UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y DESARROLLO DE UN DISPOSITIVO ELECTRONICO QUE PERMITE LA INTERCONEXION SIMULTANEA Y AUTOMATICA DE 8 DISPOSITIVOS PERIFERICOS CON UNA PC, A TRAVES DEL PUERTO PARALELO

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A:

LIDIA BOTELLO ACOSTA

DIRECTOR DE TESIS:

ING. VICTOR M. TORRES GODINEZ

México, D. F. 1997

A Hamlet

A mis padres

A mis hermanos

Agradezco a Hamlet el haberme dado todo su apoyo profesional para el desarrollo de este proyecto, el haberme guiado acertadamente cuando tuve que enfrentar y solucionar todos los problemas que se me presentaron.

Agradezco al Ing. Victor M. Torres Godinez, mi asesor, por su apoyo profesional y comprensión.

Agradezco al Ing. Antonio Salvá Calleja por brindarme su apoyo profesional.

Agradezco al Ing. Francisco Rodríguez Ramírez por sus comentarios y sugerencias en el trabajo escrito.

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

En la actualidad los procesadores de textos y formadores de publicaciones permiten generar documentos cuyo contenido puede estar formado por caracteres de diferentes tipos y tamaños, dibujos, imágenes, gráficos, etc., ya sea en blanco y negro o en color. Ésta es una de las razones por las que se utiliza una computadora en la elaboración de documentos que requieren calidad y presentación con un esfuerzo mínimo. Si se desea reproducir fielmente en papel este tipo de documentos, es necesario elegir el dispositivo de impresión adecuado. Actualmente hay una gran variedad en lo que a dispositivos de impresión se refiere, desde impresoras de matriz de puntos, hasta impresoras láser e impresoras de inyección de tinta, con impresión en color o blanco y negro. Para la reproducción de planos, que normalmente son de un tamaño mayor, generalmente se utiliza un dispositivo de impresión especial conocido comúnmente como graficador (plotter).

Muchos usuarios de computadoras que cuentan con más de un dispositivo de impresión y desean aprovechar en forma ágil las ventajas que cada uno de estos les ofrece, han optado por utilizar una interface que les permite conectar un determinado número de dispositivos al puerto paralelo de la computadora personal (PC) en forma simultánea.

Este tipo de interfaces se conocen comúnmente con el nombre de "semáforos" o "parallel switch", y por la forma en que conmutan hacia los diferentes puertos de salida se clasifican como manuales y automáticos.

Los "parallel switch" manuales que se encuentran en el mercado, tienen un puerto de entrada y 2, 3 o 4 puertos de salida, están provistos de un selector con el número de posiciones igual al número de puertos de salida. Al mover el selector, las líneas del puerto de entrada quedan unidas físicamente una a una a las líneas del puerto de salida

seleccionado, quedando los demás puertos desconectados. Carecen de elementos electrónicos por lo que no requieren voltaje de polarización.

Algunos de los "parallel switch" automáticos comerciales son de un puerto de entrada y dos de salida (tipo A) y de 4 puertos de entrada con 2 de salida (tipo B)[1] .

El que sean automáticos, implica que su fabricación se realiza con componentes electrónicos y por lo tanto se requiera de un voltaje externo para la polarización de los mismos.

En un dispositivo tipo A, el puerto de entrada se conecta directamente al puerto paralelo de la PC y a cada uno de los 2 puertos de salida, los dispositivos de impresión. En un dispositivo tipo B, se conectan simultáneamente hasta 4 PC a los respectivos puertos de entrada (4) y a los 2 puertos de salida los dispositivos de impresión correspondientes. En ambos casos el usuario selecciona, desde la PC por la que va a transmitir su información, el puerto de salida en el que se encuentra conectado el dispositivo de impresión que utilizará para imprimir su documento. En términos generales la conmutación entre los dos puertos de salida se realiza de la siguiente manera: partiendo de que el puerto de salida #1 está activo y el usuario desea transmitir por el puerto de salida #2, el usuario deberá enviar desde la computadora n veces un determinado caracter de control, al ser detectada esta secuencia por el dispositivo, éste realiza automáticamente la conmutación hacia el puerto #2, y viceversa.

El objetivo de este trabajo es diseñar y desarrollar un dispositivo electrónico que permita la interconexión simultánea y automática de 8 dispositivos periféricos con una computadora personal, a través del puerto paralelo. Este dispositivo está diseñado con base en el microcontrolador MC68HC11 de Motorola.

El dispositivo electrónico diseñado se denominó Interface Electrónica para Puerto Paralelo (IEPP) y en términos generales opera de la siguiente manera:

Al conectar la IEPP directamente al puerto paralelo de una computadora personal, ésta será capaz de recibir toda la información que la PC transmita bajo el protocolo de

[1] Nombrados tipo A y tipo B, sólo con la finalidad de referirse a los mismos de manera más sencilla en el documento.

comunicación "Centronics Parallel Interface". Identificar hacia cuál de los 8 puertos de salida va dirigido cada bloque de información y por consiguiente transmitirlo íntegramente al dispositivo periférico correspondiente. Los dispositivos periféricos se conectan directamente a los puertos de salida de la IEPP en forma simultánea. El usuario decide el puerto de salida que desea mantener activo, para ello se dispone de un software de apoyo que le permite hacerlo a través de un menú, así como también, puede solicitar el estado en el que opera cada uno de los puertos de salida de la IEPP.

Se puede conectar a la IEPP cualquier dispositivo periférico que utilice el protocolo de comunicación estándar para puerto paralelo "Centronics Parallel Interface", por ejemplo, impresoras de matriz de puntos, impresoras láser, impresoras de inyección de tinta, graficadores, etc., y bajo algunas restricciones es posible conectar algunos dispositivos que no utilizan este protocolo, por ejemplo, sistemas integrados con motores de pasos (brazos mecánicos, bandas transportadoras, etc.) y PC's (si el propósito es de que éstas reciban información a través del puerto paralelo).

Pensando en que se puede conectar a la IEPP algún dispositivo que no disponga de memoria RAM para almacenar la información antes de procesarla, la interface cuenta con 32 Kbytes de RAM.

El trabajo está organizado de la siguiente manera:

En el Capítulo 2 se presenta la descripción de cada uno de los módulos que conforman la IEPP, en el Capítulo 3 se presenta y se analiza el hardware implementado en el diseño de cada uno de estos bloques o módulos, en el Capítulo 4 se presenta la descripción de la estructura de los programas que conforman el software del sistema, en el Capítulo 5 se analizan las pruebas experimentales efectuadas a la IEPP con el propósito de comprobar que la interface satisface plenamente sus objetivos, y en el Capítulo 6 se analizan los resultados arrojados en las pruebas experimentales, así como algunas posibles mejoras en la construcción del hardware de la IEPP.

En el Apéndice A se presenta un listado de cada uno de los dos programas que conforman el sistema, así como de los programas desarrollados para llevar a cabo algunas de las pruebas que se le realizaron a la interface. En el Apéndice B se presenta

un manual de usuario, con la finalidad de evitar principalmente algún error durante las conexiones entre los diferentes dispositivos. Finalmente en el Apéndice C, se presenta el diagrama completo de la interconexión de los diferentes bloques que conforman el hardware de la IEPP.

CAPÍTULO 2

DESCRIPCIÓN DEL PROYECTO

En este capítulo se presenta el modelo conceptual de la Interface Electrónica para Puerto Paralelo, así como una descripción general de cada uno de los bloques o módulos que la conforman.

Como se mencionó en el capítulo anterior, la IEPP es un dispositivo que permite la interconexión simultánea de 8 dispositivos periféricos a una computadora personal a través del puerto paralelo.

En este sentido, se puede decir que la IEPP es un dispositivo electrónico diseñado para multiplexar de 1 a 8 el puerto paralelo de una computadora personal. La interconexión entre la PC, la IEPP y los diferentes periféricos se ilustra en el diagrama a bloques de la figura 2.1.

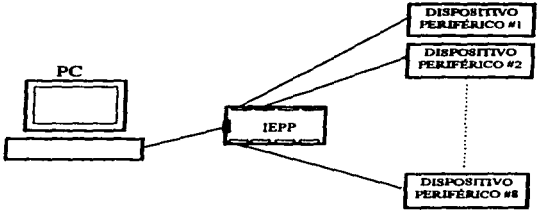


Figura 2.1.- Diagrama de interconexiones con la IEPP.

El protocolo de comunicación entre la PC y la IEPP es el "Centronics Parallel Interface" (CPI) o "Standard Parallel Port" (SPP)[2], debido a que la mayoría de periféricos para

[2] Para referirse al protocolo de comunicación estándar para puerto paralelo se usará indistintamente Centronics Parallel Interface o Standard Parallel Port (SPP).

puerto paralelo existentes en el mercado siguen este protocolo de comunicación. En este sentido la IEPP es un dispositivo completamente autónomo y portable, ya que se conecta directamente al puerto paralelo de cualquier computadora personal y, ésta lo ve como cualquier otro periférico.

Además de los dispositivos mencionados en el capítulo anterior que pueden conectarse a los puertos de salida de la IEPP, también es posible conectar a la interface algunos dispositivos que no utilizan el protocolo SPP para recibir información, por ejemplo, los sistemas integrados con motores de pasos (brazos mecánicos, bandas transportadoras, etc.) y bajo ciertas condiciones (descritas con detalle en el capítulo 5) computadoras personales operando como esclavas, es decir, que reciben información a través del puerto paralelo.

Una de las razones por las que se utilizó un microcontrolador en el diseño de la IEPP, fue precisamente la de lograr la conmutación automática entre los diferentes periféricos conectados a la interface. El microcontrolador utilizado fue el MC68HC11E1 de Motorola, debido a que satisface las características requeridas para realizar en torno a él una interface electrónica que cumpla con los objetivos planteados en este proyecto, además de que este dispositivo es muy fácil de conseguir en el mercado y es el microcontrolador con el que más familiarizados estamos los estudiantes de esta facultad.

Para realizar la conmutación automática desde la PC, el usuario envía una palabra de control[3] para indicarle a la IEPP por cual de los 8 puertos de salida deberá transferir la información que a continuación reciba del puerto paralelo de la PC.

Debido a que tanto el protocolo de comunicación como el microcontrolador utilizados son determinantes en el diseño de la interface, enseguida se mencionan las principales características de los mismos.

[3] Secuencia de 8 bytes que le permite al usuario indicarle al M68HC11E1 que realice una determinada acción.

2.1 Características de la Interface Centronics (Centronics Parallel Interface o Standard Parallel Port).

Inicialmente el puerto paralelo se concibió sólo para el proceso de impresión, por esta razón las normas establecidas por el protocolo de comunicación estándar desarrollado por Centronics, sólo permite transmisión de datos de la PC hacia el periférico conectado a dicho puerto. Actualmente se manejan diferentes protocolos de comunicación para el puerto paralelo con la finalidad de satisfacer las diferentes necesidades de cada uno de los productos existentes en el mercado: impresoras, plotters, scaners, cd-roms, discos duros, adaptadores de red, etc. A continuación se mencionan los diferentes protocolos de comunicación desarrollados para el puerto paralelo.

1. Standard Parallel Port o Centronics Parallel Interface (SPP o CPI).- Dirección de datos de la computadora hacia el dispositivo periférico, introducido por la compañía Centronics.
2. Nibble Mode.- Dirección de datos del dispositivo hacia la computadora, transmite 4 bits de datos a la vez usando las líneas de estado, originalmente introducido por Hewlett Packard bajo el nombre de Bi-tronics.
3. Byte Mode.- Dirección de datos del dispositivo periférico hacia la computadora, transmite 8 bits de datos a la vez usando las líneas de datos, fue introducido por IBM en las PS/2.
4. ECP (Extended Capability Port).- Dirección de datos bidireccional usado principalmente por la nueva generación de impresoras y scaners, propuesto originalmente por Hewlett Packard y Microsoft.
5. EPP (Enhanced Parallel Port).- Dirección de datos bidireccional, usado principalmente por CD-ROM, cintas, discos duros y adaptadores de red, creado en 1994 por el IEEE 1284 Committee.

Debido a que la IEPP recibe y transmite datos siguiendo las normas del "Standard Parallel Port" o "Centronics Parallel Interface", este protocolo se describe a detalle en los párrafos subsecuentes.

Centronics parallel Interface o Standard Parallel Port fue el protocolo de señalización para el puerto paralelo adoptado por IBM y otras compañías. Esta es una interface TTL estándar (0.0 -0.8 V para un nivel lógico cero y 2.4 - 5.0 V para un nivel lógico uno). En cuanto a la longitud del cable se tiene un máximo de 4.5 metros para un ensamblado típico, y hasta 9.15 metros (30 pies) cuando el cable es construido utilizando las normas de control de impedancia, capacitancia e interferencia, del estándar 1284 de la IEEE. El cable entre la computadora y la impresora utiliza un conector macho Amphenol 57-30360 de 36 terminales (pines) hacia el lado de la impresora y un conector macho DB-25 hacia el lado de la computadora.

Para explicar el Standard Parallel Port, en la Figura 2.2 se presenta el diagrama de tiempos de las señales que intervienen en la transmisión de un caracter desde la computadora hacia el dispositivo terminal y en la Tabla 2.1 y Tabla 2.2 las señales asociadas a cada terminal de los respectivos conectores.

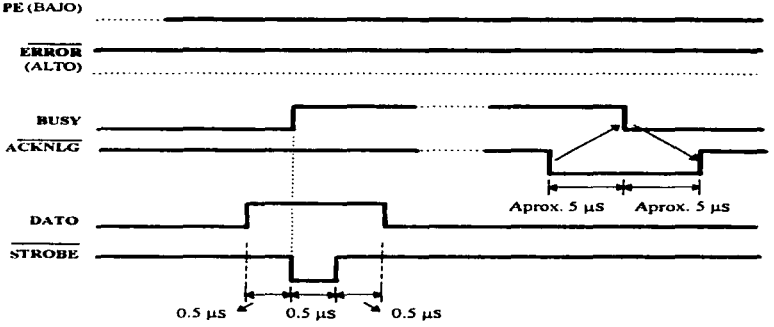


Figura 2.2.- Diagrama de tiempos que sigue el protocolo de comunicación SPP.

