	jr z,coman	;z=1 (bit=0), es comando
0827	0896 3A 2A 20	ld a,(tecl)	;z=0 (bit=1), es dígito
0828	0899 FD 77 00	ld (iy),a	;guarda dígito en su posición
0829	089C 32 29 20	ld (digi1),a	;y en dig1 siempre
0830	089F 32 2D 20	ld (temp),a	;para traspasar a rutina convertidora de
0831	08A2 CD 60 05	call convi	;'segmentos'
0832	08A5 3A 2D 20	ld a,(temp)	;
0833	08A8 DD 77 00	ld (ix),a	;y guarda segmentos en adnhh-datnl
0834	08AB 32 19 20	ld (digte),a	;y en el dígito1 como segmento
0835	08AE DD 23	inc ix	;apunta al siguiente valor de segmentos
0836	08B0 FD 23	inc iy	;apunta al siguiente valor de dígitos
0837	08B2 15	dec d	;contador=contador-1
0838	08B3 C3 79 08	jp busca	;espera el siguiente dígito del teclado
0839	08B6 21 1D 20	coman: ld hl,bande	;toma el valor de 'bande'
0840	08B9 CB 46	bit 0,(hl)	;prueba si es dígito o comando
0841	08BB 28 06	jr z,sicom	;z=1, es comando
0842	08BD CD A0 04	call displ	;Z=0 no es dígito ni comando
0843	08C0 C3 79 08	jp busca	
0844	08C3 3A 2A 20	sicom: ld a,(tecl)	;toma el valor del comando y verifica si
0845	08C6 FE 1B	cp 1bh	;es un comando y es 'Enter'
0846	08C8 28 0E	jr z,hazlo	;si es 1Bh ejecuta 'Enter'
0847	08CA FE 1A	cp 1ah	;es un comando y es 'BRK'

0848	08CC 28 58	jr z,acaba	;si es BRK sale de la rutina
0849	08CE FE 10	cp 10h	;si es 10h ejecuta retro(ceso)
0850	08D0 18 1D	jr retro	
0851	08D2 CD A0 04	call displ	;si no es 'Enter' muestra todo como está
0852	08D5 C3 79 08	jp busca	;y espera nueva tecla
0853	08D8 CD 50 09	hazlo: call agrupa	;si es 'Enter' guarda dato en RAM elegida
0854	08DB 2A 1A 20	ld hl,(addrl)	;toma el valor agrupado de RAM
0855	08DE 3A 1C 20	ld a,(data)	;toma el DATO tecleado agrupado
0856	08E1 77	ld (hl),a	;guarda en HL el valor del dato
0857	08E2		;
0858	08E2 2A 1A 20	ld hl,(addrl)	;selecciona RAM
0859	08E5 23	inc hl	
0860	08E6 22 1A 20	ld (addrl),hl	;devuelve el valor incrementado
0861	08E9 CD 30 0E	call desag	;desagrupa, incluye 'convi' para cada dígito
0862	08EC C3 6F 08	jp poste	;recomienza en poste(riores)
0863	08EF DD 2B	retro: dec ix	;retrocede el apuntador de segmentos
0864	08F1 FD 2B	dec iy	;retrocede el apuntador de dígitos
0865	08F3 3E 00	ld a,00h	;borra el contenido de esa dirección
0866	08F5 DD 77 00	ld (ix),a	
0867	08F8 CD A0 04	call displ	
0868	08FB C3 79 08	jp busca	
0869	08FE DB 04	bscb: in a,(04h)	;lee dato de entrada
0870	0900 CB 67	bit 4,a	;prueba el bit4 de Acc (1 = Dato en Rs232)
0871	0902 28 1F	jr z,nobct	;si z=1, el bit4=0 no hay dato, regresa
0872	0904 00	nop	;si z=0, el bit4=1 si hay dato en RS232
0873	0905 DB 09	in a,(09h)	;introduce el dato en Acc

0874	0907 D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
0875	0909 32 1F 20	ld (serdat),a	;guarda dato en 'serdat'
0876	090C CD A0 0B	call hexa	;convierte de ASCII a hex: serdat es hex
0877	090F 3E 01	ld a,01h	
0878	0911 D3 80	out (80h),a	;avisa a PIC dato leído
0879	0913 CD D0 0D	call retar1	
0880	0916 3E 00	ld a,00h	
0881	0918 D3 80	out (80h),a	
0882	091A 3A 1F 20	ld a,(serdat)	;toma el dato convertido a Hex
0883	091D 32 2A 20	ld (tecl),a	;y lo guarda en "tecl" para proceder
0884	0920 C3 8F 08	jp r232b	;regresa a la rutina
0885	0923 C3 79 08	nobct: jp busca	
0886	0926 E1	acaba: pop hl	
0887	0927 D1	pop de	
0888	0928 C1	pop bc	
0889	0929 F1	pop af	
0890	092A C9	ret	
0891	092B	;=====	
0892	0950	org 0950h	
0893	0950 F5	agrupa: push af	;esta rutina junta el contenido de digi8-
0894	0951 C5	push bc	;digi7 y lo coloca en addrh. Luego digi6-5
0895	0952 D5	push de	;y lo coloca en addrl. Lo mismo para digi3-2
0896	0953 E5	push hl	;para colocarlo en "data".
0897	0954 3A 22 20	ld a,(digi8)	;rutina que reagrupa las direcciones
0898	0957 CB 27	sla a	;recorre a la izquierda 4 posiciones
0899	0959 CB 27	sla a	

0900	095B CB 27	sla a	
0901	095D CB 27	sla a	
0902	095F 21 23 20	ld hl,digi7	;apunta con hl la dirección de adhnl
0903	0962 86	add a,(hl)	;súmalos
0904	0963 32 1B 20	ld (addrh),a	;guarda el resultado en addrh
0905	0966	;	
0906	0966 3A 24 20	ld a,(digi6)	;carga el nibble bajo de dirección
0907	0969 CB 27	sla a	
0908	096B CB 27	sla a	
0909	096D CB 27	sla a	
0910	096F CB 27	sla a	
0911	0971 21 25 20	ld hl,digi5	;apunta con hl la dirección de adhnl
0912	0974 86	add a,(hl)	;súmalos
0913	0975 32 1A 20	ld (addrl),a	;guarda el resultado en addrl
0914	0978	;	
0915	0978 3A 27 20	ld a,(digi3)	;carga el nibble alto de dato
0916	097B CB 27	sla a	
0917	097D CB 27	sla a	
0918	097F CB 27	sla a	
0919	0981 CB 27	sla a	
0920	0983 21 28 20	ld hl,digi2	;apunta con hl la dirección de datnl
0921	0986 86	add a,(hl)	;súmalos
0922	0987 32 1C 20	ld (data),a	;guarda el resultado en addrl
0923	098A E1	pop hl	
0924	098B D1	pop de	
0925	098C C1	pop bc	

0926	098D F1	pop af	
0927	098E C9	ret	
0928	098F	;=====	
0929	09A0	org	09a0h
0930	09A0 F5	goir: push af	;rutina GO que ejecuta un programa montado
0931	09A1 C5	push bc	
0932	09A2 D5	push de	
0933	09A3 E5	push hl	
0934	09A4 21 1D 20	ld hl,bande	;en esta rutina se muestra el dig1 por eso
0935	09A7 CB CE	set 1,(hl)	;se pone bit1=1 de bande
0936	09A9 3E 00	ld a,00h	;con GO: desaparece todo el display y solo
0937	09AB 32 12 20	ld (adnhn),a	;se muestra un '0' en el dígito 1.
0938	09AE 32 13 20	ld (adhnI),a	
0939	09B1 32 14 20	ld (adlnh),a	
0940	09B4 32 15 20	ld (adlnI),a	
0941	09B7 32 17 20	ld (datnh),a	
0942	09BA 32 18 20	ld (datnI),a	
0943	09BD 3E FC	ld a,0fch	;0Fh son los segmentos de '0'
0944	09BF 32 19 20	ld (digte),a	;que será mostrado en el dígito1
0945	09C2 DD 21 12 20	ld ix,adnhn	;primera dirección de código de segmentos
0946	09C6 FD 21 22 20	ld iy,digi8	;primera dirección de código de dígitos
0947	09CA 16 05	ld d,05h	;se esperan 5 dígitos inclusive DP (Enter)
0948	09CC CD C0 01	busc: call scan	;sensa la presión de una tecla
0949	09CF 3A 2C 20	ld a,(colum)	;verifica si se presionó una tecla
0950	09D2 FE 00	cp 00h	;compara con 0.
0951	09D4 20 06	jr nz,site	;z=0, si hay tecla. La introduce en memoria

0952	09D6 CD A0 04	call displ	;no tecla. Muestra el display sin cambios
0953	09D9 C3 47 0A	jp bscd	;y regresa a sensar tecla
0954	09DC 7A	site: ld a,d	
0955	09DD 28 68	jr z,bscd	
0956	09DF CD 60 02	call decod	;primero decodifica tecla numérica (00h-0Fh)
0957	09E2 21 1D 20	r232d: ld hl,bande	;toma el valor de 'bande'
0958	09E5 CB 46	bit 0,(hl)	;prueba bit0 de 'bande'
0959	09E7 28 20	jr z,comand	;z=1 (bit=0), es comando
0960	09E9 3A 2A 20	ld a,(tecl)	;z=0 (bit=1), es dígito
0961	09EC FD 77 00	ld (iy),a	;guarda dígito en su posición
0962	09EF 32 29 20	ld (digi1),a	;y en dig1 siempre
0963	09F2 32 2D 20	ld (temp),a	;para traspasar a rutina convertidora de
0964	09F5 CD 60 05	call convi	;'segmentos'
0965	09F8 3A 2D 20	ld a,(temp)	;
0966	09FB DD 77 00	ld (ix),a	;y guarda segmentos en adnhh-datnl
0967	09FE 32 19 20	ld (digte),a	;y en el dígito1 como segmento
0968	0A01 DD 23	inc ix	;apunta al siguiente valor de segmentos
0969	0A03 FD 23	inc iy	;apunta al siguiente valor de dígitos
0970	0A05 15	dec d	;contador=contador-1
0971	0A06 C3 CC 09	jp busc	;espera el siguiente dígito del teclado
0972	0A09 21 1D 20	comand: ld hl,bande	;toma el valor de 'bande'
0973	0A0C CB 46	bit 0,(hl)	;prueba si es dígito o comando
0974	0A0E 28 06	jr z,sico	;z=1, es comando
0975	0A10 CD A0 04	call displ	;Z=0 no es dígito ni comando
0976	0A13 C3 CC 09	jp busc	
0977	0A16 3A 2A 20	sico: ld a,(tecl)	;toma el valor del comando y verifica si