El diagrama de tiempos de la Figura 2.2 muestra la señalización del ciclo de transferencia de un caracter o dato. El ciclo comienza cuando la computadora pone el caracter en las líneas de datos del puerto paralelo, a continuación verifica que la señal de

ERROR tenga un nivel alto y la señal **PE** un nivel bajo. La condición siguiente es que la señal de **BUSY** presente un nivel bajo, ya que es la forma que tiene el dispositivo periférico de indicar que está listo para recibir un caracter. A continuación la PC genera un pulso bajo en la señal de **STROBE** para indicarle al periférico que hay un caracter listo en las líneas de datos. Cuando éste ha sido recibido, el periférico genera como señal de reconocimiento (acknoledge), un pulso bajo de 10 microsegundos en la señal de **ACKNLG**. En un dispositivo de impresión la señal de **BUSY** permanece en nivel alto durante la recepción e impresión de datos y cuando se encuentra en el estado de pausa o error.

# Pin	Nombre	Dirección	Activa	Función
1	STROBE	Entrada	Baja	Indica que el caracter está listo en las conexiones de datos D0-D7.
2	D0	Entrada	Alta	Conexión Dato 0.
3	D1	Entrada	Alta	Conexión Dato 2.
4	D2	Entrada	Alta	Conexión Dato 2.
5	D3	Entrada	Alta	Conexión Dato 3.
6	D4	Entrada	Alta	Conexión Dato 4.
7	D5	Entrada	Alta	Conexión Dato 5.
8	D6	Entrada	Alta	Conexión Dato 6.
9	D7	Entrada	Alta	Conexión Dato 7.
10	ACKNLG	Salida	Baja	Indica que el caracter ha sido recibido por la impresora.
11	BUSY	Salida	Alta	Indica que la impresora está ocupada recibiendo o imprimiendo un caracter, por lo que la PC no debe enviar otro caracter por el momento.
12	PE	Salida	Alta	Indica que la impresora está sin papel.
13	SLCT	Salida	Alta	Indica que la impresora está en linea.
14	AUTO FEED	Entrada	Baja	La PC indica a la impresora que inserte una linea después de cada Carry Return (CR)
15	NC			Sin conectar
16	GND			Tierra.
17	CHS			Tierra física.
18	NC			Sin Conectar
19-30	GND			Tierra.
31	INIT	Entrada	Baja	Inicializa la impresora. Limpia el buffer de impresión.
32	ERROR	Salida	Baja	Indica que ocurrió algún error en la impresora.
33	GND			Tierra
34-35	NC			Sin Conectar
36	SLCT IN	Entrada	Baja	Siempre en nivel bajo para indicar a la impresora que está seleccionada.

Tabla 2.1.- Descripción de señales del SPP vista desde la impresora.

# Pin	Nombre	Dirección	Activa	Función
1	**STROBE**	Salida	Baja	Indica que el caracter (D0-D7) está listo.
2	**D0**	Salida	Alta	Conexión Dato 0.
3	**D1**	Salida	Alta	Conexión Dato 2.
4	**D2**	Salida	Alta	Conexión Dato 2.
5	**D3**	Salida	Alta	Conexión Dato 3.
6	**D4**	Salida	Alta	Conexión Dato 4.
7	**D5**	Salida	Alta	Conexión Dato 5.
8	**D6**	Salida	Alta	Conexión Dato 6.
9	**D7**	Salida	Alta	Conexión Dato 7.
10	**ACKNLG**	Entrada	Baja	Indica que el caracter ha sido recibido por la impresora.
11	**BUSY**	Entrada	Alta	Indica que la impresora está ocupada recibiendo o imprimiendo un caracter, por lo que la PC no debe enviar otro caracter por el momento.
12	**PE**	Entrada	Alta	Indica que la impresora está sin papel.
13	**SLCT**	Entrada	Alta	Indica que la impresora está en línea.
14	**AUTO FEED**	Salida	Baja	La PC indica a la impresora que inserte una línea después de cada Carry Return (CR)
15	**ERROR**	Entrada	Baja	Indica estado de error en la impresora.
16	**INIT**	Salida	Baja	Inicializa la impresora. Limpia el buffer de impresión.
17	**SLCT IN**	Salida	Baja	Siempre en nivel bajo para indicar a la impresora que está seleccionada.
18-25	**GND.**			Tierra.

Tabla 2.2- Descripción de señales (SPP) vista desde el puerto paralelo de la PC.

2.2. Microcontrolador M68HC11.

El microcontrolador M68HC11 contiene un CPU, memoria (RAM y ROM), terminales de entrada/salida para comunicación serial y paralela, convertidor analógico-digital y temporizador. Es un microcontrolador que normalmente se utiliza en aplicaciones de adquisición y control de datos. Motorola empleó en la construcción de esta familia de microcontroladores la tecnología HCMOS (high-density complementary metal-oxide semiconductor), la cual combina tamaño pequeño, alta velocidad, bajo consumo de potencia y alta inmunidad al ruido. Como en el diseño se emplea específicamente el microcontrolador MC68HC11E1, las características que se describen en seguida corresponden a las series E.

2.2.1 Características generales en las series E de la familia de microcontroladores M68HC11.

- 8 bits de datos y 16 líneas de direcciones.
- Frecuencia de operación hasta 3 MHz.
- Memoria interna RAM y EEPROM.
- Memoria interna ROM o EPROM.
- Puerto serie síncrono (SCI).
- Puerto serie asíncrono (SPI).
- Convertidor Analógico/Digital (A/D) de 8 bits y 8 canales de entrada.
- Temporizador de 16 bits:
  - Tres canales de captura de entrada (IC).
  - Cuatro canales de comparación de salida (OC).
  - Un canal seleccionable como cuarto IC o quinto OC.
  - Acumulador de pulsos de 8 bits.
  - Circuito de interrupciones en tiempo real.
- 38 terminales de propósito general: 16 entrada/salida, 11 de entrada y 11 de salida.
- Terminales para requerimiento de interrupción mascarable (IRQ) y no mascarable XIRQ.

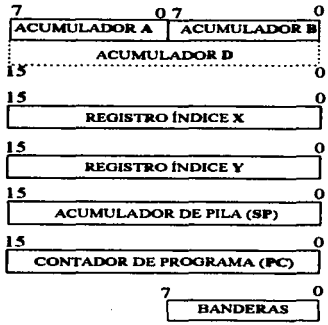


Figura 2.3.- Modelo para el programador del microcontrolador M68HC11 integrado por los registros internos de la CPU.

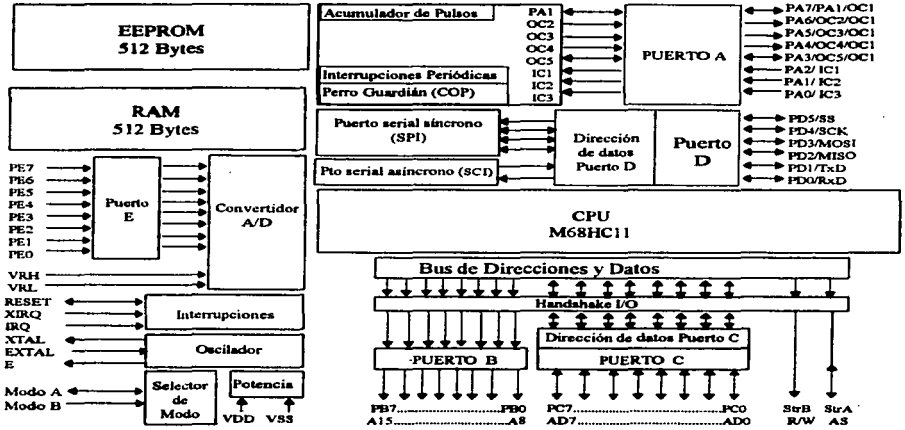


Figura 2.4.- Diagrama interno a bloques del microcontrolador MC68HC11E1.

En la Figura 2.3 se nuestra el modelo para el programador formado por los registros internos de la CPU. En la Figura 2.4 se muestra el diagrama a bloques de la estructura interna del MC68HC11E1.

2.2.2 Modos de operación.

El M68HC11 puede trabajar en 4 modos de operación, cada uno de los cuales está determinado por los valores que presentan las terminales MODA y MODB durante el reset.

El modo single-chip no requiere bus de datos ni de direcciones, por lo que las terminales involucradas para generar estas señales quedan disponibles como puertos de entrada/salida de propósito general, por lo que todo el software necesario para controlar el microcontrolador queda contenido en las memorias internas de éste. La particularidad del modo bootstrap consiste en que el programa bootloader contenido en la ROM interna del microcontrolador es ejecutado para establecer comunicación vía puerto serial con otro dispositivo.

En modo expandido (expanded) sí se generan tanto el bus de datos como el de direcciones, lo cual le permite al microcontrolador tener acceso a circuitos periféricos externos (RAM, EPROM, etc.). El modo test es la operación especial del modo expandido el cual le permite al usuario tener acceso a características internas del microcontrolador.

La Tabla 2.3 muestra la polarización que deberán tener las terminales MODA y MODB para que el microcontrolador opere correctamente en el modo seleccionado.

MODB	MODA	MODO DE OPERACIÓN
1	0	Single-Chip
1	1	Expanded
0	0	Bootstrap
0	1	Special Test

Tabla 2.3

El mapa de memoria en modo expandido para el MC68HC11E1 es el que se muestra en la Figura 2.5, cada tipo de microcontrolador tiene un mapa de memoria asociado dependiendo del modo de operación en el que esté operando. Los bloques marcados como externos son espacios de memoria disponibles, por lo que el usuario los puede utilizar, dependiendo de su aplicación específica, para direccionar memoria (RAM, EPROM, etc.) o bien puertos de entrada/salida externos.

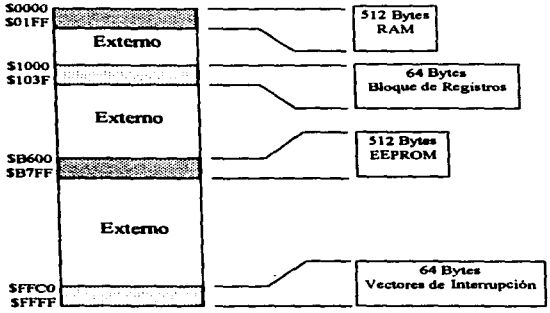


Figura 2.5.- Mapa de memoria del MC68HC11E1 en modo expandido.

2.2.3 Reset e interrupciones.

Las operaciones de reset e interrupciones cargan al program counter (PC) con un vector que apunta a una nueva localidad donde se encuentra la instrucción que va a ser ejecutada. Hay cuatro tipos de reset, el de mayor prioridad es el "power on reset (POR)" que ocurre al energizar el sistema, después de un retardo de 4064 ciclos de reloj el program counter es cargado con el vector $FFFE-$FFFF, el cual contiene la dirección de la primera instrucción del programa a ejecutar. El segundo tipo de reset es el "external reset", que ocurre cuando la terminal física Reset cambia de nivel bajo a alto, inmediatamente después el PC es cargado con el vector $FFFE-$FFFF. El tercer tipo de reset es el "clock monitor fail", el cual se activa o desactiva con el bit de control CME

contenido en el registro OPTION; si está habilitado (CME=1) y la frecuencia del reloj baja a menos de 10 KHz, esto es detectado como error y el reset es generado, en este caso el vector es $FFFC-$FFFD. El último tipo de reset es el "computer operating properly", éste se activa o desactiva con el bit de control NOCOP contenido en el registro CONFIG; si está habilitado (NOCOP=0) y se pierde la secuencia en la ejecución de las instrucciones de tal forma que no se realice a tiempo el reset del "COP timer", el sistema es inicializado con el nuevo vector $FFFA-$FFFB.

La CPU en un microcontrolador ejecuta instrucciones en forma secuencial. En muchas aplicaciones es necesario ejecutar un grupo de instrucciones en respuesta al requerimiento de algún dispositivo periférico. Las interrupciones proveen la forma de suspender la ejecución normal del programa de tal manera que la CPU quede libre para atender dicho requerimiento. Después de que una interrupción ha sido atendida, la CPU continúa con la ejecución normal del programa. El grupo de instrucciones ejecutadas en respuesta a una interrupción es llamado rutina de servicio a interrupción (interrupt service routine).

Las interrupciones de hardware son requeridas al microcontrolador a través de las terminales IRQ y XIRQ, éstas se habilitan por software con los bits tipo máscara I y X del CCR (Condition Code Register). Las interrupciones no mascarables (XIRQ) no pueden deshabilitarse con instrucciones de software una vez que el bit X se ha puesto en cero. Las interrupciones mascarables si se pueden habilitar y deshabilitar cuantas veces sea necesario cambiando el valor del bit I. Las interrupciones generadas a través de la terminal XIRQ tienen mayor prioridad sobre las que se generan en la terminal IRQ, esto significa que aún cuando la CPU esté atendiendo una interrupción mascarable (IRQ) suspenderá el proceso para ir a atender una interrupción que en ese momento se solicite a través de la terminal XIRQ.

El proceso que sigue la lógica del microcontrolador cuando hay requerimiento de interrupciones es el siguiente: la instrucción que se está ejecutando en ese momento se termina normalmente; se almacena en la pila (stack) el contenido de todos los registros (PCL, PCH, IYL, IYH, IXL, IXH, B, A, CCR); si fue requerida una interrupción XIRQ

los bits I y X se ponen en 1 para inhibir interrupciones posteriores y el PC se carga con el vector $FFF4-$FFF5, si son tipo IRQ sólo el bit I se pone en 1 y el PC se carga con el vector que corresponda al tipo de interrupción requerida siguiendo el orden de prioridades establecido. La ejecución de la instrucción RTI causa que los valores de los registros almacenados previamente en el stack sean recuperados y de esta manera la CPU continúa con la secuencia normal del programa como si no hubiese habido interrupción alguna.

2.2.4 Temporizador.

El temporizador está basado en un contador de 16 bits de carrera libre programable en 4 escalas. Además de cumplir con la característica de puerto entrada/salida de propósito general, el puerto A tiene asociadas las funciones del temporizador.

Se tienen 3 funciones de captura (input-capture function) independientes PA0/IC3, PA1/IC2, PA2/IC1 y una más configurable PA3/OC5/IC4/OC2. Son usadas para almacenar automáticamente el tiempo en el que un evento externo ocurre, es decir, almacena en el registro de captura correspondiente (TIC1, TIC2, TIC3, TI4/O5) el contenido del contador de carrera libre TCNT, en el instante que detecta el cambio de estado seleccionado en la respectiva terminal de entrada. Esta función tiene la opción de habilitar la generación de interrupción como acción de respuesta automática.

Se tienen 5 funciones de comparación de salida (output-compare function) configurables PA6/OC2/OC1, PA5/OC3/OC1, PA4/OC4/OC1, PA3/OC5/IC4/OC1, utilizadas para generar señales de salida o retardos de software, esto es, cuando el contenido de cualquiera de los registros de 16 bits TOC1, TOC2, TOC3, TOC4 o TI4/O5 coincide con el contenido del registro TCNT se ejecuta la acción indicada por la función correspondiente. Esta acción puede ser un requerimiento de interrupción o un cambio de estado en las respectivas terminales de salida.

Tiene un acumulador de pulsos de 8 bits que puede ser configurado para operar como un simple contador de eventos o bien, para contar el tiempo total en que PAI permanece activa.

El M68HC11 cuenta además con un convertidor analógico-digital de 8 canales 8 bits, una interface serial síncrona y una interface serial asíncrona, de estos subsistemas no se habla en este trabajo porque no son utilizados en el diseño de la IEPP.

2.3 Diagrama a bloques de los módulos que conforman al sistema.

La utilización de un microcontrolador en el diseño fue necesaria para realizar en forma automática la conmutación entre los ocho puertos de salida y tener así, una interface completamente autónoma y portable, es decir, que se pueda conectar directamente en el puerto paralelo de cualquier computadora personal. La información que recibe la IEPP de la PC se almacena primero en la memoria RAM externa y posteriormente el microcontrolador se encarga de establecer comunicación con el dispositivo conectado en el puerto de salida de la IEPP para transmitirle la información correspondiente.

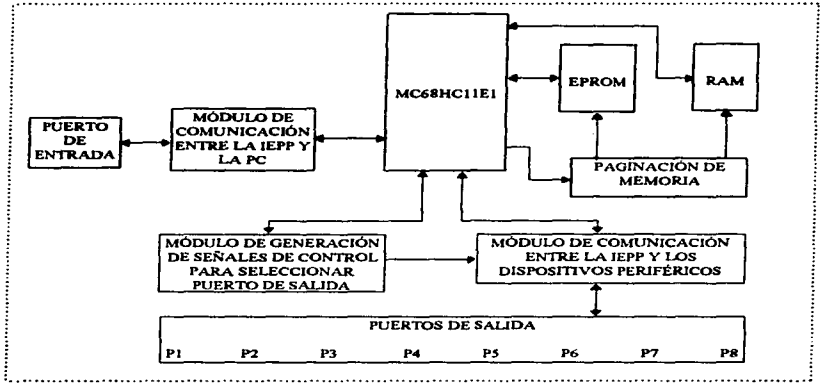


Figura 2.5.- Diagrama a bloques de la Interface Electrónica para Puerto Paralelo

Los módulos que conforman el sistema así como la interconexión entre cada uno de ellos se muestra en el diagrama a bloques de la Figura 2.5.

Al principio de este capítulo se dio una descripción general de las características que debe cubrir el diseño de la IEPP. Independientemente del microcontrolador que se utilice en el diseño, éste deberá contener cada uno de los módulos indicados en el diagrama a bloques de la figura 2.5. La función que desempeña cada uno de ellos se describe en los siguientes párrafos.

2.3.1 Puerto de entrada.

El puerto de entrada se conecta directamente al puerto paralelo de una computadora personal uniendo una a una las terminales de ambos puertos, mediante un cable de 25 hilos con conectores DB25 macho en cada extremo. A través de este puerto la IEPP se mantiene en comunicación física con la PC.

2.3.2 Módulo de comunicación entre la IEPP y la PC.

La computadora ve a la IEPP como cualquier periférico con el que se comunica mediante el protocolo de comunicación estándar para puerto paralelo (Standard Parallel Port). En este módulo están presentes todas las señales necesarias para establecer la comunicación con la PC, tanto la recepción de datos como la generación de las señales de estado y de control involucradas en el proceso de recepción de datos o caracteres transmitidos por la PC.

2.3.3 Microcontrolador M68HC11E1

El M68HC11E1 es el elemento inteligente de la IEPP, la razón fundamental por la que la interface está diseñada con un microcontrolador es por tener una conmutación automática entre los 8 puertos de salida. El microcontrolador se encarga de llevar a cabo la comunicación con la PC para recibir los datos que ésta transmite a través del puerto paralelo y también de establecer el enlace de comunicación con los dispositivos periféricos conectados en los puertos de salida para transferir la información contenida en la memoria RAM externa.

2.3.4 Memoria RAM.

El microcontrolador M68HC11E1 tiene 512 bytes de memoria RAM interna. Como una de las características de la interface es de que esta cuente con un buffer de memoria RAM apreciable para almacenar los datos destinados a cada uno de los puertos de salida, y además que el sistema sea autónomo, es necesario que el microcontrolador opere en modo expandido. De esta forma el M68HC11E1 podrá direccionar 32 Kbytes de RAM externa. Este espacio de 32 Kbytes está distribuido en secciones de 4 Kbytes, cada uno de ellos asociado a su respectivo puerto de salida, con el propósito de almacenar la información que va dirigida hacia cada uno de los periféricos que estén conectados en ese momento a dichos puertos.

2.3.5 Memoria EPROM.

La memoria EPROM externa se ubica en los últimos 8 Kbytes del mapa de memoria ($E000-$FFFF), por especificaciones del fabricante. En ella se almacenan los vectores de reset e interrupciones y el código del programa que va a controlar al microcontrolador.

2.3.6 Paginación de memoria.

Al realizar una decodificación utilizando las líneas de direcciones, es posible generar señales de habilitación que se activen al accesar una determinada localidad de memoria, si se involucran todas las líneas de dirección. O bien al accesar un determinado bloque de memoria, si se involucran sólo algunas de las líneas de dirección. De esta manera, al conectar adecuadamente estas señales de habilitación en otros circuitos periféricos (RAM, EPROM, decodificadores, etc.), el microcontrolador tendrá acceso a los mismos direccionándolos directamente.

El microcontrolador debe operar en modo expandido, ya que es el modo de operación donde se generan los 16 bits de direcciones y los 8 de datos. Dadas las características del sistema es conveniente paginar este espacio de memoria en bloques de 8 Kbytes, para ello sólo se utilizan las 3 líneas de direcciones más significativas (A13, A14 y A15). Se

tendrán 8 líneas de habilitación, cada una de las cuales permite activar a un circuito periférico dentro de un espacio de 8 Kbytes consecutivos.

2.3.7 Módulo de generación de señales de control para seleccionar puerto de salida.

Como son 8 los puertos de salida de la IEPP, se necesitan 3 señales de control (**Slc1, Slc2, Slc3**) para generar las 8 señales que habilitan a cada uno de los puertos de salida. Las tres señales de control se pueden generar utilizando directamente tres puertos de salida del microcontrolador conectadas a un circuito decodificador 3 a 8 como el 74HC138. Otra forma es utilizar un circuito decodificador 3 a 8 con latches de direccionamiento como el 74HC137, en cuyo caso para habilitarlo es necesario utilizar una de las 8 líneas de habilitación generadas en el módulo de paginación de memoria.

Slc3	Slc2	Slc1	Puerto activo
0	0	0	P1
0	0	1	P2
0	1	0	P3
0	1	1	P4
1	0	0	P5
1	0	1	P6
1	1	0	P7
1	1	1	P8

Tabla 2.4.- Selección de puertos de salida

En la tabla 2.4 se indica que puerto permanece activo de acuerdo al estado que presentan las señales de control **Slc1, Slc2** y **Slc3** generadas para este propósito.

2.3.8 Módulo de comunicación entre la IEPP y los dispositivos periféricos.

Para establecer un enlace de comunicación adecuado entre el sistema y cada uno de los periféricos, el sistema activa sólo uno de los puertos de salida a la vez. El proceso de transmisión de información hacia los dispositivos periféricos, se puede desglosar en los siguientes bloques:

a) Detección de señales de estado de los dispositivos periféricos.

b) Transmisión de señales de control hacia los dispositivos periféricos.

c) Transmisión de caracteres hacia los dispositivos periféricos.

A continuación se describe en que consiste cada una cada uno de estos bloques.

2.3.8.1 Detección de señales de estado de los dispositivos periféricos.

Una vez que el microcontrolador establece comunicación con el puerto deseado, el sistema es capaz de establecer comunicación con el periférico conectado en dicho puerto. Las señales **PE**, **ERROR**, **ACKNLG**, **SLCT** y **BUSY** de cada uno de los periféricos son señales que deben conectarse a la IEPP como señales de entrada, a través de un circuito que permita seleccionar sólo las que correspondan con el puerto activo en ese momento, por ejemplo un circuito multiplexor de 8 canales 74HC151. Al M68HC11E1 se conectan las salidas de estos circuitos, ya sea utilizando directamente puertos de entrada del microcontrolador o bien (con excepción de la señal de **ACKNLG**), utilizando un puerto de entrada externo como el circuito 74HC373.

2.3.8.2 Transmisión de señales de control hacia los dispositivos periféricos.

Las señales de **STROBE**, **AUTO FEED**, **INIT** y **SLCT IN** son señales de control que se conectan a las respectivas terminales en cada uno de los 8 puertos de salida de la IEPP. La señal de **STROBE** se genera con un puerto de salida del microcontrolador. Las señales **AUTO FEED**, **INIT** y **SLCT IN** no intervienen en el proceso de transmisión de cada caracter, por esta razón el microcontrolador no las genera directamente, es decir, el estado de estas señales es el que se encuentra presente en el puerto paralelo de la PC.

Para reflejar el estado de estas cuatro señales solamente en las terminales correspondientes del puerto activo en ese momento se utilizan nuevamente las señales de control **Slc1**, **Slc2**, **Slc3**. Esto se explica con detalle en el siguiente capítulo.

2.3.8.3 Transmisión de caracteres hacia los dispositivos periféricos.

El propósito de este bloque es que el caracter transmitido por la IEPP sea recibido solamente por el dispositivo periférico conectado en el puerto de salida seleccionado por el microcontrolador.

El M68HC11 tiene un determinado número de terminales que operan como puertos de salida o de entrada, en este caso particular no es posible asignar 8 de las terminales de salida para los 8 bits del caracter que se va a transmitir. Será necesario agregar un circuito que pueda operar como puerto de salida. Para generar una línea de habilitación que se active exclusivamente en una cierta localidad de memoria, sería necesario involucrar a todas las 16 líneas de direcciones y por lo tanto se agregarían más elementos de hardware al sistema, lo cual resulta innecesario si se dispone ya de un módulo de paginación de memoria.

De las 8 líneas de habilitación generadas en el módulo de paginación de memoria, se puede utilizar una de las líneas para habilitar un circuito que pueda operar como puerto de salida, por ejemplo el 74HC273. Basta con escribir el dato en cualquier localidad de memoria comprendida dentro del bloque de 8 Kbytes que corresponde a la línea de habilitación utilizada para que el circuito lo capture en sus 8 líneas de salida de datos. Este dato debe ser enrutado posteriormente hacia el puerto de salida por el que se desea transmitir y retenido en las líneas de datos correspondientes hasta que el dispositivo periférico conectado en dicho puerto lo reciba.

2.3.9 Puertos de salida

La IEPP cuenta con 8 puertos de salida, a cada uno de ellos se puede conectar un periférico, generalmente dispositivos de impresión, aunque con algunas restricciones también pueden conectarse computadoras personales y sistemas integrados con motores de pasos (brazos mecánicos, bandas transportadoras etc.).

CAPÍTULO 3

DISEÑO DE CADA MÓDULO

En el capítulo anterior se describió la función que desempeña cada uno de los módulos que conforman el sistema, en este capítulo se presenta y se analiza el hardware implementado en el diseño de la IEPP. Debido a que el desarrollo de la interface fue realizado por módulos, esta será la forma en la que se analizará la circuitería que conforma a la IEPP. Se parte de la arquitectura básica de operación del microcontrolador M68HC11 en modo expandido, a continuación se agrega el circuito de paginación de memoria y las memorias externas (RAM y EPROM). Una vez que se tiene el modelo básico para el diseño de la IEPP, se irán incorporando los elementos que satisfagan las características descritas para cada uno de los módulos restantes.

3.1 Esquema básico de operación en modo expandido del M68HC11.

Motorola propone en el "HC11 Reference Manual" el diagrama de conexiones para la operación del microcontrolador tanto en modo expandido como modo prueba (test), sin embargo; no se adopta íntegramente en el diseño de la IEPP, las razones se exponen a continuación.

En este diagrama la EPROM aparece 2 veces en el mapa de memoria: en $A000-$BFFF cuando A14 tiene nivel bajo y en $E000-$FFFF cuando A14 tiene nivel alto. La IEPP opera siempre en modo expandido, por lo tanto la EPROM se direcciona únicamente en el espacio $E000-$FFFF. Se tiene también un circuito decodificador 74HC138 utilizado para direccionar 4 RAM de 8K por 8 cada una, ocupando los siguientes espacios de memoria: $0000-$1FFF, $2000-$3FFF, $4000-$5FFF, $6000-$7FFF. En la IEPP es necesario direccionar 32 Kbytes consecutivos en un espacio libre del mapa de memoria, razón por la cual se hacen las modificaciones a este circuito para tener 8 páginas de 8

Kbytes cada una, esto se explica con más detalle en párrafos posteriores.

El circuito de reset MC34064 no se consigue fácilmente en el mercado, es por esto que se utiliza un circuito de reset diferente, éste se presenta y se discute posteriormente.

Partiendo del esquema básico propuesto por el fabricante y los cambios antes mencionados se obtiene el esquema básico para el desarrollo de la IEPP, el cual se muestra en la Figura 3.2. A continuación se describe la operación de los módulos que conforman este circuito.

1) Alimentación de voltaje.- El sistema se alimenta con un voltaje de 9V de corriente directa proporcionados por rectificador de voltaje. Para disminuir este voltaje a 5 VDC (voltaje de polarización de todos los elementos del sistema), se utiliza un regulador de voltaje LM7805.

2) Circuito de reset.- Un nivel bajo en la terminal de RESET (pin 17) inicializa al M68HC11E1, el circuito mostrado en la figura 3.1 es un circuito de autoreset que permite asegurar la inicialización correcta del microcontrolador. Al energizar el sistema el transistor está en saturación, por lo que el voltaje en la terminal 17 (RESET) es bajo. Conforme el capacitor C6 se va cargando el transistor va entrando a la región de corte, teniendo así un voltaje alto en la terminal 17. El interruptor provee el mecanismo para realizar un reset al microcontrolador en cualquier momento que se desee.