0978	0A19 FE 1B	cp 1Bh	;es un comando y es 'Enter'
0979	0A1B 28 0E	jr z,haz	;si es 1Bh ejecuta 'Enter'
0980	0A1D FE 1A	cp 1ah	;es 'BRK' sale de esta rutina
0981	0A1F 28 4E	jr z,acabo	
0982	0A21 FE 10	cp 10h	;es <--
0983	0A23 28 13	jr z,retra	;retrocede un caracter
0984	0A25 CD A0 04	call displ	;si no es 'Enter' muestra todo como está
0985	0A28 C3 CC 09	jp busc	;y espera nueva tecla
0986	0A2B CD 50 09	haz: call agrupa	;agrupa los datos de ADD introducidos
0987	0A2E CD 00 21	call 2100h	;salta a rutina escrita por el usuario
0988	0A31 00	nop	;regresa de la subrutina del usuario
0989	0A32 CD A0 04	call displ	
0990	0A35 C3 6F 0A	jp acabo	
0991	0A38 DD 2B	retra: dec ix	
0992	0A3A FD 2B	dec iy	
0993	0A3C 3E 00	ld a,00h	
0994	0A3E DD 77 00	ld (ix),a	
0995	0A41 CD A0 04	call displ	
0996	0A44 C3 CC 09	jp busc	
0997	0A47 DB 04	bscd: in a,(04h)	;lee dato de entrada
0998	0A49 CB 67	bit 4,a	;prueba el bit4 de Acc (1 = Dato en Rs232)
0999	0A4B 28 1F	jr z,nobdt	;si z=1, el bit4=0 no hay dato, regresa
1000	0A4D 00	nop	;si z=0, el bit4=1 si hay dato en RS232
1001	0A4E DB 09	in a,(09h)	;introduce el dato en Acc
1002	0A50 D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
1003	0A52 32 1F 20	ld (serdat),a	;guarda dato en 'serdat'

1004	0A55 CD A0 0B	call hexa		;convierte de ASCII a hex: serdat es hex
1005	0A58 3E 01	ld a,01h		
1006	0A5A D3 80	out (80h),a		;avisa a PIC dato leído
1007	0A5C CD D0 0D	call retar1		
1008	0A5F 3E 00	ld a,00h		
1009	0A61 D3 80	out (80h),a		
1010	0A63 3A 1F 20	ld a,(serdat)		;toma el dato convertido a Hex
1011	0A66 32 2A 20	ld (tecl),a		;y lo guarda en "tecl" para proceder
1012	0A69 C3 E2 09	jp r232d		;regresa a la rutina
1013	0A6C C3 CC 09	nobdt: jp busc		
1014	0A6F			
1015	0A6F E1	acabo: pop hl		
1016	0A70 D1	pop de		
1017	0A71 C1	pop bc		
1018	0A72 F1	pop af		
1019	0A73 C9	ret		
1020	0A74	;=====		
1021	0A90	org	0a90h	
1022	0A90 F5	ars: push af		;rutina que carga en la RAM los datos
1023	0A91 C5	push bc		;desde el puerto RS232 de una PC a través
1024	0A92 D5	push de		;del PIC16F628A iniciando desde la 2100h
1025	0A93 E5	push hl		;ARS tiene el código 18h y provoca llegar aquí.
1026	0A94 00	nop		
1027	0A95 DD 21 00 21	ld ix,2100h		;inicializa el apuntador en 2100h de RAM
1028	0A99 00	nop		;apuntador iniciado
1029	0A9A DB 04	noeof: in a,(04h)		;sensa dato listo en PIC16F628A

1030	0A9C CB 67	bit 4,a	;prueba el bit4 de Acc (1 = Dato en Rs232)
1031	0A9E 28 FA	jr z,noeof	;si z=1, el bit4=0 no hay dato, regresa
1032	0AA0 00	nop	;si z=0, el bit4=1 si hay dato en RS232
1033	0AA1 DB 09	in a,(09h)	;introduce el dato en Acc
1034	0AA3 D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
1035	0AA5 32 1F 20	ld (serdat),a	;lo almacena en RAM para protección
1036	0AA8 3E 01	ld a,01h	
1037	0AAA D3 80	out (80h),a	;avisa a PIC dato leído
1038	0AAC CD D0 0D	call retar1	;retardo para el pulso de activación
1039	0AAF 3E 00	ld a,00h	
1040	0AB1 D3 80	out (80h),a	
1041	0AB3 3A 1F 20	ld a,(serdat)	;recupera valor de Acc
1042	0AB6 DD 77 00	ld (ix+00h),a	;lo guarda en la localidad inicial y subsecuentes
1043	0AB9 DD 23	inc ix	;incrementa localidad RAM
1044	0ABB D6 5A	sub 5ah	;subtrae EOF de ACC para ver si es el fin (Z=5a)
1045	0ABD 20 DB	jr nz,noeof	;del archivo. Si no es EOF, captura otro dato
1046	0ABF 00	nop	;si es EOF procede a convertir de ASCII a HEX
1047	0AC0 DD 21 00 21	ld ix,2100h	;inicializa IX con 2100h
1048	0AC4 DD 7E 00	otrda: ld a,(ix+00h)	;carga el primer dato y subsecuentes
1049	0AC7 32 1F 20	ld (serdat),a	;prepara conversión
1050	0ACA CD A0 0B	call hexa	;convierte de ASCII a HEX
1051	0ACD 3A 1F 20	ld a,(serdat)	;toma valor convertido y
1052	0AD0 DD 77 00	ld (ix+00h),a	;lo guarda de donde lo tomó
1053	0AD3 DD 23	inc ix	;incrementa ix
1054	0AD5 D6 5A	sub 5ah	;compara con EOF(hex) para ver si es el fin
1055	0AD7 20 EB	jr nz,otrda	;si no es el fin, convierte el siguiente dato

1056	0AD9 00	nop		;termina la captura y conversión regresa
1057	0ADA E1	pop hl		
1058	0ADB D1	pop de		
1059	0ADC C1	pop bc		
1060	0ADD F1	pop af		
1061	0ADE C9	ret		
1062	0ADF		;=====	
1063	0AF0	org	0af0h	
1064	0AF0 C3 90 12	jp	1290h	
1065	0AF3		;=====	
1066	0BA0	org	0ba0h	
1067	0BA0 F5	hexa: push af		;Rutina hexa que convierte de ASCII a HEX
1068	0BA1 C5	push bc		;el dato llega en 'serdat' y regresa en el
1069	0BA2 D5	push de		;mismo 'serdat'
1070	0BA3 E5	push hl		
1071	0BA4 21 1D 20	ld hl,bande		
1072	0BA7 3A 1F 20	ld a,(serdat)		;toma el dato en ASCII
1073	0BAA FE 30	cp 30h		;es 0 ?
1074	0BAC CA 48 0C	jp z,cer0		
1075	0BAF FE 31	cp 31h		
1076	0BB1 CA 52 0C	jp z,uno1		
1077	0BB4 FE 32	cp 32h		
1078	0BB6 CA 5C 0C	jp z,dos2		
1079	0BB9 FE 33	cp 33h		
1080	0BBB CA 66 0C	jp z,tres3		
1081	0BBE FE 34	cp 34h		

1082	OBC0 CA 70 0C	jp z,cuat4	
1083	OBC3 FE 35	cp 35h	
1084	OBC5 CA 7A 0C	jp z,cinc5	
1085	OBC8 FE 36	cp 36h	
1086	OBCA CA 84 0C	jp z,seis6	
1087	OBCD FE 37	cp 37h	
1088	OBCF CA 8E 0C	jp z,siet7	
1089	OBD2 FE 38	cp 38h	
1090	OBD4 CA 98 0C	jp z,ocho8	
1091	OBD7 FE 39	cp 39h	
1092	OBD9 CA A2 0C	jp z,nuev9	
1093	OBDC FE 61	cp 61h	
1094	OBDE CA AC 0C	jp z,ahex	
1095	OBE1 FE 62	cp 62h	
1096	OBE3 CA B6 0C	jp z,bhex	
1097	OBE6 FE 63	cp 63h	
1098	OBE8 CA C0 0C	jp z,chex	
1099	OBEB FE 64	cp 64h	
1100	OBED CA CA 0C	jp z,dhex	
1101	OBF0 FE 65	cp 65h	
1102	OBF2 CA D4 0C	jp z,ehex	
1103	OBF5 FE 66	cp 66h	
1104	OBF7 CA DE 0C	jp z,fhex	
1105	OBFA FE 0D	cp 0dh	;cr
1106	OBFC CA E8 0C	jp z,crhex	
1107	OBFF FE 0B	cp 0bh	;lf

1108	0C01 CA F2 0C	jp z,lfhex	
1109	0C04 FE 1B	cp 1bh	
1110	0C06 CA FC 0C	jp z,brkhex	
1111	0C09 FE 3C	cp 3ch	;<----- Retrocede
1112	0C0B CA 06 0D	jp z,ret hex	
1113	0C0E FE 3E	cp 3eh	;-----> Avanza
1114	0C10 CA 10 0D	jp z,ava hex	
1115	0C13 FE 41	cp 41h	
1116	0C15 CA 1A 0D	jp z,ars hex	
1117	0C18 FE 42	cp 42h	
1118	0C1A CA 24 0D	jp z,bk1hex	
1119	0C1D FE 44	cp 44h	
1120	0C1F CA 2E 0D	jp z,dphex	
1121	0C22 FE 47	cp 47h	
1122	0C24 CA 38 0D	jp z,gohex	
1123	0C27 FE 49	cp 49h	
1124	0C29 CA 42 0D	jp z,inchex	;IN (input) código 49 ASCII "I"
1125	0C2C FE 4B	cp 4bh	
1126	0C2E CA 4C 0D	jp z,lahex	
1127	0C31 FE 4C	cp 4ch	
1128	0C33 CA 56 0D	jp z,ldhex	
1129	0C36 FE 4E	cp 4eh	
1130	0C38 CA 60 0D	jp z,sndhex	
1131	0C3B FE 53	cp 53h	
1132	0C3D CA 6A 0D	jp z,sshex	
1133	0C40 FE 48	cp 48h	

```

1134 0C42 CA 74 0D  jp z,sthex
1135 0C45 C3 7E 0D  jp fine
1136 0C48                                     ;
1137 0C48 3E 00  cer0:  ld a,00h
1138 0C4A 32 1F 20  ld (serdat),a
1139 0C4D CB C6  set 0,(hl)                ;(bande) = xxxx xxx1 DATOS
1140 0C4F C3 7E 0D  jp fine
1141 0C52 3E 01  uno1:  ld a,01h
1142 0C54 32 1F 20  ld (serdat),a
1143 0C57 CB C6  set 0,(hl)                ;(bande) = xxxx xxx1
1144 0C59 C3 7E 0D  jp fine
1145 0C5C 3E 02  dos2:  ld a,02h
1146 0C5E 32 1F 20  ld (serdat),a
1147 0C61 CB C6  set 0,(hl)                ;(bande) = xxxx xxx1
1148 0C63 C3 7E 0D  jp fine
1149 0C66 3E 03  tres3:  ld a,03h
1150 0C68 32 1F 20  ld (serdat),a
1151 0C6B CB C6  set 0,(hl)                ;(bande) = xxxx xxx1
1152 0C6D C3 7E 0D  jp fine
1153 0C70 3E 04  cuat4:  ld a,04h
1154 0C72 32 1F 20  ld (serdat),a
1155 0C75 CB C6  set 0,(hl)                ;(bande) = xxxx xxx1
1156 0C77 C3 7E 0D  jp fine
1157 0C7A 3E 05  cinc5:  ld a,05h
1158 0C7C 32 1F 20  ld (serdat),a
1159 0C7F CB C6  set 0,(hl)                ;(bande) = xxxx xxx1