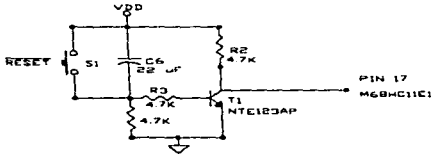


Figura 3.1.- Circuito de Autoreset

3) Demultiplexaje de las señales de direcciones y datos del puerto C.- En modo expandido, el puerto C es utilizado para generar los 8 bits de datos y los 8 bits bajos de direcciones. La señal AS (pin 4) permite realizar el demultiplexaje de estas

señales. Para ello se utiliza un circuito 74HC373 (U15) conectado como se muestra en la Figura 3.2. En la primer mitad del ciclo de reloj (E) el microcontrolador genera los 8 bits de direcciones y mantiene el nivel de la señal AS en alto para habilitar al 74HC373, una vez que éste ha capturado los 8 bits, el nivel de la señal AS cambia a nivel bajo quedando el 74HC373 desactivado. Durante la segunda mitad del ciclo de reloj se generan los 8 bits de datos.

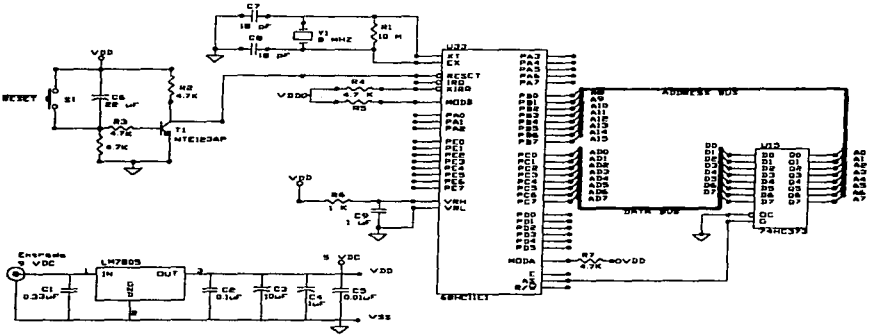


Figura 3.2.- Esquema básico para el desarrollo del diseño de la IEPP.

En la Figura 3.3 se muestra un esquema que incluye además del circuito de la Figura 3.2, el módulo de paginación de memoria y los circuitos auxiliares utilizados para la conexión de RAM y EPROM externas.

En modo expandido el microcontrolador tiene la capacidad de direccionar un espacio de 64 Kbytes, el cual está determinado por los 16 bits de direcciones generados en este modo de operación. Para paginar este espacio de memoria se utiliza el circuito decodificador 74HC138 (U8). Conectando las tres líneas de direcciones más significativas (A13, A14 y A15) a las terminales de entrada de este circuito, se generan 8 señales de habilitación en las salidas correspondientes. La señal de reloj E se conecta a la terminal de habilitación G1 para activar el circuito sólo en la primer mitad del ciclo.

Cada una de las señales de salida de este circuito se activa al escribir o leer a una localidad de memoria comprendida dentro de un bloque determinado de 8 Kbytes consecutivos. En la siguiente tabla de estados se indica el espacio de memoria que direcciona cada una de las terminales habilitadas del circuito U8.

G1 (E)	G2* (GND)	C (A15)	B (A14)	A (A13)	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	ESPACIO DE MEMORIA
•	H	•	•	•	H	H	H	H	H	H	H	H	NINGUNO
L	•	•	•	•	H	H	H	H	H	H	H	H	NINGUNO
H	L	L	L	L	L	H	H	H	H	H	H	H	$0000-$1FFF
H	L	L	L	H	H	L	H	H	H	H	H	H	$2000-$3FFF
H	L	L	H	L	H	H	L	H	H	H	H	H	$4000-$5FFF
H	L	L	H	H	H	H	H	L	H	H	H	H	$6000-$7FFF
H	L	H	L	L	H	H	H	H	L	H	H	H	$8000-$9FFF
H	L	H	L	H	H	H	H	H	H	L	H	H	$A000-$BFFF
H	L	H	H	L	H	H	H	H	H	H	L	H	$C000-$DFFF
H	L	H	H	H	H	H	H	H	H	H	H	L	$E000-$FFFF

Tabla 3.1.- Tabla de estados para el circuito de paginación de memoria U8.

El espacio de 64 Kbytes direccionables por el microcontrolador ($0000-$FFFF), puede distribuirse de acuerdo a las necesidades específicas del diseño, respetando obviamente, los espacios que ocupan la memoria interna (RAM y EEPROM) y los bloques de registros y vectores de interrupción del microcontrolador.

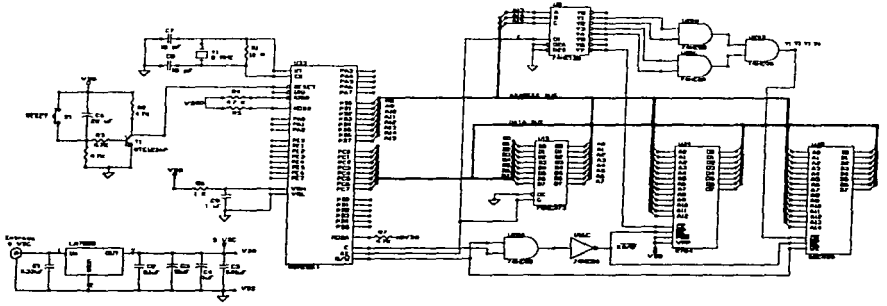


Figura 3.3.- Incorporación del módulo de paginación de memoria y circuitos adicionales para el manejo de RAM y EPROM externas.

La IEPP utiliza el mapa de memoria mostrado en la Figura 3.4, en éste se indica el espacio de 32 Kbytes ocupado por la RAM externa, el puerto Y6 y los 8 Kbytes ocupados por la EPROM externa.

Para sincronizar la habilitación de los circuitos periféricos con los procesos de escritura y lectura de datos realizados por el microcontrolador se utilizan las señales E y R/W conectadas a una compuerta NAND como se muestra en la Figura 3.3. La señal resultante se conecta a las terminales OE (Output Enable) de los circuitos de memoria RAM y EPROM.

El circuito 2764 (U34) es una memoria tipo EPROM de 8 Kbytes por 8, éste contiene el código del programa que controla al microcontrolador, así como el valor de los vectores de reset e interrupciones utilizados, por esta razón este circuito debe quedar habilitado en el último bloque de 8 Kbytes del mapa de memoria ($E000-$FFFF). Esto se realiza conectando directamente la salida Y7 del circuito decodificador U8 a la terminal CE (Chip Enable) de la EPROM.

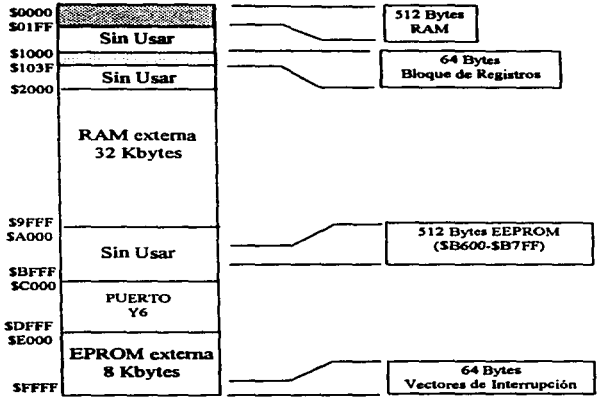


Figura 3.4.- Mapa de memoria manejado en la interface para puerto paralelo (IEPP).

El circuito 62256 (U35) es una memoria tipo RAM de 32 Kbytes por 8, para habilitarlo dentro del bloque indicado en el mapa de memoria de la Figura 3.4, es necesario sumar las señales Y1, Y2, Y3 y Y4 del circuito decodificador U8 utilizando compuertas AND, la señal resultante se conecta a la terminal CS (Chip Select) de la RAM. Ahora bien, para indicarle que se ha realizado un proceso de escritura de datos dentro del intervalo $2000-$9FFF, se conecta la señal R/W directamente a la terminal WE (Write Enable), como se ilustra en el diagrama de la Figura 3.3.

3.2 Generación de las señales para la selección de los dispositivos periféricos.

Se utilizan 3 líneas de salida del microcontrolador para generar las señales de control **Slc1**, **Slc2** y **Slc3**, estas líneas son respectivamente, PD3, PD4 y PD5. Los circuitos de la tecnología HCMOS tienen en promedio un fanout de 10, sin embargo y aunque no es estrictamente necesario, se optó por proteger las terminales de salida del microcontrolador y las de otros circuitos que presentan demasiada sobrecarga, con circuitos buffer 74HC244. El circuito 74HC138 (U12) se utiliza para decodificar las señales **Slc1**, **Slc2** y **Slc3** y generar las señales de selección (S0-S7), como se muestra en la figura 3.5. La combinación de valores que habilitan a cada una de las señales de selección S0-S7 se muestran en la Tabla 3.2.

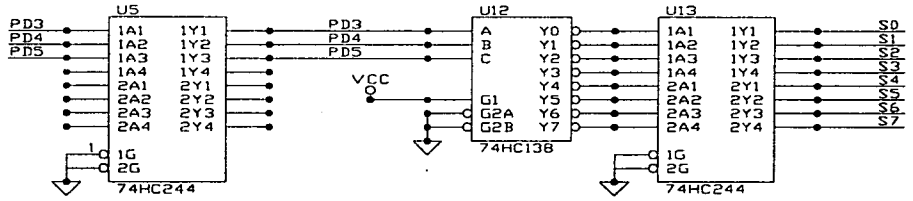


Figura 3.5.- Diagrama eléctrico del módulo de generación de señales de selección S0-S7

Slc3 PD5	Slc2 PD4	Slc1 PD3	Señales de Selección
0	0	0	S0
0	0	1	S1
0	1	0	S2
0	1	1	S3
1	0	0	S4
1	0	1	S5
1	1	0	S6
1	1	1	S7

Tabla 3.2. Generación de las señales S0-S7.

3.3 Módulo de comunicación entre la PC y la IEPP

Este módulo se encarga únicamente de establecer comunicación entre la PC y la IEPP como se indica en el diagrama de la figura 3.6.

Figura 3.6.- Flujo de datos entre la PC y la IEPP.

Cada vez que la PC solicita transmitir un caracter lo indica poniendo en nivel bajo la señal de **STROBE**, aproximadamente 10 microsegundos, al utilizar esta señal para generar una interrupción al microcontrolador se asegura que el sistema atenderá cualquier requerimiento solicitado por la PC. Para que el sistema responda de acuerdo a las normas establecidas por el protocolo de comunicación "Standard Parallel Port", la manipulación de la mayoría de las señales se realiza a nivel de software, exceptuando la señal de **BUSY** que involucra el uso de algunos circuitos electrónicos adicionales. Esta señal debe cambiar de nivel bajo a nivel alto en el instante en que el sistema detecta un nivel bajo en la señal de **STROBE**, por software resulta imposible satisfacer este requerimiento considerando que cada instrucción tiene una duración mínima de 2 ciclos de reloj (0.5 microsegundos) y el período de latencia de atención a interrupción es de aproximadamente 8 microsegundos.

Las señales de entrada hacia la computadora **PE**, **ERROR** y **ACKNLG** se generan directamente con las terminales PD2, PD1 del puerto D y PA4 del puerto A respectivamente. Para la generación de la señal de **BUSY** se involucran las líneas PA3, PA5 y PA6 de puerto A, como se indica en el diagrama de la figura 3.7.

Para iniciar un ciclo de transmisión las terminales deben tener el estado que se indica en la siguiente tabla:

TERMINAL	ESTADO
PA3	BAJO
PA4	ALTO
PA5	ALTO
PA6	BAJO
PD1	ALTO
PD2	BAJO

Lo que significa que la PC registra un estado en las señales de **BUSY**, **ACKNLG**, **PE** y **ERROR**, bajo, alto, bajo y alto respectivamente. Esto se puede observar fácilmente en el diagrama de la figura 3.7.

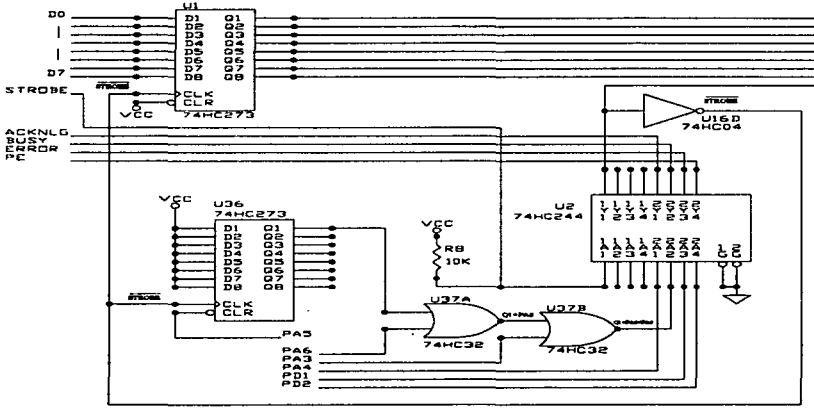


Figura 3.7.- Módulo de comunicación entre la PC y la IEPP.

Las líneas de datos del puerto paralelo de la PC (D0-D7) se conectan a las correspondientes entradas (D0-D7) de un flip-flop D 74HC273 (U1). Un flanco de subida en la terminal CLK de este circuito genera que el estado presente en ese momento en las terminales de estrada D0-D7 se reflejen en las terminales de salida Q0-Q7. Las normas del "Centronics Parallel Interface" establecen que el caracter a transmitir está presente en las líneas de datos (en este caso del puerto paralelo de la PC) 0.5 microsegundos antes de que la señal de **STROBE** baje. Por lo tanto al invertir esta señal de **STROBE** se tendrá el flanco de subida necesario en la terminal CLK de U1 para capturar el caracter. Las terminales Q0-Q7 de U1 se conectan a las terminales del puerto E del microcontrolador, para posteriormente leer el caracter en la rutina de atención a interrupción, de la cual se habla con detalle en el siguiente capítulo.

La señal de **STROBE** se utiliza también para activar la señal de **BUSY**, esto lo realiza el flip-plop D 74HC273 (U36) en conjunto con 2 compuertas OR del circuito 74HC32 (U37). La señal de **STROBE** invertida se conecta a la terminal CLK del circuito U36, con un flanco de subida en esta terminal el nivel alto de D0 se refleja en la salida Q0, logrando por lo tanto que la señal de **BUSY** se ponga en nivel alto. Hasta este momento se ha cumplido con el primer requisito, que es precisamente que la IEPP ponga el nivel de la señal de **BUSY** en alto cuando sensa un nivel bajo de la señal de **STROBE**.

Dentro de la rutina de atención a interrupción, la terminal PA6 que inicialmente tiene un nivel bajo se cambia a un nivel alto, a continuación se genera un pulso bajo por PA5 que está conectada a la terminal CLR del 74HC273 (U36), con el propósito de inicializar este circuito y dejar la salida Q0 en nivel bajo. En este momento quien sostiene en nivel alto la señal de **BUSY** es la terminal PA6.

Al finalizar la rutina de atención de interrupción la terminal PA6 retorna a su estado inicial (bajo). Quedando lista la IEPP para el siguiente ciclo de recepción.

La terminal PA3 se utiliza para poner en nivel alto la señal de **BUSY** durante el proceso de transmisión de datos de la IEPP hacia los diferentes dispositivos periféricos conectados. Esto con el propósito de indicarle a la PC que la IEPP está ocupada y por lo tanto no puede recibir información.

La señal de **ACKNLG** es activa baja y se genera como ya se mencionó, con la terminal PA4. La IEPP genera un pulso bajo en la señal **ACKNLG** al finalizar un ciclo re recepción, para indicarle a la PC que el caracter transmitido se recibió con éxito.
Durante el proceso de recepción de caracteres la IEPP mantiene en nivel bajo la señal **PE** (PD2) y en nivel alto la señal de ERROR (PD1) ya que son requisitos necesarios para que la PC transmita información.
Las terminales PD1 y PD2 se utilizan además para transmitir hacia la PC el estado de las señales **PE** y **ERROR** presentes en los 8 puertos de salida de la IEPP. Este proceso se explica con detalle en el siguiente capítulo.

3.4 Módulo de comunicación entre la IEPP y los dispositivos periféricos.

Para que la IEPP establezca comunicación con los dispositivos periféricos que tiene conectados, necesita sensar las señales de estado (**AKCNLG**, **BUSY**, **SELECT**, **PE** y **ERROR**) en cada uno de los puertos de salida, generar las señales de control (**STROBE**, **INIT**, **SLCT IN** y **AUTO FEED**) para cada uno de los puertos de salida y finalmente, direccionar el caracter que va a transmitir hacia el puerto de salida que haya indicado el usuario. Cada una de estos bloques se discute con detalle a continuación.

3.4.1 Detección de señales de estado de los dispositivos periféricos.

En los circuitos de tecnología CMOS deben polarizarse adecuadamente todas las terminales de entrada no usadas, evitando así ruido y oscilaciones que conllevan a un aumento en el consumo de corriente. El valor de la resistencia de pullup o pulldown típico es de 10 KΩ. Para sensar las señales de estado de los 8 puertos de salida se utilizan circuitos multiplexores de 8 canales 74HC151 conectados como se indica en la figura 3.8. Cuando alguno de los puertos de salida de la IEPP no tiene ningún dispositivo periférico conectado, las entradas de estos circuitos quedan protegidas porque están debidamente polarizadas.

Las señales **Slc1 (PD3), Slc2 (PD4)** y **Slc3 (PD5)** determinan cual de los puertos de salida está activo, y se utilizan como ya se mencionó en párrafos anteriores, para conmutar entre los diferentes puertos de salida de la IEPP.

Las señales de estado **ACKNLG, BUSY, SLCT, PE** y **ERROR** de los 8 puertos de salida se agrupan y se conectan a las respectivas entradas D0-D7 de los multiplexores 74HC151, como se indica en la figura 3.8. El microcontrolador detecta únicamente el estado de las señales de **ACKNLG, BUSY, PE** y **ERROR** del puerto de salida indicado por **Slc1** (PD3), **Slc2** (PD4) y **Slc3** (PD5). Para ilustrar esto de una manera mas clara, a continuación se presenta la tabla de estados del circuito U21 el cual multiplexa la señal **PE** de los 8 puertos de salida.

C (PD5)	B (PD4)	A (PD3)	G (GND)	Y (SEÑAL PE)
X	X	X	H	L
L	L	L	L	PE del puerto P1
L	L	H	L	PE del puerto P2
L	H	L	L	PE del puerto P3
L	H	H	L	PE del puerto P4
H	L	L	L	PE del puerto P5
H	L	H	L	PE del puerto P6
H	H	L	L	PE del puerto P7
H	H	H	L	PE del puerto P8

Tabla 3.3. Tabla de estados del circuito 74HC151 (U21).

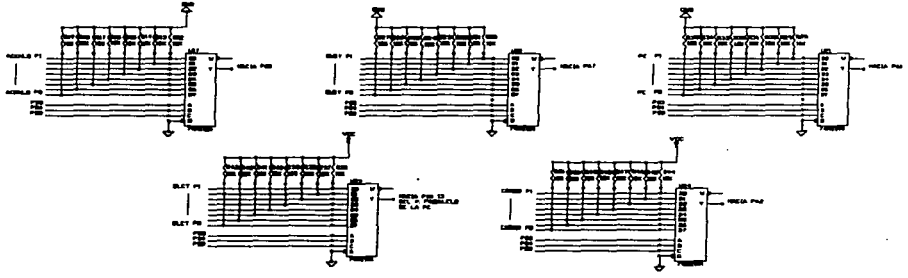


Figura 3.8.- Detección de las señales de estado de los puertos de salida de la IEPP.

3.4.2 Transmisión de señales de control hacia los dispositivos periféricos.

Las señales de control que hay que transmitir hacia los dispositivos periféricos son: **STROBE, SLCT IN, INIT** y **AUTO FEED**. De éstas sólo la señal de **STROBE** interviene en el proceso de transmisión de datos, por esta razón el microcontrolador tiene control directo de ella a través de la terminal de salida PD0. El estado de las señales **SLCT IN, INIT** y **AUTO FEED** del puerto paralelo de la PC, es demultiplexado con compuertas OR hacia el puerto activo como se indica en el diagrama de la figura 3.9. El mismo método se utiliza con la señal de **STROBE.**

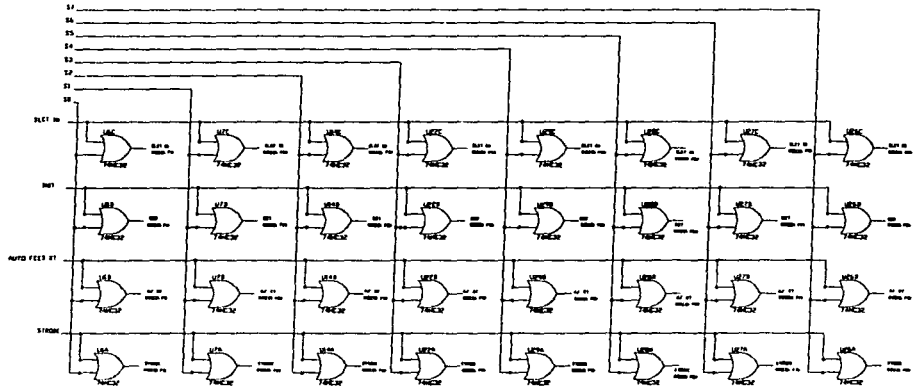


Figura 3.9.- Transmisión de las señales de control hacia los puertos de salida de la IEPP.

3.4.3 Transmisión de caracteres hacia los dispositivos periféricos.

La información contenida en los buffer's asociados a los puertos de salida de la IEPP se transmite siguiendo el protocolo de comunicación estándar para puerto paralelo (SPP). Sin embargo se mencionó que la IEPP acepta dispositivos tales como brazos mecánicos

(control de motores de pasos), y computadoras personales, los cuales obviamente no siguen este protocolo de comunicación. La solución simple es identificar si el periférico es un dispositivo que sigue el protocolo de comunicación estándar (generalmente dispositivos de impresión), si es así, la señal de **ACKNLG** en dicho puerto tendrá un nivel alto al inicio de cada ciclo de transmisión.

Cuando no hay dispositivo conectado en un puerto de salida de la IEPP, la señal **ACKNLG** presentará un nivel bajo en todo momento, lo mismo ocurre si el dispositivo es un mecanismo controlado con motores de pasos o una PC.

Cuando la IEPP verifica al inicio del ciclo de transmisión que el periférico conectado sigue el protocolo SPP, espera un pulso bajo en la señal de **ACKNLG** como indicación de que la transmisión del caracter tuvo éxito. En caso contrario la IEPP da por finalizado el ciclo de transmisión sin esperar esta indicación.

Lo anterior significa que aún cuando no haya ningún dispositivo conectado a un puerto de salida determinado, la IEPP transmitirá toda la información contenida en el buffer asociado por dicho puerto.

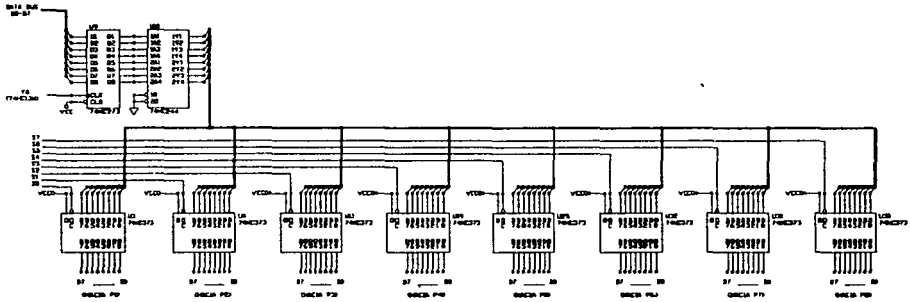


Figura 3.10- Transmisión de caracteres hacia los dispositivos periféricos.

El ciclo de transmisión de un caracter de la IEPP hacia un puerto de salida es el siguiente:

1) Se verifica que la señal **PE** tenga un nivel bajo y la señal de **ERROR** un nivel alto. Si esto no se cumple no se lleva a cabo la transmisión y se volverá a intentar cuando corresponda transmitir nuevamente por este puerto.
2) Si la señal de **ACKNLG** presenta un nivel alto se trata de un dispositivo que se comunica a través del protocolo de comunicación estándar para puerto paralelo (SPP).
3) Se verifica que la señal de **BUSY** presente un nivel bajo.
4) El puerto Y6 del circuito U8 se utiliza como puerto de salida para habilitar el flip-flop D 74HC273 (U9). El caracter a transmitir se almacena en cualquier localidad de memoria comprendida entre $C000-$DFFF, esto provoca que Y6 cambie a nivel bajo y por lo tanto que U9 capture el caracter en sus terminales de salida.
5) El puerto de salida seleccionado, que será aquel por el que se desea transmitir el caracter, determina cual de las señales S0-S7 es activa. Las señales S0-S7 están conectadas respectivamente a las terminales OC de los flip-flop's D (74HC373) como se muestra en la figura 3.10. Un nivel alto en la terminal OC de estos circuitos provoca un estado de alta impedancia en sus terminales de salida. Un nivel bajo en la terminal OC provoca que el estado presente en ese momento en las terminales de entrada de datos (D0-D7) del circuito se refleje en las respectivas terminales de salida (Q0-Q7).
6) Una vez que el caracter está presente en las líneas de datos del puerto de salida por el cual se va a transmitir, el microcontrolador genera un pulso bajo en la señal de **STROBE** para indicarle al dispositivo conectado que hay un dato listo.
7) El ciclo de transmisión finaliza en este momento si el dispositivo no sigue el protocolo de comunicación estándar para puerto paralelo (SPP), de lo contrario la IEPP espera un pulso bajo en la señal de **ACKNLG**.

CAPÍTULO 4

ESTRUCTURA DE LOS PROGRAMAS.

El objetivo de este capítulo es explicar la estructura del software del sistema el cual está formado por dos programas, uno que es ejecutado por el M68HC11E1 en la IEPP y otro que es ejecutado por el microprocesador de una computadora personal. A continuación se presenta la descripción de cada uno de ellos.

4.1 Programa ejecutado por el microcontrolador M68HC11E1.

El código ejecutable de este programa se encuentra grabado en la memoria externa EPROM. Tiene la finalidad de controlar el microcontrolador para que el sistema realice adecuadamente las funciones descritas en el capítulo 2. Este programa está estructurado en cuatro bloques principales:

1) Inicialización del sistema.- Esta sección del programa se encarga de establecer las condiciones iniciales de operación de la IEPP.

2) Recepción de caracteres o datos.- Esta sección del programa es la que se encarga de recibir los datos enviados por la PC a través del puerto paralelo bajo el protocolo de comunicación SPP, de reconocer una palabra de control, y de almacenar los caracteres de información en los buffer's correspondientes de 4 kbytes asociados a cada uno de los puertos de salida de la IEPP.

3) Transmisión de caracteres hacia los dispositivos periféricos.- La información contenida en los buffer's asociados a los puertos de salida de la IEPP es transmitida respectivamente hacia los dispositivos periféricos conectados.

4) Transmisión del estado de los puertos de salida de la IEPP hacia la PC.- El programa de apoyo que se ejecuta en la PC tiene una opción que le permite al usuario saber si los dispositivos periféricos conectados a la IEPP operan correctamente o si se encuentran en estado de error o sin papel. Esta sección del programa verifica si el usuario ha solicitado esta tarea, y de ser así, lee el estado de las señales **PE** y **ERROR** de cada puerto de salida y lo transmite a la PC.

En el diagrama de flujo de la Figura 4.1 se indican las rutinas principales en las que está estructurado el programa ejecutado por el M68HC11E1. Cada una de estas rutinas es explicada con detalle a continuación.

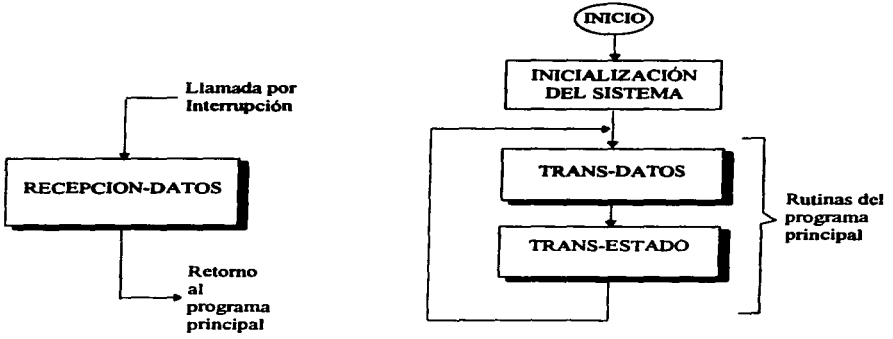


Figura 4.1.- Diagrama de flujo de la estructura del programa principal.