```

```

1160 0C81 C3 7E 0D  jp fine
1161 0C84 3E 06  seis6:  ld a,06h
1162 0C86 32 1F 20  ld (serdat),a
1163 0C89 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1164 0C8B C3 7E 0D  jp fine
1165 0C8E 3E 07  siet7:  ld a,07h
1166 0C90 32 1F 20  ld (serdat),a
1167 0C93 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1168 0C95 C3 7E 0D  jp fine
1169 0C98 3E 08  ocho8:  ld a,08h
1170 0C9A 32 1F 20  ld (serdat),a
1171 0C9D CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1172 0C9F C3 7E 0D  jp fine
1173 0CA2 3E 09  nuev9:  ld a,09h
1174 0CA4 32 1F 20  ld (serdat),a
1175 0CA7 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1176 0CA9 C3 7E 0D  jp fine
1177 0CAC 3E 0A  ahex:  ld a,0ah
1178 0CAE 32 1F 20  ld (serdat),a
1179 0CB1 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1180 0CB3 C3 7E 0D  jp fine
1181 0CB6 3E 0B  bhex:  ld a,0bh
1182 0CB8 32 1F 20  ld (serdat),a
1183 0CBB CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1184 0CBD C3 7E 0D  jp fine
1185 0CC0 3E 0C  chex:  ld a,0ch

```

```

1186  OCC2 32 1F 20  ld (serdat),a
1187  OCC5 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1188  OCC7 C3 7E 0D  jp fine
1189  OCCA 3E 0D      dhex:  ld a,0dh
1190  OCCC 32 1F 20  ld (serdat),a
1191  OCCF CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1192  OCD1 C3 7E 0D  jp fine
1193  OCD4 3E 0E      ehex:  ld a,0eh
1194  OCD6 32 1F 20  ld (serdat),a
1195  OCD9 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1196  OCDB C3 7E 0D  jp fine
1197  OCDE 3E 0F      fhex:  ld a,0fh
1198  OCE0 32 1F 20  ld (serdat),a
1199  OCE3 CB C6      set 0,(hl)          ;(bande) = xxxx xxx1
1200  OCE5 C3 7E 0D  jp fine
1201  OCE8 3E 1B      crhex: ld a,1bh      ;DP o 'Enter'
1202  OCEA 32 1F 20  ld (serdat),a
1203  OCED CB 86      res 0,(hl)          ;(bande) = xxxx xxx0 COMANDOS
1204  OCEF C3 7E 0D  jp fine
1205  OCF2 3E 1B      lfhex: ld a,1bh      ;DP o 'Enter'
1206  OCF4 32 1F 20  ld (serdat),a
1207  OCF7 CB 86      res 0,(hl)          ;(bande) = xxxx xxx0
1208  OCF9 C3 7E 0D  jp fine
1209  OCFC 3E 1A      brkhex: ld a,1ah      ;BRK
1210  OCFE 32 1F 20  ld (serdat),a
1211  OD01 CB 86      res 0,(hl)          ;(bande) = xxxx xxx0

```

```

1212 0D03 C3 7E 0D  jp fine

1213 0D06 3E 10    rethex:  ld a,10h

1214 0D08 32 1F 20    ld (serdat),a

1215 0D0B CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1216 0D0D C3 7E 0D  jp fine

1217 0D10 3E 11    avahex:  ld a,11h

1218 0D12 32 1F 20    ld (serdat),a

1219 0D15 CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1220 0D17 C3 7E 0D  jp fine

1221 0D1A 3E 18    arshex:  ld a,18h

1222 0D1C 32 1F 20    ld (serdat),a

1223 0D1F CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1224 0D21 C3 7E 0D  jp fine

1225 0D24 3E 1A    bk1hex:  ld a,1ah

1226 0D26 32 1F 20    ld (serdat),a

1227 0D29 CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1228 0D2B C3 7E 0D  jp fine

1229 0D2E 3E 1B    dphehex:  ld a,1bh

1230 0D30 32 1F 20    ld (serdat),a

1231 0D33 CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1232 0D35 C3 7E 0D  jp fine

1233 0D38 3E 19    gohex:   ld a,19h

1234 0D3A 32 1F 20    ld (serdat),a

1235 0D3D CB 86      res 0,(hl)                ;(bande) = xxxx xxx0

1236 0D3F C3 7E 0D  jp fine

1237 0D42 3E 16    inchex:  ld a,16h                ;código comando para IN (input) = 16 HEX

```

1238	0D44 32 1F 20	ld (serdat),a	
1239	0D47 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1240	0D49 C3 7E 0D	jp fine	
1241	0D4C 3E 13	lahex: ld a,13h	
1242	0D4E 32 1F 20	ld (serdat),a	
1243	0D51 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1244	0D53 C3 7E 0D	jp fine	
1245	0D56 3E 17	ldhex: ld a,17h	
1246	0D58 32 1F 20	ld (serdat),a	
1247	0D5B CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1248	0D5D C3 7E 0D	jp fine	
1249	0D60 3E 14	sndhex: ld a,14h	
1250	0D62 32 1F 20	ld (serdat),a	
1251	0D65 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1252	0D67 C3 7E 0D	jp fine	
1253	0D6A 3E 15	sshex: ld a,15h	
1254	0D6C 32 1F 20	ld (serdat),a	
1255	0D6F CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1256	0D71 C3 7E 0D	jp fine	
1257	0D74 3E 12	sthex: ld a,12h	
1258	0D76 32 1F 20	ld (serdat),a	
1259	0D79 CB 86	res 0,(hl)	;(bande) = xxxx xxx0
1260	0D7B C3 7E 0D	jp fine	
1261	0D7E E1	fine: pop hl	
1262	0D7F D1	pop de	
1263	0D80 C1	pop bc	

1264	0D81 F1	pop af	
1265	0D82 C9	ret	
1266	0D83	;=====	
1267	0DD0	org	Odd0h
1268	0DD0 F5	retar1: push af	;Rutina retar1 con un tiempo largo
1269	0DD1 C5	push bc	
1270	0DD2 D5	push de	
1271	0DD3 06 0F	ld b,0fh	
1272	0DD5 0E 08	aci: ld c,08h	
1273	0DD7 16 04	aqi: ld d,04h	
1274	0DD9 15	aki: dec d	
1275	0DDA 20 FD	jr nz,aki	
1276	0DDC 0D	dec c	
1277	0DDD 20 F8	jr nz,aqi	
1278	0DDF 05	dec b	
1279	0DE0 20 F3	jr nz,aci	
1280	0DE2 D1	pop de	
1281	0DE3 C1	pop bc	
1282	0DE4 F1	pop af	
1283	0DE5 C9	ret	
1284	0DE6	;=====	
1285	0DF0	org	0df0h
1286	0DF0 F5	retar2: push af	;Rutina retar2 con un tiempo largo
1287	0DF1 C5	push bc	
1288	0DF2 D5	push de	
1289	0DF3 06 49	ld b,49h	

1290	0DF5 0E 29	aci2:	ld c,29h	
1291	0DF7 16 15	aqi2:	ld d,15h	
1292	0DF9 15	aki2:	dec d	
1293	ODFA 20 FD		jr nz,aki2	
1294	ODFC 0D		dec c	
1295	ODFD 20 F8		jr nz,aqi2	
1296	ODFF 05		dec b	
1297	OE00 20 F3		jr nz,aci2	
1298	OE02 D1		pop de	
1299	OE03 C1		pop bc	
1300	OE04 F1		pop af	
1301	OE05 C9		ret	
1302	OE06		;=====	
1303	OE10	org	0e10h	
1304	OE10 F5	delay1:	push af	;Rutina delay1 con un tiempo largo
1305	OE11 C5		push bc	
1306	OE12 D5		push de	
1307	OE13 06 02		ld b,02h	
1308	OE15 0E 03	aci1:	ld c,03h	
1309	OE17 16 01	aqi1:	ld d,01h	
1310	OE19 15	aki1:	dec d	
1311	OE1A 20 FD		jr nz,aki1	
1312	OE1C 0D		dec c	
1313	OE1D 20 F8		jr nz,aqi1	
1314	OE1F 05		dec b	
1315	OE20 20 F3		jr nz,aci1	

1316	0E22 D1	pop de	
1317	0E23 C1	pop bc	
1318	0E24 F1	pop af	
1319	0E25 C9	ret	
1320	0E26	;=====	
1321	0E30	org	0e30h
1322	0E30 F5	desag: push af	;esta rutina descompone lo que hay en
1323	0E31 C5	push bc	;addrh-addr1 y lo pone en digi7,7,5 y 4
1324	0E32 D5	push de	;lo mismo para su contenido en digi2-1
1325	0E33 E5	push hl	
1326	0E34 2A 1A 20	ld hl,(addr1)	;carga toda la dirección RAM
1327	0E37 7C	ld a,h	;nibble alto de h (addh nibble alto)
1328	0E38 CB 3F	srl a	
1329	0E3A CB 3F	srl a	
1330	0E3C CB 3F	srl a	
1331	0E3E CB 3F	srl a	;recorre el nibble alto al bajo
1332	0E40 E6 0F	and 0fh	;pone a 0000 los 4 bits del nibble alto.
1333	0E42 32 22 20	ld (digi8),a	;lo guarda en digi8
1334	0E45 32 2D 20	ld (temp),a	;pone el valor en una localidad temporal
1335	0E48 CD 60 05	call convi	;convierte numero hex a segmentos
1336	0E4B 3A 2D 20	ld a,(temp)	;recupera valor convertido
1337	0E4E 32 12 20	ld (adnhh),a	;guarda el dígito convertido
1338	0E51		;
1339	0E51 7C	ld a,h	;nibble bajo de h (addh nibble bajo)
1340	0E52 E6 0F	and 0fh	
1341	0E54 32 23 20	ld (digi7),a	;lo guarda en digi7

1342	0E57 32 2D 20	ld (temp),a	
1343	0E5A CD 60 05	call convi	
1344	0E5D 3A 2D 20	ld a,(temp)	
1345	0E60 32 13 20	ld (adhn1),a	
1346	0E63	;	
1347	0E63 7D	ld a,l	;nibble alto de l (addl nibble alto)
1348	0E64 CB 3F	srl a	
1349	0E66 CB 3F	srl a	
1350	0E68 CB 3F	srl a	
1351	0E6A CB 3F	srl a	
1352	0E6C E6 0F	and 0fh	
1353	0E6E 32 24 20	ld (digi6),a	;lo guarda en digi6
1354	0E71 32 2D 20	ld (temp),a	
1355	0E74 CD 60 05	call convi	
1356	0E77 3A 2D 20	ld a,(temp)	
1357	0E7A 32 14 20	ld (adlnh),a	
1358	0E7D 7D	ld a,l	;nibble bajo de l (addl nibble bajo)
1359	0E7E E6 0F	and 0fh	
1360	0E80 32 25 20	ld (digi5),a	;lo guarda en digi5
1361	0E83 32 2D 20	ld (temp),a	
1362	0E86 CD 60 05	call convi	
1363	0E89 3A 2D 20	ld a,(temp)	
1364	0E8C 32 15 20	ld (adlnl),a	
1365	0E8F 3E 00	ld a,00h	;digi4 apagado
1366	0E91 32 26 20	ld (digi4),a	
1367	0E94 32 2D 20	ld (temp),a	

1368	0E97 CD 60 05	call convi	
1369	0E9A 3A 2D 20	ld a,(temp)	
1370	0E9D 32 26 20	ld (digi4),a	
1371	0EA0 7E	ld a,(hl)	;Lee el contenido de la RAM
1372	0EA1 CB 3F	srl a	
1373	0EA3 CB 3F	srl a	
1374	0EA5 CB 3F	srl a	
1375	0EA7 CB 3F	srl a	;recorre el nibble alto al bajo
1376	0EA9 E6 0F	and 0fh	;pone a 0000 los 4 bits del nibble alto.
1377	0EAB 32 27 20	ld (digi3),a	;lo guarda en digi3
1378	0EAE 32 2D 20	ld (temp),a	;pone el valor en una localidad temporal
1379	0EB1 CD 60 05	call convi	;convierte numero hex a segmentos
1380	0EB4 3A 2D 20	ld a,(temp)	;recupera valor convertido
1381	0EB7 32 17 20	ld (datnh),a	;guarda el dígito convertido
1382	0EBA 7E	ld a,(hl)	;nibble bajo de DATA (data nibble bajo)
1383	0EBB E6 0F	and 0fh	
1384	0EBD 32 28 20	ld (digi2),a	;lo guarda en digi2
1385	0EC0 32 2D 20	ld (temp),a	
1386	0EC3 CD 60 05	call convi	
1387	0EC6 3A 2D 20	ld a,(temp)	
1388	0EC9 32 18 20	ld (datnl),a	
1389	0ECC E1	pop hl	
1390	0ECD D1	pop de	
1391	0ECE C1	pop bc	
1392	0ECF F1	pop af	
1393	0ED0 C9	ret	

```

1394 0ED1      ;=====
1395 0EF0      org      0ef0h
1396 0EF0 3E 6E  hola:    ld a,6eh      ;H
1397 0EF2 32 12 20 ld (adnhn),a
1398 0EF5 3E 3A    ld a,3ah      ;o
1399 0EF7 32 13 20 ld (adhnl),a
1400 0EFA 3E 0C    ld a,0ch      ;L
1401 0EFC 32 14 20 ld (adlnh),a
1402 0EFF 3E EE    ld a,0eeh      ;A
1403 0F01 32 15 20 ld (adlnl),a
1404 0F04 3E 00    ld a,00h      ;
1405 0F06 32 16 20 ld (vacio),a
1406 0F09 3E EE    ld a,0eeh      ;A
1407 0F0B 32 17 20 ld (datnh),a
1408 0F0E 3E 1C    ld a,1ch      ;L
1409 0F10 32 18 20 ld (datnl),a
1410 0F13 C9      ret
1411 0F14      ;=====
1412 0F30      org      0f30h
1413 0F30 F5      flags:    push af      ;esta rutina prueba los flags y los
1414 0F31 C5      push bc      ;envía a sus localidades ledsI y ledsh
1415 0F32 D5      push de
1416 0F33 E5      push hl
1417 0F34 F5      push af      ;guarda en el stack el par AF
1418 0F35 C1      pop bc      ;y lo recupera en BC
1419 0F36 79      ld a,c

```

1420	0F37 32 10 20	ld (ledsl),a	;guarda C en ledsl
1421	0F3A E1	pop hl	
1422	0F3B D1	pop de	
1423	0F3C C1	pop bc	
1424	0F3D F1	pop af	
1425	0F3E C9	ret	
1426	0F3F	;=====	
1427	0F40	org	0f40h
1428	0F40 C5	bienv:	push bc
1429	0F41 D5	push de	
1430	0F42 00	nop	;rutina para mostrar el texto:
1431	0F43 01 8F 80	ld bc,808fh	; "Bienvenido al sistema:"
1432	0F46 ED 41	out (c),b	
1433	0F48 01 8C 00	ld bc,008ch	;PIA 1 controla el LCD display
1434	0F4B ED 41	out (c),b	;prepara los comandos propios
1435	0F4D 01 8E 00	ld bc,008eh	;de LCD: clr, home, mode, on/off
1436	0F50 ED 41	out (c),b	;function, etc. PA y PC
1437	0F52 01 8C 38	ld bc,388ch	;PA=00111000 -> Function set: DL=8 bit
1438	0F55 ED 41	out (c),b	;N=1 (2 líneas) y F=0 (5x8)
1439	0F57 CD E0 0F	call pulso1	;genera pulso en E y WR de LCD
1440	0F5A 00	nop	
1441	0F5B 01 8C 01	ld bc,018ch	;PA=00000001 -> Clear Display y
1442	0F5E ED 41	out (c),b	;DDRAM=00000000
1443	0F60 CD E0 0F	call pulso1	
1444	0F63 00	nop	
1445	0F64 01 8C 06	ld bc,068ch	;PA=00000110 -> Entry Mode Set

1446	0F67 ED 41	out (c),b	;I/D=1 (Incrementa), S=0 (no scroll)
1447	0F69 CD E0 0F	call pulso1	
1448	0F6C 00	nop	
1449	0F6D 01 8C 03	ld bc,038ch	;PA=00000011 -> Return Home,
1450	0F70 ED 41	out (c),b	;DDRAM=0000000, Cursor al inicio.
1451	0F72 CD E0 0F	call pulso1	
1452	0F75 00	nop	
1453	0F76 01 8C 0F	ld bc,0f8ch	;PA=00001111 -> Display on/off control
1454	0F79 ED 41	out (c),b	;D=1 (Display on), C=1 (cursor on)
1455	0F7B CD E0 0F	call pulso1	;B=1 (Blink on)
1456	0F7E 00	nop	
1457	0F7F 01 8C 14	ld bc,148ch	;PA=00010100 -> Cursor o Display Shift
1458	0F82 ED 41	out (c),b	;S/C= (no display shift), R/L=1 (shift
1459	0F84 CD E0 0F	call pulso1	;to the right)
1460	0F87 00	nop	
1461	0F88 01 8C 80	ld bc,808ch	;PA=10000000 -> DDRAM = 00h para que el
1462	0F8B ED 41	out (c),b	;texto inicial aparezca al extremo
1463	0F8D CD E0 0F	call pulso1	;izquierdo del LCD y avance a la derecha
1464	0F90 D1	pop de	
1465	0F91 C1	pop bc	
1466	0F92 C9	ret	
1467	0F93	;=====	
1468	0FA0	muest: org 0fa0h	;rutina que muestra textos
1469	0FA0 C5	push bc	
1470	0FA1 D5	push de	
1471	0FA2 16 0D	ld d,0dh	;primer renglon del LCD; solo 13 caracteres

1472	0FA4 21 2F 20	ld hl,carac	;apunta a la dirección de caracteres
1473	0FA7 5E	ld e,(hl)	;para alimentar al contador de texto
1474	0FA8 0E 8C	ld c,8ch	;Bienvenido al sistema:
1475	0FAA DD 2A 30 20	ld ix,(inice)	;apuntador del segmento de ROM o RAM
1476	0FAE DD 46 00	masda: ld b,(ix+00h)	;donde está el texto...Toma caracter
1477	0FB1 ED 41	out (c),b	;lo envía a LCD PA de PIA-1
1478	0FB3 CD F5 0F	call pulso2	;activa E y WR
1479	0FB6 DD 23	inc ix	;siguiente caracter
1480	0FB8 1D	dec e	
1481	0FB9 15	dec d	
1482	0FBA 20 F2	jr nz,masda	;los primeros caracteres exhibidos
1483	0FBC 01 8C C0	ld bc,0c08ch	;apunta al segundo renglon del LCD
1484	0FBF ED 41	out (c),b	;para el resto del texto
1485	0FC1 CD E0 0F	call pulso1	
1486	0FC4 DD 46 00	masde: ld b,(ix+00h)	;donde está el texto...Toma caracter
1487	0FC7 ED 41	out (c),b	;lo envía a LCD PA de PIA-1
1488	0FC9 CD F5 0F	call pulso2	;activa E y WR
1489	0FCC DD 23	inc ix	;siguiente caracter
1490	0FCE 1D	dec e	
1491	0FCF 20 F3	jr nz,masde	;hasta terminar
1492	0FD1 D1	pop de	
1493	0FD2 C1	pop bc	
1494	0FD3 C9	ret	
1495	0FD4	;=====	
1496	0FE0	org	0fe0h
1497	0FE0 C5	pulso1:	push bc

1498	0FE1 D5	push de	
1499	0FE2 01 8E 04	ld bc,048eh	;por el PC de PIA-1 saca pulso
1500	0FE5 ED 41	out (c),b	
1501	0FE7 CD 35 10	call retar5	;retardo
1502	0FEA 01 8E 00	ld bc,008eh	;quita pulso
1503	0FED ED 41	out (c),b	
1504	0FEF D1	pop de	
1505	OFF0 C1	pop bc	
1506	OFF1 C9	ret	
1507	OFF2	;=====	
1508	OFF5	org Off5h	;pulso por PC de PIA-1
1509	OFF5 C5	pulso2: push bc	;para activar LCD
1510	OFF6 01 8E 01	ld bc,018eh	
1511	OFF9 ED 41	out (c),b	
1512	OFFB CD 35 10	call retar5	
1513	OFFE 01 8E 05	ld bc,058eh	
1514	1001 ED 41	out (c),b	
1515	1003 CD 35 10	call retar5	
1516	1006 01 8E 01	ld bc,018eh	
1517	1009 ED 41	out (c),b	
1518	100B CD 35 10	call retar5	
1519	100E 01 8E 00	ld bc,008eh	
1520	1011 ED 41	out (c),b	
1521	1013 C1	pop bc	
1522	1014 C9	ret	
1523	1015	;=====	

1524	101A	org	101ah
1525	101A F5	retar4:	push af
1526	101B C5	push bc	
1527	101C D5	push de	
1528	101D 06 29	ld b,29h	
1529	101F 0E 15	ret3:	ld c,15h
1530	1021 16 12	ret2:	ld d,12h
1531	1023 15	ret1:	dec d
1532	1024 20 FD	jr nz,ret1	
1533	1026 0D	dec c	
1534	1027 20 F8	jr nz,ret2	
1535	1029 05	dec b	
1536	102A 20 F3	jr nz,ret3	
1537	102C D1	pop de	
1538	102D C1	pop bc	
1539	102E F1	pop af	
1540	102F C9	ret	
1541	1030	;=====	
1542	1035	org	1035h
1543	1035 F5	retar5:	push af
1544	1036 C5	push bc	
1545	1037 D5	push de	
1546	1038 06 03	ld b,03h	
1547	103A 0E 02	ret53:	ld c,02h
1548	103C 16 12	ret52:	ld d,12h
1549	103E 15	ret51:	dec d