4.1.1 Inicialización del sistema.

Después del reset (Inicio), el cual ocurre cuando se energiza el sistema o bien se presiona el botón externo del "reset", las funciones del microcontrolador quedan

programadas con las condiciones iniciales descritas por el fabricante. Dependiendo de la aplicación, el usuario decide si cambia o no estas condiciones de operación y es precisamente en esta sección del programa donde se realiza esta tarea.

Del bloque de 64 bytes asignado a los registros de programación del M68HC11E1, sólo se programan los que están asignados a las funciones que utiliza la IEPP y siempre y cuando sea necesario modificar las condiciones de reset impuestas por el fabricante en cada uno de los registros. El contenido de la mayoría de los registros se puede modificar en cualquier momento, pero existen algunos registros especiales cuyo contenido sólo puede ser cambiado durante los primeros 64 ciclos de reloj después del reset.

Como parte de la inicialización del sistema es necesario también inicializar en cero todas las variables, seleccionar uno de los 8 puertos de salida de la IEPP como puerto activo y finalmente habilitar las interrupciones mascarables a la CPU del microcontrolador.

El grupo de instrucciones que se presenta a continuación constituye el bloque de inicialización del sistema, aunque en el Apéndice A se encuentra un listado completo del programa fuente, resulta más cómodo para la lectura del documento tener acceso directo a este grupo de instrucciones. En los párrafos siguientes se explica con detalle la programación de los registros involucrados en las funciones del microcontrolador que se utilizan en la IEPP.

```
a)      LDS     #STACK          ; STACK es igual a $00FF.
b)      LDX     #$1000          ; Se carga registro X con un valor de $1000.
        BSET    OPTION,X,$20    ; El bit IRQE del registro OPTION ($1039) se pone en 1.
c)      BSET    DDRD,X,$3F      ; Todas las terminales del puerto D configurados como salidas.
        BSET    PACTL,X,$08     ;PA3 se programa como salida.
d)      LDAA    #$3             ;PD5=PD4=PD3=0 (Queda seleccionado el puerto de salida P1)
        STAA    PORTD,X         ; PD2=PE=0, PD1=ERROR=1, PD0=STROBE=1.
e)      BSET    TFLG1,X,$79     ;Limpiar las banderas de estado (IC3F,OC5F,OC4F,OC3F,OC2F)
f)      BCLR    TMKS1,X,$FF     ;Configura acción Polled Operation.
g)      LDAA    #$BE            ;Configuración de bits de las funciones Output Compare.
        STAA    TCTL1,          ;OC2=PA6=0, OC3=PA5=1, OC4=PA4=1, OC5=PA3=0.
        BSET    CFORC,X,$78     ; Pone en 1 lógico los bits FOC2, FOC3, FOC4, FOC5.
h)      LDAA    #$2             ;Configuración de la función IC3 de PA0.
```

```
        STAA    TCTL2,X         ;La captura ocurre con el flanco de bajada.
i)      JSR     INIVAR          ; Pone en 0 las variables utilizadas en el programa.
j)      LDAA    #1              ;Se selecciona al puerto de salida P1 de la IEPP.
        STAA    PTO-PRED
k)      TPA
        ANDA    #$EF            ; Bit I=0, interrupciones mascarables (IRQ) habilitadas.
        TAP
```

Este grupo de instrucciones está dividido en secciones compuestas por una o más instrucciones con el objeto de explicar mejor el módulo de inicialización del sistema.

a) Se asignan para el Stack 256 bytes de la RAM interna del microcontrolador. Este espacio es usado por la CPU en el llamado de subrutinas, generación de interrupciones y almacenamiento temporal de datos.

b) El bit IRQE del registro OPTION permite seleccionar la forma en que se habilitarán las interrupciones generadas a la CPU a través de la terminal IRQ. Si IRQE vale cero queda habilitado el modo "level sensitive", lo que significa que se generará una interrupción a la CPU sólo si la señal presente en la terminal IRQ permanece en nivel bajo durante un mínimo de 24 ciclos de reloj para el caso más crítico (16 ciclos de reloj que dura el período de latencia de atención a interrupción mas 8 ciclos de reloj que dura la instrucción más larga del programa). Si IRQE vale uno queda habilitado el modo "edge sensitive", lo que significa que un flanco de bajada de la señal presente en la terminal IRQ generará una interrupción a la CPU del microcontrolador.
Debido a la necesidad de que la IEPP reciba íntegramente la información que transmite la PC, se optó por hacer uso del manejo de interrupciones a la CPU cada vez que la PC transmite un caracter. Para realizar esto se conecta la terminal 1 del puerto paralelo (señal de **STROBE**) a la terminal IRQ del microcontrolador. La señal de **STROBE** tiene una duración aproximada de 10 microsegundos, por la razón expuesta en el párrafo anterior resulta inadecuado dejar habilitado el modo "level sensitive".

Después de un reset el bit IRQE tiene un valor inicial de cero, y debido a que el registro OPTION se puede modificar sólo durante los primeros 64 ciclos de reloj el cambio de IRQE a uno lógico se realiza en las primeras instrucciones.

	7	6	5	4	3	2	1	0
OPTION $1039	ADPU	CSEL	IRQE	DLY	CME	O	CR1	CR0

c) Los puertos bidireccionales del MC68HC11E1 tienen asignado un determinado registro para determinar su dirección. Después del Reset estos puertos se inicializan como puertos de entrada. Las terminales bidireccionales que se utilizan son: PD0, PD1, PD2, PD3, PD4, PD5, PA3 y PA7. El registro asociado para direccionar el puerto D es el DDRD ($1009). En el caso del puerto A donde sólo se tienen estas dos terminales bidireccionales, el registro que contiene el DDRA3 y el DDRA7 es el PACTL ($1026). En el bit correspondiente se escribe un 1 para programar la terminal bidirecional como salida y un 0 para programarla como entrada. Las 6 terminales del puerto D se utilizan como salidas, por esta razón se escribe un $3F en el DDRD. En el bit DDRA3 también se escribe un 1 porque el puerto PA3 se utiliza como salida. La terminal PA7 se utiliza como entrada por lo que no es necesario cambiar el bit DDRA7.

d) La terminal PD0 genera la señal de **STROBE** para los dispositivos periféricos, PD1 y PD2 generan respectivamente las señales **ERROR** y **PE** para la PC. **STROBE** tiene un valor inicial de 1, **ERROR** un valor inicial de 1 y **PE** un valor inicial de 0. Las líneas PD3, PD4 y PD5 generan las señales **Slc1**, **Slc2** y **Slc3** respectivamente. Como se desea seleccionar al puerto de salida P1 de la IEPP como puerto activo, las terminales PD3, PD4 y PD5 tienen un valor inicial de 0.

e) El registro TFLG1 contiene las banderas de estado asociadas a las funciones Input Capture y Output Compare del puerto A. Cuando una función Output Compare o Input Capture ocurre, el microcontrolador lo indica poniendo en 1 el bit correspondiente en el registro TFLG1. Para limpiar cualquiera de estas banderas, se escribe un 1 en la posición del bit asociado. En condiciones iniciales las banderas asociadas a las funciones input

capture y output compare utilizadas, tienen que presentar un valor de 0, por esta razón se escribe un $79 en el registro TFLG1 ($1023).

	7	6	5	4	3	2	1	0
TFLG1 $1023	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F

f) Los bits del registro TMSK1 son para programar la acción que va a ejecutar la CPU cuando ocurra una captura o comparación, un 0 establece el modo "Polled Operation", un 1 habilita la acción de generación de interrupción a la CPU del microcontrolador. En este caso la acción que se programa es la "polled operation", para ello se escribe un 0 en todos los bits del registro TMSK1.

	7	6	5	4	3	2	1	0
TMSK1 $1022	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I
	Funciones Output Compare					Funciones Input Capture		

g) Para configurar la función output compare asociada a cada una de las terminales PA6, PA5, PA4 y PA3 se utiliza el registro TCTL1 ($1020). El valor de los bits OMx y OLx determina que nivel se presentará en la terminal correspondiente cuando ocurra un "output compare". Es decir, cuando el contenido de los registros TCNT y TOCx sea el mismo (TCNT=TOCx) o bien cuando de ejecute una comparación forzada mediante el registro CFORC.

	7	6	5	4	3	2	1	0
TCTL1 $1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
	PA6/OC2		PA5/OC3		PA4/OC4		PA3/OC5	

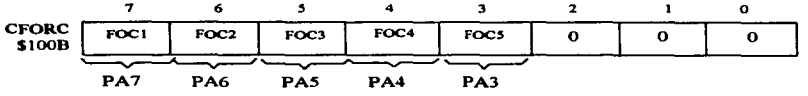
En la siguiente tabla se indican los valores que deberán tener los bits OMx y OLx en cada una de las cuatro configuraciones posibles. El registro TCTL1 contiene el patrón de bits para configurar la acción a seguir sobre los puertos asociados a cada una de las funciones Output Compare de acuerdo a la tabla 4.1.

OMx	OLx	Configuración
0	0	Se conserva nivel presente
0	1	Nivel contrario (toggle)
1	0	Nivel bajo (clear)
1	1	Nivel alto (set)

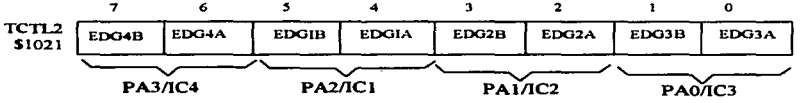
Tabla 4.1

El registro TCNT es un contador interno que se incrementa cada ciclo de reloj (si PR1 y PR0 del registro TMSK2 son iguales a 0) en forma cíclica desde $0000 hasta $FFFF. Si deseamos que una terminal OCx presente un determinado nivel después de un número N de ciclos de reloj, se procede entonces a almacenar la suma de TCNT+N en el registro TOCx correspondiente.

Ahora bien, si lo que se desea es forzar la terminal OCx a que presente el nivel indicado por la combinación de bits OMx y OLx del registro TCTL1, es necesario poner un 1 lógico en el bit FOCx del registro CFORC ($100B).



h) Para que el microcontrolador detecte la señal de ACKNLG proveniente del dispositivo periférico conectado al puerto activo, se conecta la salida Y del circuito multiplexor 74HC151 (U17) a la terminal PA0 la cual tiene asociada la función Input Capture 3 (IC3).



El valor de los bits EDGxB y EDGxA del registro TCTL2 determinan las condiciones de operación de las funciones "input capture" asociadas a las terminales del puerto A del microcontrolador, como se indica en la tabla 4.2.

EDGxB	EDGxA	Configuración
0	0	Función de captura deshabilitada
0	1	Captura en flanco de subida
1	0	Captura en flanco de bajada
1	1	Captura con cualquier flanco

Tabla 4.2

Escribiendo 00000010 en el registro TCTL2 quedan deshabilitadas las funciones de captura para el puerto PA3, PA2 y PA1, y habilitadas con flanco de bajada para el puerto PA0.

i) La subrutina INIVAR tiene la función de inicializar en cero el valor de todas las variables utilizadas en el programa.

j) Se selecciona el puerto de salida P1 de la IEPP como puerto activo o predeterminado.

k) Finalmente se cambia el valor del bit I del Condition Code Register (CCR) a cero para activar las interrupciones mascarables a la CPU del microcontrolador. A partir de este instante el sistema está listo para recibir información enviada por la PC vía el puerto paralelo bajo el protocolo de comunicación Centronics Parallel Interface.

4.1.2 Recepción de caracteres o datos (RECIBE-DATO).

La señal de **STROBE** que genera la PC para indicar el inicio de un ciclo de transmisión, provoca la generación de una interrupción de hardware al microcontrolador. La rutina de servicio a interrupción (RECIBE-DATO) se encarga del proceso de recepción de caracteres, de su almacenamiento en el buffer correspondiente y de la identificación de una palabra de control.

Como ya se mencionó en el capítulo anterior, para iniciar un ciclo de transmisión de un caracter bajo las normas del protocolo SPP las señales **PE**, **ERROR** y **BUSY** deberán tener respectivamente los niveles lógicos 010. Cuando se cumple esta condición, la PC pone el caracter en las líneas de datos del puerto paralelo y a continuación genera un pulso bajo en la señal de **STROBE**. El flanco de bajada de esta señal genera una interrupción a

la CPU del microcontrolador M68HC11E1, por esta razón el ciclo de recepción de un caracter es completamente asíncrono.

El pulso bajo de la señal de **STROBE** también activa el circuito U36, es decir, el estado presente en las entradas Dn se refleja en las salidas Qn. Lo que significa que con el flanco de bajada de la señal de **STROBE**, la salida Q0 cambia de nivel bajo a nivel alto. En este momento es Q0 quien mantiene en nivel alto a la señal de **BUSY**, ver figura 3.7.

En esta rutina de servicio de atención a interrupción (RECIBE-DATO), se lee el caracter presente en el puerto E y se almacena en la variable DATO, para posteriormente, determinar si éste forma parte de la información dirigida al puerto activo en ese momento o bien si éste forma parte de una palabra de control.

Los primeros 7 bytes de una palabra de control son: ^<*&%$#, siendo el byte número 8 el que determina la acción que se llevará a cabo. Si este byte es un ASCII comprendido entre $31-$38, que corresponden respectivamente a los números 1-8, significará que el usuario solicita activar el puerto de salida Pn ($1 \leq n \leq 8$). Si este byte es el ASCII comprendido entre $41-$48, que corresponden respectivamente a las letras A-H, significa que el usuario solicita que se transmita hacia la PC el estado de las señales **ERROR** y **PE** de los puertos de salida de la IEPP.

El DATO que se lee del puerto E del microcontrolador se almacena en el buffer de recepción de 8 bytes de longitud (BUFFER). A continuación se verifica si éste forma parte de la palabra de control, si es así permanece almacenado en BUFFER hasta que la IEPP haya recibido los 8 bytes y ejecutar así la acción que se indica. Si los caracteres contenidos en BUFFER no forman parte de la palabra de control, entonces se guardan en el buffer de 4 Kbytes asociado al puerto activo en ese momento.[4]

La variable APUNn ($1 \leq n \leq 8$) se utiliza para llevar una cuenta del número de bytes almacenada en BUFFERn. Si APUNn es igual a $1000, entonces BUFFERn está lleno y la PC no debe enviar mas caracteres a la IEPP hasta que la información almacenada en BUFFERn sea transmitida por el puerto de salida activo.