1550	103F 20 FD	jr nz,ret51	
1551	1041 0D	dec c	
1552	1042 20 F8	jr nz,ret52	
1553	1044 05	dec b	
1554	1045 20 F3	jr nz,ret53	
1555	1047 D1	pop de	
1556	1048 C1	pop bc	
1557	1049 F1	pop af	
1558	104A C9	ret	
1559	104B	;=====	
1560	1070	org 1070h	;Rutina 'motor' invocada por INT 038
1561	1070 01 8F 80	motor: LD BC,808fh	;Prepara PIA-1, PA salidas
1562	1073 ED 41	OUT (C),B	;PC, salidas
1563	1075 01 93 80	LD BC,8093h	;Prepara PIA-2, PA,PB y PC
1564	1078 ED 41	OUT (C),B	;salidas
1565	107A 3E 00	LD A,00h	;limpia bandera identificadores
1566	107C 32 2B 20	LD (202bh),A	
1567	107F 1E 1C	LD E,1ch	;contador de caracteres a mostrar
1568	1081 21 30 11	LD HL,1130h	;en 1130h inician los caracteres
1569	1084 22 00 22	LD (2200h),HL	;usa 2200h como variable
1570	1087 16 07	repit: LD D,07h	;contador de dígitos del display
1571	1089 DD 2A 00 22	LD IX,(2200h)	;apunta al primer caracter
1572	108D FD 21 20 20	LD IY,2020h	;apunta a la RAM de caracteres
1573	1091 DD 7E 00	nuevo: LD A,(IX+00)	;acarrea caracteres de este
1574	1094 FD 77 00	LD (IY+00),A	;programa a la RAM de caracteres
1575	1097 D3 08	OUT (08h),A	;muestra en Puerto 08 como ayuda

1576	1099 FD 23	INC IY	;mueve apuntadores Y
1577	109B DD 23	INC IX	;y X
1578	109D 15	DEC D	;nuevo dígito
1579	109E 20 F1	JR NZ,nuevo	;mueve nuevo chr
1580	10A0 01 90 01	LD BC,0190h	;al terminar, mueve motor,01
1581	10A3 ED 41	OUT (C),B	;en sus posiciones de giro
1582	10A5 CD 00 11	CALL r2180	;haz pausa para movimiento
1583	10A8 01 90 03	LD BC,0390h	;correcto, 03
1584	10AB ED 41	OUT (C),B	;Se trata de mantener girando
1585	10AD CD 00 11	CALL r2180	;el motor, mientras los caracteres
1586	10B0 01 90 02	LD BC,0290h	;de la RAM se muestran en, 02
1587	10B3 ED 41	OUT (C),B	;círculos sobre el display, para
1588	10B5 CD 00 11	CALL r2180	;lo cual se diseña un movimiento
1589	10B8 01 90 06	LD BC,0690h	; de todos los caracteres, 06
1590	10BB ED 41	OUT (C),B	
1591	10BD CD 00 11	CALL r2180	
1592	10C0 01 90 04	LD BC,0490h	;04
1593	10C3 ED 41	OUT (C),B	
1594	10C5 CD 00 11	CALL r2180	
1595	10C8 01 90 0C	LD BC,0c90h	;0c
1596	10CB ED 41	OUT (C),B	
1597	10CD CD 00 11	CALL r2180	
1598	10D0 01 90 08	LD BC,0890h	;08
1599	10D3 ED 41	OUT (C),B	
1600	10D5 CD 00 11	CALL r2180	
1601	10D8 01 90 09	LD BC,0990h	;09

1602	10DB ED 41	OUT (C),B	
1603	10DD CD 00 11	CALL r2180	;al terminar movimiento básico...
1604	10E0 CD 60 11	CALL r2250	;ve a la rutina para mostrar
1605	10E3 21 00 22	LD HL,2200h	;caracteres. Apuntador en 2230h
1606	10E6 34	INC (HL)	;incrementa el contenido de 2230h
1607	10E7 1D	DEC E	;lleva la cuenta de caracteres
1608	10E8 20 9D	JR NZ,repit	;salta a 2117h
1609	10EA C3 70 10	JP motor	;
1610	10ED 00	NOP	
1611	10EE	;=====	
1612	1100	org	1100h
1613	1100 F5	r2180:	PUSH AF
1614	1101 C5		PUSH BC
1615	1102 D5		PUSH DE
1616	1103 06 19		LD B,19h
1617	1105 0E 11	c180:	LD C,11h
1618	1107 16 08	b180:	LD D,08h
1619	1109 15	a180:	DEC D
1620	110A 20 FD		JR NZ,a180
1621	110C 0D		DEC C
1622	110D 20 F8		JR NZ,b180
1623	110F 05		DEC B
1624	1110 20 F3		JR NZ,c180
1625	1112 D1		POP DE
1626	1113 C1		POP BC
1627	1114 F1		POP AF

```

1628 1115 C9      RET

1629 1116      ;=====

1630 1130      org    1130h

1631 1130 00 00 00 00 r2200: byte    00h,00h,00h,00h      ; , , ,
1632 1134 00 00 00 00      byte    00h,00h,00h,00h      ; , , ,
1633 1138 02 80 02 10      byte    02h,80h,02h,10h      ; , , -, _
1634 113C 02 80 02 10      byte    02h,80h,02h,10h      ; -, -, _
1635 1140 02 80 02 10      byte    02h,80h,02h,10h      ; -, -, _
1636 1144 02 80 02 10      byte    02h,80h,02h,10h      ; -, -, _
1637 1148 02 80 02 10      byte    02h,80h,02h,10h      ; -, -, _
1638 114C 02 80 02 10      byte    02h,80h,02h,10h      ; -, -, _
1639 1150 02 00 00 00      byte    02h,00h,00h,00h      ; -, , ,

1640 1154 00      NOP

1641 1155 00      NOP

1642 1156      ;=====

1643 1160      org    1160h

1644 1160 F5      r2250:    PUSH AF

1645 1161 C5      PUSH BC

1646 1162 D5      PUSH DE

1647 1163 E5      PUSH HL

1648 1164 DD E5    PUSH IX

1649 1166 CD 50 09  CALL agrupa      ;rutina "agrupa" caracteres

1650 1169 CD 30 0F  CALL 0f30h      ;rutina leds1 y leds2

1651 116C DD 21 20 20 LD IX,2020h    ;inicio de dígitos del display

1652 1170 16 07      LD D,07h

1653 1172 0E 05      LD C,05h

```

1654	1174 06 0F	LD B,0fh	
1655	1176 DD 7E 00	s2250: LD A,(IX+00)	
1656	1179 CB 80	RES 0,B	
1657	117B ED 41	OUT (C),B	
1658	117D CD 10 0E	CALL 0e10h	;rutina Delay con tiempo largo
1659	1180 CB C0	SET 0,B	
1660	1182 ED 41	OUT (C),B	
1661	1184 CD 10 0E	CALL 0e10h	
1662	1187 CB 80	RES 0,B	
1663	1189 ED 41	OUT (C),B	
1664	118B CD 10 0E	CALL 0e10h	
1665	118E 2F	CPL	
1666	118F CB 87	RES 0,A	
1667	1191 ED 79	OUT (C),A	
1668	1193 CD D0 0D	CALL Odd0h	;rutina retar1 con tiempo largo
1669	1196 CD D0 0D	CALL Odd0h	
1670	1199 DD 23	INC IX	
1671	119B 05	DEC B	
1672	119C 05	DEC B	
1673	119D 15	DEC D	
1674	119E 20 D6	JR NZ,s2250	
1675	11A0 DD E1	POP IX	
1676	11A2 E1	POP HL	
1677	11A3 D1	POP DE	
1678	11A4 C1	POP BC	
1679	11A5 F1	POP AF	

```

1680 11A6 C9      RET
1681 11A7      ;=====
1682 11B0      org    11b0h      ;Subrutina 'reloj' invocada por INT 66
1683 11B0 F5      reloj:    PUSH AF
1684 11B1 C5      PUSH BC
1685 11B2 D5      PUSH DE
1686 11B3 E5      PUSH HL
1687 11B4 DD E5      PUSH IX
1688 11B6 0E A0      LD C,0a0h
1689 11B8 3E 7F      LD A,7fh      ;prepara el 8570 para Funcionar
1690 11BA ED 79      OUT (C),A
1691 11BC 3E 3F      LD A,3fh
1692 11BE 0C      INC C
1693 11BF ED 79      OUT (C),A
1694 11C1 0E A6      segund: LD C,0a6h
1695 11C3 ED 78      IN A,(C)      ;lee los segundos
1696 11C5 32 2D 20    LD (temp),A      ;los carga en 'temp'
1697 11C8      ;    CALL decim      ;los convierte a decimal
1698 11C8 3A 2D 20    LD A,(temp)      ;recupera el dato convertido
1699 11CB 32 FF 29    LD (29ffh),A      ;los guarda en RAM
1700 11CE CD 40 12    CALL prepr      ;prepara datos para convi:
1701 11D1 3A FF 29    LD A,(29ffh)      ;ya convertido a segmentos los
1702 11D4 32 03 23    LD (2303h),A      ;pone en la RAM
1703 11D7 3A FE 29    LD A,(29feh)      ;para exhibirlos en el
1704 11DA 32 04 23    LD (2304h),A      ;display
1705 11DD 3E 90      LD A,90h      ;este es el semicolon ":"

```

1706	11DF 32 02 23	LD (2302h),A	;del reloj
1707	11E2 0C	INC C	
1708	11E3 ED 78	IN A,(C)	;lee minutos del 8570
1709	11E5 32 2D 20	LD (temp),A	;los carga en 'temp'
1710	11E8 ;	CALL decim	;los convierte a decimal
1711	11E8 3A 2D 20	LD A,(temp)	;recupera el dato convertido
1712	11EB 32 FF 29	LD (29ffh),A	;y hace todo igual
1713	11EE CD 40 12	CALL prepr	
1714	11F1 3A FF 29	LD A,(29ffh)	
1715	11F4 32 00 23	LD (2300h),A	
1716	11F7 3A FE 29	LD A,(29feh)	
1717	11FA 32 01 23	LD (2301h),A	;para dejarlo en la RAM
1718	11FD 16 07	LD D,07h	;prepara contadores
1719	11FF 0E 05	LD C,05h	
1720	1201 06 0A	LD B,0ah	
1721	1203 DD 21 00 23	LD IX,2300h	;apunta a la RAM datos convi:
1722	1207 DD 7E 00	carg: LD A,(IX+00)	;y los carga
1723	120A 2F	CPL	;se deben complementar
1724	120B CB 80	RES 0,B	;esta secuencia manda el dato
1725	120D ED 41	OUT (C),B	;al display rojo
1726	120F CD 10 0E	CALL 0e10h	;delay en el BIOS
1727	1212 CB C0	SET 0,B	
1728	1214 ED 41	OUT (C),B	
1729	1216 CD 10 0E	CALL 0e10h	
1730	1219 CB 80	RES 0,B	
1731	121B ED 41	OUT (C),B	

1732	121D CD 10 0E	CALL 0e10h	
1733	1220 CB 87	RES 0,A	
1734	1222 ED 79	OUT (C),A	;pone el dato en display
1735	1224 CD 78 12	CALL visio	;delay local para ajustar visión
1736	1227 DD 23	INC IX	;otro dato
1737	1229 05	DEC B	;se salta dígito 3
1738	122A 05	DEC B	
1739	122B 15	DEC D	
1740	122C 20 D9	JR NZ,carg	;regresa por el otro dato de RAM
1741	122E C3 C1 11	JP segund	;lee otra vez seg y min del 8570
1742	1231 DD E1	POP IX	
1743	1233 E1	POP HL	
1744	1234 D1	POP DE	
1745	1235 C1	POP BC	
1746	1236 F1	POP AF	
1747	1237	;=====	
1748	1240	org	1240h
1749	1240 F5	prepr:	PUSH AF
1750	1241 C5	PUSH BC	
1751	1242 D5	PUSH DE	
1752	1243 DD E5	PUSH IX	
1753	1245 3A FF 29	LD A,(29ffh)	;recupera dato de RAM
1754	1248 47	LD B,A	
1755	1249 CB 3F	SRL A	;recorre nibbles a LSB
1756	124B CB 3F	SRL A	;4 posiciones
1757	124D CB 3F	SRL A	

1758	124F CB 3F	SRL A	
1759	1251 E6 0F	AND 0fh	
1760	1253 32 2D 20	LD (202dh),A	;lo pone en variable 'temp'
1761	1256 CD 60 05	CALL 0560h	;convi: convierte de HEX a
1762	1259 3A 2D 20	LD A,(202dh)	;segmentos de display
1763	125C 32 FF 29	LD (29ffh),A	;lo devuelve a su RAM
1764	125F 78	LD A,B	;lo mismo para el nibble alto
1765	1260 E6 0F	AND 0fh	
1766	1262 32 2D 20	LD (202dh),A	
1767	1265 CD 60 05	CALL 0560h	;convi:
1768	1268 3A 2D 20	LD A,(202dh)	
1769	126B 32 FE 29	LD (29feh),A	;lo devuelve a su RAM
1770	126E DD E1	POP IX	
1771	1270 D1	POP DE	
1772	1271 C1	POP BC	
1773	1272 F1	POP AF	
1774	1273 C9	RET	;regresa a principal
1775	1274	;=====	
1776	1278	org	1278h
1777	1278 F5	visio: PUSH AF	;delay local para ajustar
1778	1279 C5	PUSH BC	;la visualización al ojo
1779	127A D5	PUSH DE	
1780	127B 06 08	LD B,08h	
1781	127D 0E 10	atras: LD C,10h	
1782	127F 16 05	atrs: LD D,05h	
1783	1281 15	atra: DEC D	

1784	1282 20 FD	JR NZ,atra	
1785	1284 0D	DEC C	
1786	1285 20 F8	JR NZ,atrs	
1787	1287 05	DEC B	
1788	1288 20 F3	JR NZ,atras	
1789	128A D1	POP DE	
1790	128B C1	POP BC	
1791	128C F1	POP AF	
1792	128D C9	RET	
1793	128E	;=====	
1794	1290	org	1290h
1795	1290 F5	input: push af	;esta rutina permite seleccionar un puerto
1796	1291 C5	push bc	;para entrada de datos, por lo que se tecleara
1797	1292 D5	push de	;el puerto valido de 00-ff + DP. El dato
1798	1293 E5	push hl	;aparecerá en el display y quedara en 3000h
1799	1294 01 0A 0F	ld bc,0f0ah	;prepara el puerto PIO2-A para salidas
1800	1297 ED 41	out (c),b	
1801	1299 21 1D 20	ld hl,bande	;en esta rutina se muestra el dig1 por eso
1802	129C CB CE	set 1,(hl)	;se pone bit1 = 1 de bande
1803	129E 3E 00	ld a,00h	;con IN: desaparece todo el display y solo
1804	12A0 32 12 20	ld (adnhn),a	;se muestra un '0' en el dígito 1. <2>
1805	12A3 32 13 20	ld (adhnI),a	; <1>
1806	12A6 32 14 20	ld (adlnh),a	; <0>
1807	12A9 32 15 20	ld (adlnI),a	; <0>
1808	12AC 32 17 20	ld (datnh),a	; <4>
1809	12AF 32 18 20	ld (datnI),a	; <5>

1810	12B2 3E 00	ld a,00h	;carga con '0' digi8 y digi7
1811	12B4 32 22 20	ld (digi8),a	;digi8 no se usa porque no existe en display
1812	12B7 32 23 20	ld (digi7),a	;digi7 es el nibble mas alto de ADDRESS
1813	12BA 3E FC	ld a,0fch	;FCh son los segmentos de '0'
1814	12BC 32 19 20	ld (digte),a	;que será mostrado en el dígito1
1815	12BF DD 21 14 20	ld ix,adlnh	;primera dirección de código de segmentos 2014h
1816	12C3 FD 21 24 20	ld iy,digi6	;primera dirección de código de dígitos 2024h
1817	12C7 16 01	ld d,01h	;se esperan 2 dígitos (puerto) mas DP (Enter)
1818	12C9 CD C0 01	buske: call scan	;escanea teclado y obtiene 'column' y 'fila'
1819	12CC 3A 2C 20	ld a,(column)	;para luego decodificar estas coordenadas
1820	12CF FE 00	cp 00h	
1821	12D1 20 06	jr nz,sidat	;si hubo dato del teclado, ve adelante
1822	12D3 CD A0 04	call displ	;si no hubo dato, muestra el display sin cambios
1823	12D6 C3 4D 13	jp bsclu	;Ahora detecta dato de RS232 <---PIC16F628A
1824	12D9 7A	sidat: ld a,d	;si hubo dato del teclado, mira si son los esperados
1825	12DA 28 ED	jr z,buske	;verifica si d = 0, es decir 'terminado'
1826	12DC CD 60 02	call decod	;primero decodifica coordenadas
1827	12DF 21 1D 20	r232m: ld hl,bande	;toma el valor de 'bande'
1828	12E2 CB 46	bit 0,(hl)	;prueba bit0 de 'bande'
1829	12E4 28 20	jr z,comdo	;z=1 (bit=0), es comando
1830	12E6 3A 2A 20	ld a,(tecl)	;z=0 (bit=1), es dígito.'tecl' tiene el código
1831	12E9 FD 77 00	ld (iy),a	;guarda dígito en su posición
1832	12EC 32 29 20	ld (digi1),a	;y en dig1 siempre
1833	12EF 32 2D 20	ld (temp),a	;para traspasar a rutina convertidora de
1834	12F2 CD 60 05	call convi	; 'segmentos'
1835	12F5 3A 2D 20	ld a,(temp)	;

1836	12F8 DD 77 00	ld (ix),a	;y guarda segmentos en adnhh-datl
1837	12FB 32 19 20	ld (digte),a	;y en el dígito1 como segmento 2019h
1838	12FE DD 23	inc ix	;apunta al siguiente valor de segmentos
1839	1300 FD 23	inc iy	;apunta al siguiente valor de dígitos
1840	1302 15	dec d	;contador = contador - 1
1841	1303 C3 C9 12	jp buske	;espera el siguiente dígito del teclado
1842	1306 21 1D 20	comdo: ld hl,bande	;Para comando, toma el valor de 'bande'
1843	1309 CB 46	bit 0,(hl)	;y prueba si es dígito o comando
1844	130B 28 06	jr z,sicdo	;z=1, es comando
1845	130D CD A0 04	call displ	;Z=0 no es dígito ni comando
1846	1310 C3 C9 12	jp buske	
1847	1313 3A 2A 20	sicdo: ld a,(tecl)	;toma el valor del comando y verifica si
1848	1316 FE 1B	cp 1Bh	;es 'Enter' (DP) o BRK
1849	1318 28 0E	jr z,hazya	;si es 1Bh ejecuta 'Enter'
1850	131A FE 1A	cp 1ah	;es BRK
1851	131C 28 57	jr z,acba	;sale de esta rutina
1852	131E FE 10	cp 10h	;Verifica si es retroceso <--
1853	1320 28 1C	jr z,retu	;retrocede un caracter
1854	1322 CD A0 04	call displ	;si no es 'DP' ni 'BRK' ni '<' muestra display
1855	1325 C3 C9 12	jp buske	;y espera nuevo dígito de ADDRESS.
1856	1328 CD 50 09	hazya: call agrupa	;agrupa los datos de ADDRESS introducidos
1857	132B 3A 1A 20	ld a,(addrl)	;para ejecutar IN (input)
1858	132E 4F	ld c,a	;prepara el PUERTO
1859	132F ED 40	in b,(c)	;introduce el DATO del PUERTO especificado
1860	1331 78	ld a,b	
1861	1332 32 1C 20	ld (data),a	;pone el dato leído del puerto en 'data'

1862	1335 CD 90 13	call dsgru	;y muestra el valor introducido por el puerto
1863	1338 CD A0 04	call displ	;muestra
1864	133B C3 75 13	jp acba	;termina esta rutina
1865	133E DD 2B	rettru: dec ix	
1866	1340 FD 2B	dec iy	
1867	1342 3E 00	ld a,00h	
1868	1344 DD 77 00	ld (ix),a	
1869	1347 CD A0 04	call displ	
1870	134A C3 C9 12	jp buske	
1871	134D DB 04	bsclu: in a,(04h)	;verifica si hay dato en PIC16F628A
1872	134F CB 67	bit 4,a	;prueba el bit4 de Acc (1 = Dato en Rs232)
1873	1351 28 1F	jr z,nobut	;si z=1, el bit4=0 no hay dato, regresa
1874	1353 00	nop	;si z=0, el bit4=1 si hay dato en RS232
1875	1354 DB 09	in a,(09h)	;introduce el dato, desde el PIC16F628A
1876	1356 D3 08	out (08h),a	;pone el dato en el puerto PIO2-A
1877	1358 32 1F 20	ld (serdat),a	;guarda dato en 'serdat'
1878	135B CD A0 0B	call hexa	;convierte de ASCII a hEX: serdat es ahora HEX
1879	135E 3E 01	ld a,01h	
1880	1360 D3 80	out (80h),a	;avisa a PIC dato leído
1881	1362 CD D0 0D	call retar1	
1882	1365 3E 00	ld a,00h	
1883	1367 D3 80	out (80h),a	
1884	1369 3A 1F 20	ld a,(serdat)	;toma el dato convertido a Hex
1885	136C 32 2A 20	ld (tecl),a	;y lo guarda en "tecl" para proceder
1886	136F C3 DF 12	jp r232m	;regresa para ver si es comando o dígito
1887	1372 C3 C9 12	nobut: jp buske	