[4] Si en ese momento el puerto activo es P3 los datos se almacenan en la dirección BUFFER3+APUN3.

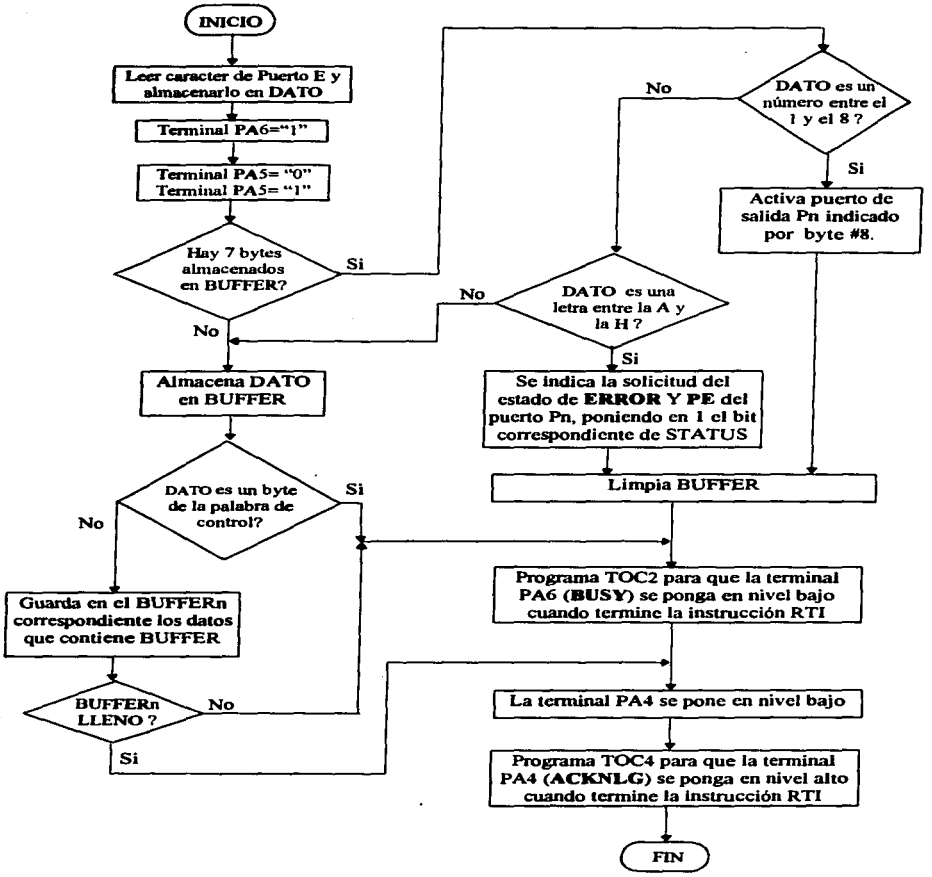


Figura 4.2.- Diagrama de flujo de la rutina de atención a interrupción (RECIBE-DATO).

Para indicar a la PC que la IEPP no puede recibir información por el momento, la señal de **BUSY** del puerto paralelo debe tener un nivel alto. Por lo tanto, cuando BUFFERn está lleno la rutina de atención a interrupción (RECIBE-DATO) termina con la terminal PA6 en nivel alto (ver figura 3.7).

En caso contrario, es decir cuando APUNn es menor a 4 Kbytes ($1000), la IEPP puede recibir el caracter y se programa la función OC2 asociada a la terminal PA6 para que la señal de **BUSY** se ponga en nivel bajo al término de la ejecución de la instrucción RTI.

Para indicarle a la PC que el caracter que transmitió fue recibido con éxito por la IEPP, el microcontrolador genera un pulso bajo en la señal de **ACKNLG** (terminal PA4). Para ello se programa también la función output compare 4 (OC4), forzando primero la terminal PA4 a un nivel bajo y programando después el registro TOC4 para que esta terminal cambie a nivel alto al término de la ejecución de la instrucción RTI.

4.1.3 Transmisión de caracteres hacia los dispositivos periféricos (TRANS-DATOS)

La información destinada a cada uno de los dispositivos conectados a los puertos de salida de la IEPP se encuentra almacenada en los buffer's de 4 Kbytes asociados (BUFFER1-BUFFER8). El proceso de transmisión se lleva a cabo como se indica en el diagrama de flujo de la Figura 4.3.

Dado que el proceso de transmisión llevado a cabo por la IEPP es similar para los 8 puertos de salida (P1-P8), será suficiente con explicar dicho proceso cuando se transmite la información hacia P1.

TRANS-DATOS inicia con la transmisión de la información contenida en BUFFER1 hacia el dispositivo conectado al puerto P1. Las terminales PD3, PD4 y PD5 del microcontrolador se ponen en 0, quedando así P1 seleccionado.

La variable APUN1 almacena el número de caracteres contenidos en BUFFER1 y la variable APUNM1 el número de caracteres transmitidos, cuando el valor de estas variables es igual significa que ya no hay más datos que enviar al puerto P1 y la transmisión de información por el puerto P1 termina.

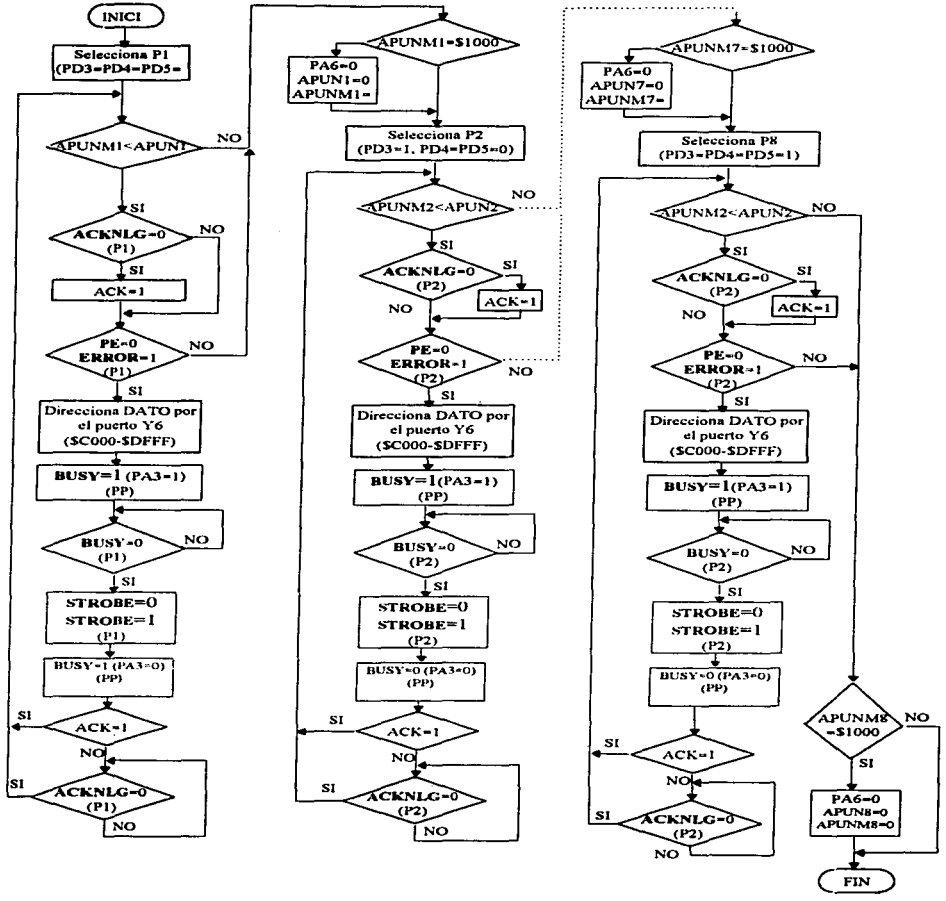


Figura 4.3.- Diagrama de flujo de la rutina TRANS-DATOS.

Para transmitir un caracter se verifica el estado de las señales PE y ERROR del puerto de salida P1. Si **PE** tiene un nivel alto, o **ERROR** un nivel bajo significa que el dispositivo periférico conectado en P1 esta en estado de error o sin papel y por lo tanto no puede recibir información en ese momento. En este caso el proceso de transmisión de caracteres hacia P1 se interrumpe hasta el siguiente ciclo de TRANS-DATOS, pero el proceso de transmisión continúa hacia el puerto P2 como se indica en la figura 4.3.

Ahora bien si el dispositivo periférico conectado en P1 tiene papel (**PE**=0) y no hay condición de error (**ERROR**=1), se procede a verificar que la señal de **BUSY** tenga un nivel bajo, es decir que el dispositivo periférico esté en condiciones de recibir información.

Una vez que se han verificado estas tres señales de estado del puerto P1, el caracter que se va a transmitir se coloca en las líneas de datos (D1-D7) de P1. Esto se realiza direccionando el dato en una localidad de memoria comprendida en el segmento $C000-$DFFF, con lo cual la terminal Y6 del circuito U8 se pone en nivel bajo (ver figura 3.7). El puerto Y6 activa el circuito 74HC273 (U9) que opera como puerto de salida de 8 bits externo. Dado que P1 es el que en este momento está activo, el caracter se refleja sólo en las líneas de datos de P1.

La IEPP le indica al dispositivo periférico conectado en P1 que el caracter está listo, generando un pulso bajo en la señal de **STROBE**. Si al inicio de la transmisión la señal de **ACKNLG** tiene un nivel alto, la IEPP espera una señal de reconocimiento (**ACKNLG**=0) para dar por terminado el ciclo de transmisión de un caracter.

El proceso de transmisión se lleva a cabo caracter por caracter y es similar para cada uno de los puertos de salida como se indica en el diagrama de flujo de la figura 4.3.

Antes de iniciar el proceso de transmisión hacia el siguiente puerto de salida de la IEPP, en este caso P2, se verifica si la variable APUNM1 es igual a $1000. En caso afirmativo, significa que lo 4 Kbytes de datos contenidos en BUFFER1 ya se transmitieron por P1 y por lo tanto se restablecen las condiciones iniciales para este puerto (APUN1=APUNM1=0) y además la terminal PA6 se pone en nivel bajo (**BUSY**=0) para indicarle a la PC que la IEPP puede recibir más datos.

4.1.4 Transmisión del estado de los puertos de salida de la IEPP hacia la PC (TRANS-ESTADO).

En la rutina encargada de realizar el ciclo de transmisión de un caracter se determina si se llevó a cabo la transmisión o si el dispositivo periférico no estuvo en condiciones de recibir el caracter. Si ocurrió esto último, el proceso de transmisión hacia ese puerto es abortado en ese momento y se reintenta cuando la transmisión le corresponda nuevamente a ese puerto de salida. Cuando el periférico sigue el protocolo de comunicación SPP el ciclo de transmisión de un caracter termina hasta que la IEPP recibe como señal de reconocimiento un pulso bajo en la señal de **ACKNLG**.

La IEPP reconoce que el usuario desea que se transmita hacia la PC el estado de las señales de **ERROR** y **PE** presentes en los diferentes puertos de salida (P1-P8) de la IEPP, cuando recibe una palabra de control cuyo byte número 8 es un ASCII comprendido entre $41-$48, que corresponden respectivamente a las letras A-H.

Para llevar a cabo esto se utilizan las terminales PD1 y PD2 (mismas que se utilizan para generar respectivamente las señales de **ERROR** y **PE** del puerto paralelo). La terminal PD1 se utiliza para generar una señal de aviso a la PC de que la respuesta está lista en la terminal PD2 de acuerdo a la tabla 4.3.

PD1	PD2	Estado de Pn
0	0	Listo
0	1	Error o sin Papel
1	*	Estado indicado por PD2 no válido

Tabla 4.3

Los 8 bits de la variable STATUS se utilizan como banderas para indicar cuando se ha recibido cada una de las 8 palabras de control de solicitud de estado. Para determinar el estado de Pn se procede a leer el estado de las señales de **ERROR** y **PE** presentes en dicho puerto. En la tabla 4.4 se indica el valor que se le asigna a PD1 y PD2 para los diferentes estados que pueden presentar estas señales.

ERROR	PE	PD1	PD2
0	0	0	1
0	1	0	1
1	0	0	0
1	1	0	1

Tabla 4.4

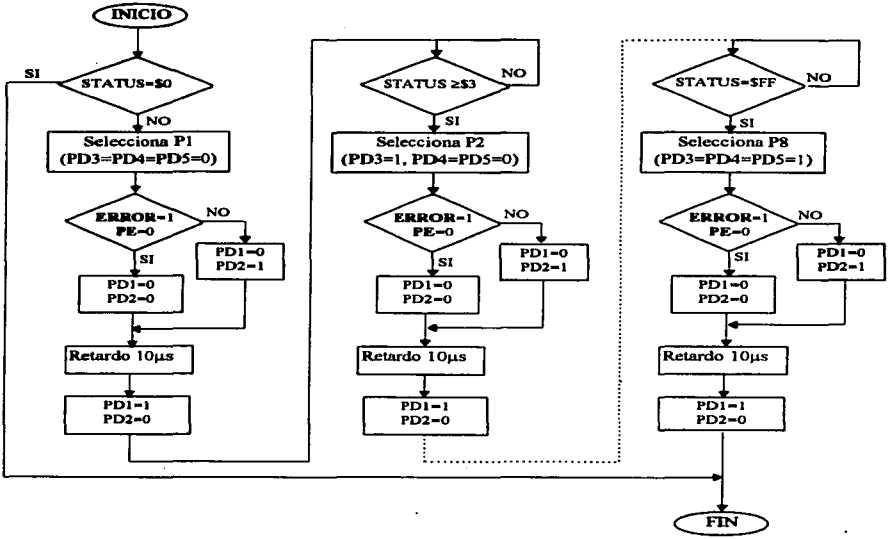


Figura 4.4.-Diagrama de flujo de la rutina TRANS-STATUS

En la rutina TRANS-ESTADO, cuyo diagrama de flujo se muestra en la figura 4.4, se lleva a cabo el proceso de transmisión del estado de cada uno de los puertos Pn hacia la PC.

Cuando se recibe la palabra de control ^<*&%$#A el bit menos significativo (bit 0) de la variable STATUS se pone en 1. Se procede entonces a leer el estado de las señales **ERROR** y **PE** del puerto P1 a continuación se genera el nivel correspondiente en PD1 y PD2 de acuerdo a la tabla 4.2. Cuando la PC ha leído el estado de estas señales procede a generar la siguiente palabra de control (^<*&%$#B) para solicitar el estado del puerto P2, y el proceso se repite nuevamente.