1888	1375 E1	acba:	pop hl	
1889	1376 D1		pop de	
1890	1377 C1		pop bc	
1891	1378 F1		pop af	
1892	1379 C9		ret	
1893	137A	;=====		
1894	1390	org	1390h	
1895	1390 F5	dsgru:	push af	;esta rutina descompone lo que hay en
1896	1391 C5		push bc	;addrl y lo pone en digi5 y 4 y lo mismo
1897	1392 D5		push de	;para data poniendolo en digi2-1
1898	1393 E5		push hl	
1899	1394 2A 1A 20	ld hl,(addrl)		;carga toda la dirección RAM
1900	1397 7D	ld a,l		;nibble alto de l (addl nibble alto)
1901	1398 CB 3F	srl a		
1902	139A CB 3F	srl a		
1903	139C CB 3F	srl a		
1904	139E CB 3F	srl a		
1905	13A0 E6 0F	and 0fh		
1906	13A2 32 24 20	ld (digi6),a		;lo guarda en digi6
1907	13A5 32 2D 20	ld (temp),a		
1908	13A8 CD 60 05	call convi		
1909	13AB 3A 2D 20	ld a,(temp)		
1910	13AE 32 14 20	ld (adlnh),a		
1911	13B1 7D	ld a,l		;nibble bajo de l (addl nibble bajo)
1912	13B2 E6 0F	and 0fh		
1913	13B4 32 25 20	ld (digi5),a		;lo guarda en digi5

1914	13B7 32 2D 20	ld (temp),a	
1915	13BA CD 60 05	call convi	
1916	13BD 3A 2D 20	ld a,(temp)	
1917	13C0 32 15 20	ld (adInI),a	
1918	13C3 3E 00	ld a,00h	;digi4 apagado
1919	13C5 32 26 20	ld (digi4),a	
1920	13C8 32 2D 20	ld (temp),a	
1921	13CB CD 60 05	call convi	
1922	13CE 3A 2D 20	ld a,(temp)	
1923	13D1 32 26 20	ld (digi4),a	
1924	13D4 3A 1C 20	ld a,(data)	;lee data tomado del puerto con IN A,(C)
1925	13D7 CB 3F	srl a	
1926	13D9 CB 3F	srl a	
1927	13DB CB 3F	srl a	
1928	13DD CB 3F	srl a	;recorre el nibble alto al bajo
1929	13DF E6 0F	and 0fh	;pone a 0000 los 4 bits del nibble alto.
1930	13E1 32 27 20	ld (digi3),a	;lo guarda en digi3
1931	13E4 32 2D 20	ld (temp),a	;pone el valor en una localidad temporal
1932	13E7 CD 60 05	call convi	;convierte numero hex a segmentos
1933	13EA 3A 2D 20	ld a,(temp)	;recupera valor convertido
1934	13ED 32 17 20	ld (datnh),a	;guarda el dígito convertido
1935	13F0 3A 1C 20	ld a,(data)	;nibble bajo de DATA (data nibble bajo)
1936	13F3 E6 0F	and 0fh	
1937	13F5 32 28 20	ld (digi2),a	;lo guarda en digi2
1938	13F8 32 2D 20	ld (temp),a	
1939	13FB CD 60 05	call convi	

```

1940 13FE 3A 2D 20 ld a,(temp)
1941 1401 32 18 20 ld (datnl),a
1942 1404 E1      pop hl
1943 1405 D1      pop de
1944 1406 C1      pop bc
1945 1407 F1      pop af
1946 1408 C9      ret
1947 1409      ;=====
1948 1420      org    1420h
1949 1420 F5      decim:  push af      ;Esta rutina convierte números HEX a
1950 1421 C5      push bc      ;números decimales. Por ejemplo
1951 1422 D5      push de      ;0Eh lo convierte en 14 decimal
1952 1423 E5      push hl
1953 1424 3A 2D 20 ran0:  ld a,(temp)
1954 1427 FE 00    cp 00h
1955 1429 CA 3D 14 jp z,tneg0
1956 142C D2 32 14 jp nc,posi0
1957 142F C3 3D 14 jp tneg0
1958 1432 FE 09    posi0:  cp 09h
1959 1434 CA 3D 14 jp z,tneg0
1960 1437 DA 3D 14 jp c,tneg0
1961 143A C3 43 14 jp ran1
1962 143D 32 2D 20 tneg0: ld (temp),a
1963 1440 C3 06 15 jp termin
1964 1443      ;-----
1965 1443 3A 2D 20 ran1:  ld a,(temp)

```

```

1966 1446 FE 0A      cp 0ah
1967 1448 CA 5C 14   jp z,tneg1
1968 144B D2 51 14   jp nc,posi1
1969 144E C3 5C 14   jp tneg1
1970 1451 FE 13      posi1:   cp 13h
1971 1453 CA 5C 14   jp z,tneg1
1972 1456 DA 5C 14   jp c,tneg1
1973 1459 C3 64 14   jp ran2
1974 145C C6 06      tneg1:   add a,06h
1975 145E 32 2D 20   ld (temp),a
1976 1461 C3 06 15   jp termin
1977 1464                                ;-----
1978 1464 3A 2D 20   ran2:   ld a,(temp)
1979 1467 FE 14      cp 14h
1980 1469 CA 7D 14   jp z,tneg2
1981 146C D2 72 14   jp nc,posi2
1982 146F C3 7D 14   jp tneg2
1983 1472 FE 1D      posi2:   cp 1dh
1984 1474 CA 7D 14   jp z,tneg2
1985 1477 DA 7D 14   jp c,tneg2
1986 147A C3 85 14   jp ran3
1987 147D C6 0C      tneg2:   add a,0ch
1988 147F 32 2D 20   ld (temp),a
1989 1482 C3 06 15   jp termin
1990 1485                                ;-----
1991 1485 3A 2D 20   ran3:   ld a,(temp)

```

1992	1488 FE 1E	cp 1eh	
1993	148A CA 9E 14	jp z,tneg3	
1994	148D D2 93 14	jp nc,posi3	
1995	1490 C3 9E 14	jp tneg3	
1996	1493 FE 27	posi3: cp 27h	
1997	1495 CA 9E 14	jp z,tneg3	
1998	1498 DA 9E 14	jp c,tneg3	
1999	149B C3 A6 14	jp ran4	
2000	149E C6 12	tneg3: add a,12h	
2001	14A0 32 2D 20	ld (temp),a	
2002	14A3 C3 06 15	jp termin	
2003	14A6		;-----
2004	14A6 3A 2D 20	ran4: ld a,(temp)	
2005	14A9 FE 28	cp 28h	
2006	14AB CA BF 14	jp z,tneg4	
2007	14AE D2 B4 14	jp nc,posi4	
2008	14B1 C3 BF 14	jp tneg4	
2009	14B4 FE 31	posi4: cp 31h	
2010	14B6 CA BF 14	jp z,tneg4	
2011	14B9 DA BF 14	jp c,tneg4	
2012	14BC C3 C7 14	jp ran5	
2013	14BF C6 18	tneg4: add a,18h	
2014	14C1 32 2D 20	ld (temp),a	
2015	14C4 C3 06 15	jp termin	
2016	14C7		;-----
2017	14C7 3A 2D 20	ran5: ld a,(temp)	

2018	14CA FE 28	cp 28h	
2019	14CC CA E0 14	jp z,tneg5	
2020	14CF D2 D5 14	jp nc,posi5	
2021	14D2 C3 E0 14	jp tneg5	
2022	14D5 FE 31	posi5: cp 31h	
2023	14D7 CA E0 14	jp z,tneg5	
2024	14DA DA E0 14	jp c,tneg5	
2025	14DD C3 E8 14	jp ran6	
2026	14E0 C6 18	tneg5: add a,18h	
2027	14E2 32 2D 20	ld (temp),a	
2028	14E5 C3 06 15	jp termin	
2029	14E8		;-----
2030	14E8 3A 2D 20	ran6: ld a,(temp)	
2031	14EB FE 3C	cp 3ch	
2032	14ED CA 01 15	jp z,tneg6	
2033	14F0 D2 F6 14	jp nc,posi6	
2034	14F3 C3 01 15	jp tneg6	
2035	14F6 FE 45	posi6: cp 45h	
2036	14F8 CA 01 15	jp z,tneg6	
2037	14FB DA 01 15	jp c,tneg6	
2038	14FE C3 06 15	jp termin	
2039	1501 C6 24	tneg6: add a,24h	
2040	1503 32 2D 20	ld (temp),a	
2041	1506 E1	termin: pop hl	
2042	1507 D1	pop de	
2043	1508 C1	pop bc	

2044	1509 F1	pop af
2045	150A C9	ret
2046	150B	;=====
2047	1550	org 1550h
2048	1550 3E 00	ld a,00h
2049	1552 32 2B 20	ld (202bh),a
2050	1555 1E 1C	ld e,1ch
2051	1557 21 50 16	ld hl,1650h
2052	155A 22 00 22	ld (2200h),hl
2053	155D 16 07	carga: ld d,07h
2054	155F DD 2A 00 22	ld ix,(2200h)
2055	1563 FD 21 20 20	ld iy,2020h
2056	1567 DD 7E 00	carga: ld a,(ix+00)
2057	156A FD 77 00	ld (iy+00),a
2058	156D D3 08	out (08h),a
2059	156F FD 23	inc iy
2060	1571 DD 23	inc ix
2061	1573 15	dec d
2062	1574 20 F1	jr nz,carga
2063	1576 CD 00 17	call 1700h
2064	1579 CD 00 16	call 1600h
2065	157C 21 00 22	ld hl,2200h
2066	157F 34	inc (hl)
2067	1580 1D	dec e
2068	1581 20 DA	jr nz,carga
2069	1583 C3 50 15	jp 1550h

2070	1586 00	nop		
2071	1600	org	1600h	
2072	1600 F5	push af		
2073	1601 C5	push bc		
2074	1602 D5	push de		
2075	1603 06 49	ld b,49h		
2076	1605 0E 19	dec3:	ld c,19h	
2077	1607 16 15	dec2:	ld d,15h	
2078	1609 15	dec1:	dec d	
2079	160A 20 FD	jr nz,dec1		
2080	160C 0D	dec c		
2081	160D 20 F8	jr nz,dec2		
2082	160F 05	dec b		
2083	1610 20 F3	jr nz,dec3		
2084	1612 D1	pop de		
2085	1613 C1	pop bc		
2086	1614 F1	pop af		
2087	1615 C9	ret		
2088	1650	org	1650h	
2089	1650 00 00 00 00	byte	00h,00h,00h,00h	; , , ,
2090	1654 9E B6 1E DE	byte	9eh,0b6h,1eh,0deh	;E,S,t,E
2091	1658 00 1E 9E 6E	byte	00h,1eh,9eh,6eh	; ,t,E,X
2092	165C 1E 3A 00 CE	byte	1eh,3ah,00h,0ceh	;t,o, ,P
2093	1660 EE B6 EE 00	byte	0eeh,0b6h,0eeh,00h	;A,S,A,
2094	1664 B6 60 EC 00	byte	0b6h,60h,0ech,00h	;S,l,n,
2095	1668 9C 9E B6 EE	byte	9ch,9eh,0b6h,0eeh	;C,E,S,A

2096	166C 0A 00 00 00	byte	0ah,00h,00h,00h	;r, , ,
2097	1670 00 00 00 00	byte	00h,00h,00h,00h	; , , ,
2098	1674 00	nop		
2099	1675 00	nop		
2100	1700	org	1700h	
2101	1700 F5	push af		
2102	1701 C5	push bc		
2103	1702 D5	push de		
2104	1703 E5	push hl		
2105	1704 DD E5	push ix		
2106	1706 CD 50 09	call 0950h		
2107	1709 CD 30 0F	call 0f30h		
2108	170C DD 21 20 20	ld ix,2020h		
2109	1710 16 07	ld d,07h		
2110	1712 0E 05	ld c,05h		
2111	1714 06 0F	ld b,0fh		
2112	1716 DD 7E 00	carx: ld a,(ix+00)		
2113	1719 CB 80	res 00h,b		
2114	171B ED 41	out (c),b		
2115	171D CD 10 0E	call 0e10h		
2116	1720 CB C0	set 00h,b		
2117	1722 ED 41	out (c),b		
2118	1724 CD 10 0E	call 0e10h		
2119	1727 CB 80	res 00h,b		
2120	1729 ED 41	out (c),b		
2121	172B CD 10 0E	call 0e10h		

2122	172E 2F	cpl
2123	172F CB 87	res 00h,a
2124	1731 ED 79	out (c),a
2125	1733 CD D0 0D	call Odd0h
2126	1736 CD D0 0D	call Odd0h
2127	1739 DD 23	inc ix
2128	173B 05	dec b
2129	173C 05	dec b
2130	173D 15	dec d
2131	173E 20 D6	jr nz,carx
2132	1740 DD E1	pop ix
2133	1742 E1	pop hl
2134	1743 D1	pop de
2135	1744 C1	pop bc
2136	1745 F1	pop af
2137	1746 C9	ret
2138	1747	;=====
2139	1747	end

Label	Value	Label	Value	Label	Value
-----	-----	-----	-----	-----	-----
adhnh	2012	adhnl	2013	adlnh	2014
adlnl	2015	addrl	201A	addrh	201B
aci3	004D	aqi3	004F	aki3	0051

acab	07E0	acaba	0926	agrupa	0950
acabo	0A6F	ars	0A90	ahex	0CAC
avahex	0D10	arshex	0D1A	aci	0DD5
aqi	0DD7	aki	0DD9	aci2	0DF5
aqi2	0DF7	aki2	0DF9	aci1	0E15
aqi1	0E17	aki1	0E19	a180	1109
atras	127D	atrs	127F	atra	1281
acba	1375	bande	201D	busk	0739
bscla	07B8	busca	0879	bscb	08FE
busc	09CC	bscd	0A47	bhex	0CB6
brkhex	0CFC	bk1hex	0D24	bienv	0F40
b180	1107	buske	12C9	bsclu	134D
colum	202C	carac	202F	cor10	0332
cor11	033A	cor12	0342	cor13	034A
cor20	0352	cor21	035A	cor22	0362
cor23	036A	cor30	0372	cor31	037A
cor32	0382	cor33	038A	cor40	0392
cor41	039A	cor42	03A2	cor43	03AA
cor50	03B2	cor51	03BA	cor52	03C2
cor53	03CA	cor60	03D2	cor61	03DA
cor62	03E2	cor63	03EA	cor70	03F2
cor71	03FA	cor72	0402	cor73	040A
convi	0560	cero	05B6	cuat	05CE
cinc	05D4	comma	0776	coman	08B6
comand	0A09	cer0	0C48	cuat4	0C70
cinc5	0C7A	chex	0CC0	crhex	0CE8

c180	1105	carg	1207	comdo	1306
cargo	155D	carga	1567	carx	1716
datnh	2017	datnl	2018	digte	2019
data	201C	datin	201E	digi10	2020
digi9	2021	digi8	2022	digi7	2023
digi6	2024	digi5	2025	digi4	2026
digi3	2027	digi2	2028	digi1	2029
decod	0260	displ	04A0	dos	05C2
der	0800	dos2	0C5C	dhex	0CCA
dphex	0D2E	delay1	0E10	desag	0E30
dsgru	1390	decim	1420	dec3	1605
dec2	1607	dec1	1609	esa	05F2
esb	05F8	esc	05FE	esd	0604
ese	060A	esf	0610	ejecut	0650
esder	0684	esizq	0688	esstp	068C
esla	0690	es2nd	069B	esss	069C
esin	069D	esld	06A1	esars	06AC
esgo	06B0	esbrk	06BB	esdp	06BC
ehex	0CD4	fila	202B	fil1	028A
fil2	02A2	fil3	02BA	fil4	02D2
fil5	02EA	fil6	0302	fil7	031A
fhex	0CDE	fine	0D7E	flags	0F30
goir	09A0	gohex	0D38	hacer	0798
hazlo	08D8	haz	0A2B	hexa	0BA0
hola	0EF0	hazya	1328	inice	2030
inice1	2031	izq	0820	inchex	0D42

input	1290	ledsl	2010	ledsh	2011
la	0700	ld	0840	lfhex	0CF2
lahex	0D4C	ldhex	0D56	most	04F1
muest	0FA0	masda	0FAE	masde	0FC4
motor	1070	nuev	05EC	ningún	0616
nobat	07DD	nobct	0923	nobdt	0A6C
noeof	0A9A	nuev9	0CA2	nuevo	1091
nobut	1372	otro1	01CD	otro	04B6
otra	04F3	ocho	05E6	otrda	0AC4
ocho8	0C98	pto08	202E	poste	086F
pulso1	0FE0	pulso2	0FF5	prepr	1240
posi0	1432	posi1	1451	posi2	1472
posi3	1493	posi4	14B4	posi5	14D5
posi6	14F6	r232s	019D	r232a	074F
retr	07A9	r232b	088F	retro	08EF
r232d	09E2	retra	0A38	rethex	0D06
retar1	0DD0	retar2	0DF0	retar4	101A
ret3	101F	ret2	1021	ret1	1023
retar5	1035	ret53	103A	ret52	103C
ret51	103E	repit	1087	r2180	1100
r2200	1130	r2250	1160	reloj	11B0
r232m	12DF	retru	133E	ran0	1424
ran1	1443	ran2	1464	ran3	1485
ran4	14A6	ran5	14C7	ran6	14E8
serdat	201F	sig	0163	sitecl	019A
scan	01C0	salt	01ED	salida	0203

seis	05DA	siet	05E0	stp	06C0
sitek	0749	sicmd	0783	sitec	0889
sicom	08C3	site	09DC	sico	0A16
seis6	0C84	siet7	0C8E	sndhex	0D60
sshex	0D6A	sthex	0D74	s2250	1176
segund	11C1	sidat	12D9	sicdo	1313
tecl	202A	temp	202D	tecpre	01E6
tres	05C8	tres3	0C66	tneg0	143D
tneg1	145C	tneg2	147D	tneg3	149E
tneg4	14BF	tneg5	14E0	tneg6	1501
termin	1506	uno	05BC	uno1	0C52
vacio	2016	visio	1278		

tasm: Number of errors = 0

APÉNDICE 3

Código en C para el PIC 16F628A que funciona como interface serial entre el sistema computador y una PC.

;

//Programa con el PIC 16F628A utilizado como interface para sustituir al

//PIC16f84A en el prototipo del Z80, pero que realiza las mismas funciones

//de comunicación del Z80 con una PC a través del puerto serial RS-232.

//Alfredo Segura. Querétaro, Mexico julio de 2013//

//-----

#include <16f628a.h>

#fuses XT, NOWDT, NOPROTECT, PUT, MCLR, NOBROWNOUT, NOCPD, NOLVP

```
#use delay (clock = 4000000)
```

```
#use rs232(baud=1200, xmit=PIN_B2, rcv=PIN_B1, bits=8, parity=N, stop=1)
```

```
#use fast_io(a)
```

```
#use fast_io(b)
```

```
#byte porta = 0x05
```

```
#byte portb = 0x06
```

```
#bit ra0 = 0x05.0 //Salida dato D4 a PB de Z80-PIO1
```

```
#bit ra1 = 0x05.1 //Salida dato D5 a PB de Z80-PIO1
```

```
#bit ra2 = 0x05.2 //Salida dato D6 a PB de Z80-PIO1
```

```
#bit ra3 = 0x05.3 //Salida dato D7 a PB de Z80-PIO1
```

```
#bit ra4 = 0x05.4 //No usada
```

```
#bit ra5 = 0x05.5 //Entrada RESET del PIC
```

```
#bit ra6 = 0x05.6 //XTAL
```

```
#bit ra7 = 0x05.7 //XTAL
```

```
#bit rb0 = 0x06.0 //Salida: Señal dato disponible en puertos A y B hacia el Z80-PIO2
```

```
#bit rb1 = 0x06.1 //Entrada: Rx de RS-232
```

```
#bit rb2 = 0x06.2 //Salida: Tx a RS-232
```

```
#bit rb3 = 0x06.3 //Entrada: Señal de Q5 de 74LS373, dato aceptado por el Z80
```

```
#bit rb4 = 0x06.4 //Salida dato D0 a PB de Z80-PIO1
```

```
#bit rb5 = 0x06.5 //Salida dato D1 a PB de Z80-PIO1
```

```
#bit rb6 = 0x06.6 //Salida dato D2 a PB de Z80-PIO1
```

```
#bit rb7 = 0x06.7 //Salida dato D3 a PB de Z80-PIO1
```

```

int i = 0, j = 0, k = 0;

char dato = "", dswp = "";

byte const bienve[12] = {0x0d,'B','i','e','n','v','e','n','i','d','o',0x20};

//-----INI-----

void ini(void) {

    setup_timer_0(t0_internal|t0_div_16);

    set_tris_a(0b11100000); //

    set_tris_b(0b00001010); //rb1-rb2 (rx-tx), rb0 = ACK dato, rb3 = Dato OK

    porta = 0; portb = 0;

}

//-----MAIN-----

void main (void) {

    ini();

    for (i=0; i<=11; i++) {

        putc(bienve[i]);

    }

    rb0 = 0;

    while (true) {

        if (kbhit()) {

            dato = getc();          //0101 0111  57d

            dswp = (dato<<4)|(dato>>4);

            porta = dswp & 0x0f;    //0111 0101  75d, sale el 5 por ra0-ra3 nibble alto de dato

            portb = dswp & 0xf2;    //0111 0101 AND 1111 0100 = 0111 0100

            rb0 = 1;              //Señal dato disponible en puertos A y B hacia el Z80-PIO2

            do {

```

```
j++;          //Si el Z80 no ha aceptado el dato, se queda aquí indefinidamente  
} while (rb3 == 0);  
rb0 = 0;  
}  
k++;  
}  
// Se cierra While  
}  
//Se cierra Main  
;
```