Como se muestra en la figura 4.4, una vez que la IEPP ha transmitido a la PC el estado del puerto P8, las terminales PD1 y PD2 que también generan las señales **ERROR** y **PE** quedan restablecidas con su valor inicial de 1 y 0 respectivamente.

Cabe mencionar que cuando el dispositivo periférico conectado al puerto Pn, no es un dispositivo de impresión o bien no hay ningún dispositivo conectado en dicho puerto, el estado de las señales que se registra en Pn es **ERROR**=1 y **PE**=0, debido a que las terminales asociadas a estas señales en cada uno de los puertos Pn están conectadas a una resistencia de pullup y pulldown respectivamente.

4.2 Programa ejecutado por la PC.

El programa MENU que es ejecutado por la PC permite seleccionar en forma automática cualquiera de los 8 puertos de salida de la IEPP y conocer si los dispositivos periféricos conectados a la interface están en estado de error o sin papel.

Este programa corre en ambiente DOS y en ambiente Windows. En el instructivo para el usuario del Apéndice B se indican los pasos necesarios para ejecutar un programa en general en ambiente Windows y evitar así lo engorroso que resulta hacer un shell a DOS cada vez que se desea ejecutarlo.

El programa está desarrollado en lenguaje ensamblador y la totalidad del código puede consultarse en el Apéndice A. Este programa despliega un menú, para que el usuario

elija cualquiera de las dos opciones: seleccionar uno de los ocho puertos de salida de la IEPP o desplegar el estado en el que operan los dispositivos periféricos conectados.

Cuando se elige la primera opción, se despliega en pantalla una lista en la que se indica que tecla hay que presionar para seleccionar un puerto Pn determinado. Si la tecla presionada es válida, el programa genera la palabra de control ^<*&%$#n donde n representa el número de puerto seleccionado es decir $1 \leq n \leq 8$.

Al elegir la segunda opción, el programa genera la primera palabra de control para solicitud de estado (^<*&%$#A). Cuando la IEPP recibe estos 8 bytes, transmite hacia la PC el estado correspondiente del puerto P1 como se describió en la sección 4.1.4 de este capítulo. A continuación el programa menú genera la segunda palabra de control (^<*&%$#B) y el proceso se repite hasta que la PC recibe el estado del puerto P8. Finalmente se despliega en pantalla el estado en el que opera cada uno de los puertos de salida de la IEPP.

Cabe mencionar que el usuario puede transmitir directamente el código ASCII de cada uno de los caracteres que forman la cadena de control que desea transmitir, la selección se llevará a cabo de igual forma. Para realizar esto, sólo hay que tener la seguridad de que el código ASCII de los 8 bytes que forman la cadena de control deben llegar en forma íntegra a la IEPP, de lo contrario estos 8 bytes son interpretados como información que deberá ser transmitida por el puerto Pn activo en ese momento.

CAPÍTULO 5

PRUEBAS EXPERIMENTALES

Como ya se explicó en capítulos anteriores, el protocolo de comunicación utilizado por la IEPP es el Centronics Parallel Interface y por lo tanto se pueden conectar en los puertos de salida del sistema cualquier dispositivo de impresión que reciba datos bajo este protocolo, y también bajo ciertas restricciones, computadoras personales y periféricos que normalmente se conectan al puerto paralelo de una PC a través de las 8 líneas de datos y tierra, por ejemplo, sistemas integrados con motores de pasos.
Para describir las pruebas experimentales realizadas a lo largo del desarrollo de la IEPP, se mencionarán primero las que se refieren a la transmisión de información hacia un dispositivo de impresión.

5.1 Transmisión de datos de la PC maestra hacia un dispositivo de impresión .

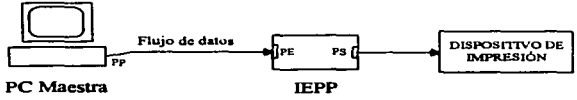


Figura 5.1

El primer prototipo del sistema fue un esquema básico formado por el puerto de entrada y un sólo puerto de salida. El puerto de entrada se conecta al puerto paralelo de la PC y al puerto de salida se conecta un dispositivo de impresión, como se muestra en la figura 5.1.

En términos generales, el prototipo operaba como se describe a continuación. La señal de **STROBE** del puerto paralelo estaba conectada a la terminal XIRQ del microcontrolador, con el propósito de generar una interrupción no mascarable a la CPU cuando la PC generara un pulso bajo en esta señal. La señal de **BUSY** se generaba con la terminal PA5 del microcontrolador y la señal de **ACKNLG** con la terminal PA6. Al inicio de la rutina de servicio a interrupción se ponía en nivel alto la señal de **BUSY**, se leía el caracter por el puerto E del microcontrolador para almacenarlo en el buffer de recepción, finalmente se generaba el pulso bajo de la señal de **ACKNLG** y la señal de **BUSY** se ponía en nivel bajo. Como se puede apreciar en este primer prototipo se intentó generar los cambios de estado de las señales **ACKNLG** y **BUSY** con instrucciones de software directamente.

Las pruebas de impresión realizadas con este prototipo revelaron lo siguiente:

a) Cuando el documento transmitido desde la PC maestra era desde el editor del DOS, Orcad, Word Perfect y Print del DOS se reproducía íntegramente por el dispositivo de impresión.
b) Cuando el documento se transmitía de AutoCad en ocasiones la impresión era correcta y en otras la reproducción era incompleta.
c) Desde Windows no fue posible imprimir íntegramente ningún documento.

La localización del problema fue difícil porque éste se presentaba en forma aleatoria, sin embargo se logró solucionar el problema una vez que la sincronía de las señales entre la PC y la IEPP fue la adecuada.

Al ir depurando cada una de las posibles causas, se obtuvo un prototipo con las siguientes modificaciones:

a) Las interrupciones no mascarables se activan por nivel únicamente, es decir, la terminal XIRQ debe permanecer en nivel bajo durante 28 ciclos de reloj, en el caso más crítico, para que se asegure la generación de una interrupción a la CPU. La señal de **STROBE** generada por la PC es aproximadamente de 10 microsegundos, lo que implicaba que no todos los requerimientos de atención a interrupción fueran atendidos por la CPU y por lo tanto se perdieran caracteres. La solución a este

problema fue utilizar las interrupciones mascarables IRQ. Este tipo de interrupciones, como ya se mencionó en el capítulo anterior, son programables para ser activadas por nivel o por flanco de bajada. Al quedar programado el modo "edge sensitive", el ancho del pulso de la señal de **STROBE** no es determinante ya que es con el flanco de bajada con el que se genera la interrupción a la CPU.

b) En el diagrama de tiempos de la figura 2.2 se observa que la señal de **BUSY** cambia a nivel alto cuando la señal de **STROBE** baja. Sin embargo, este prototipo generaba el cambio de nivel de la señal de **BUSY** con un retardo de 18 microsegundos. Esto llegó a provocar desincronía entre la PC y el prototipo, ya que este retardo era interpretado por la PC como disponibilidad por parte del prototipo para recibir el siguiente caracter. La solución factible a este problema fue agregar circuito U36 (74HC273 flip-flop tipo D con latch) como se indica en el diagrama de la figura 3.7.
La señal de $\overline{\text{STROBE}}$ se conecta a la terminal CLK, las entradas D0-D7 se conectan a VCC y CLR a la terminal de salida PA5 del microcontrolador. Dado que la terminal PA5 tiene un valor inicial de 1 lógico, de acuerdo a la tabla de estados del 74HC273, se tiene que el flanco de bajada de la señal de **STROBE** provoca un estado lógico 1 en las terminales de salida Qn.

CLR	CLK	Dn	Qn
L	X	X	L
H	↑	L	L
H	↑	H	H

Tabla 5.1.- Tabla de estados del 74HC273

En el diagrama de la figura 3.7 se puede observar que la señal de **BUSY** está formada por Q0+PA6+PA3, por lo tanto cuando Q0=1 también **BUSY**=1. En la rutina de servicio de atención a interrupción se pone la terminal PA6 en nivel alto y a continuación se genera un pulso bajo en la terminal PA5 para inicializar el 74HC273 (Q0=0).

c) Como ya se mencionó, al final de la rutina de servicio de atención a interrupción se generaba el pulso bajo en la señal de **ACKNLG** y se ponía en nivel bajo la terminal PA6 (**BUSY**). Tomando en cuenta que la PC considera que puede transmitir el

siguiente caracter inmediatamente después de que la señal de **BUSY** baja, existía la posibilidad de que generara el pulso bajo de la señal de **STROBE** cuando la instrucción **RTI** aún no se ejecutara.

Aunque esto no representaba ningún problema en cuanto a la recepción de caracteres por parte del prototipo, dado que las interrupciones generadas por la terminal IRQ del microcontrolador quedan anidadas, es preferible que la señal de **BUSY** cambie de nivel hasta que la instrucción **RTI** se ejecute completamente. Por esta razón se utilizaron las funciones "output compare" asociadas a los puertos PA6 y PA4 para generar los cambios de nivel de las señales **BUSY** y **ACKNLG** al final del ciclo de recepción de un caracter como lo indica el diagrama de tiempos de la figura 2.2.

Con los cambios indicados, se logró que la información transmitida por la PC se recibiera íntegramente caracter por caracter, y a su vez ésta fuera retransmitida por el prototipo hacia el dispositivo de impresión conectado en el puerto de salida.

Las pruebas finales realizadas con la IEPP, cuyas características se describieron en el capítulo 2, indicaron que es posible conectar los 8 dispositivos de impresión en forma simultánea y direccionar la información en forma automática desde la computadora personal hacia cualquiera de los 8 dispositivos conectados.

5.2 Transmisión de datos desde la PC maestra hacia una PC esclava.

La primer consideración importante es que la computadora esclava deberá tener un puerto paralelo cuyas lineas de datos sean bidireccionales. Las terminales que corresponden a las señales de control y de estado son unidireccionales en todas las computadoras personales. Esto hace necesario conectar la computadora esclava de manera especial a la IEPP. Para ello se utiliza un cable de 25 hilos con un conector tipo DB25 macho en cada extremo, en los cuales deberá estar debidamente indicado a cual de los dos extremos corresponde (puerto paralelo de la PC esclava o puerto de salida de la IEPP) para evitar daños eléctricos a los circuitos de salida de ambos dispositivos. En la tabla 5.1 se indica que señal deberá conectarse en cada una de las terminales de ambos conectores.

PUERTO DE SALIDA DE LA IEPP	PUERTO PARALELO DE LA PC ESCLAVA
Terminal #1 (Salida)	Terminal #10 (Entrada)
Terminal #2 (Salida)	Terminal #2 (Entrada)
Terminal #3 (Salida)	Terminal #3 (Entrada)
Terminal #4 (Salida)	Terminal #4 (Entrada)
Terminal #5 (Salida)	Terminal #5 (Entrada)
Terminal #6 (Salida)	Terminal #6 (Entrada)
Terminal #7 (Salida)	Terminal #7 (Entrada)
Terminal #8 (Salida)	Terminal #8 (Entrada)
Terminal #9 (Salida)	Terminal #9 (Entrada)
Terminal #10 (Entrada)	Terminal #1 (Salida)
Terminal #11 (Entrada)	Terminal #14 (Salida)
Terminales: 12,13,14,15,16 y17 Sin conectar	Terminales: 11,12,13,15,16 y17 Sin conectar
Terminal #18-25 (Tierra)	Terminal #18-25 (Tierra)

Tabla 5.1.- Forma de unir las terminales de los conectores del cable utilizado para conectar una PC esclava a la IEPP.

A continuación se explica como elaborar un programa en lenguaje ensamblador ejecutado por la PC esclava para establecer comunicación con la IEPP a través del LPT1.

Los puertos asociados al LPT1 son el 378H, 379H y 37AH. El puerto 378H es el byte de datos D0-D7 (terminales 2-9 del puerto paralelo), en la mayoría de las computadoras IBM-386 y posteriores este puerto es bidireccional. El 379H es un puerto de entrada exclusivamente por lo que es utilizado para sensar las señales de estado localizadas en las terminales 10, 11, 12, 13 y 15 del puerto paralelo. El 37AH es un puerto de salida y por lo tanto es utilizado para generar las señales de control localizadas en las terminales 1, 14, 16 y 17 del puerto paralelo.

La estructura del programa que deberá ejecutarse en la PC esclava para leer datos a través del puerto paralelo es el que se muestra en el diagrama de flujo de la Figura 5.1.

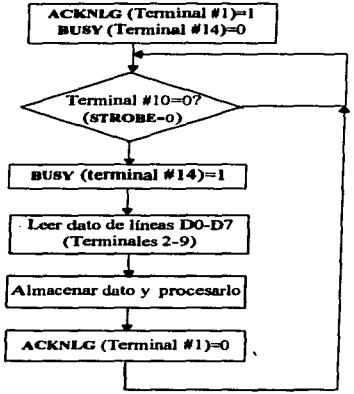


Figura 5.1.- Estructura básica de un programa ejecutado en una PC esclava para recibir datos a través del puerto paralelo.

Para manipular correctamente estas señales es necesario conocer primero cuales son los bits de los puertos correspondientes que están asociados a cada una de ellas.

	Bit 7	6	5	4	3	2	1	Bit 0
378H	Term #9 (D7)	Term #8 (D6)	Term #7 (D5)	Term #6 (D4)	Term #5 (D3)	Term #4 (D2)	Term #3 (D1)	Term #2 (D0)

	Bit 7	6	5	4	3	2	1	Bit 0
379H	Term #11 (n)	Term #10 (nn)	Term #12 (nn)	Term #13 (nn)	Term #15 (nn)	*	*	*

	Bit 7	6	5	4	3	2	1	Bit 0
37AH	*	*	*	*	Term #17 (n)	Term #16 (nn)	Term #14 (n)	Term #1 (n)

NOTA:
n=negado
nn=no negado

Dado que la rutina que se ejecuta en la IEPP para transmitir un caracter, es la misma sea cual fuere el dispositivo que se encuentre conectado, a continuación se explica de que manera se lleva a cabo el proceso de transmisión cuando el dispositivo periférico es una computadora personal.

a) Se verifica el estado de la señal **BUSY** (terminal #11) del puerto de salida de la IEPP, esta terminal está unida físicamente a la terminal #14 del puerto paralelo de la PC esclava, es por ello que en el diagrama de flujo se indica que habrá que inicializar esta señal en un nivel bajo para que la PC esclava pueda recibir datos.

b) Una vez que la PC esclava detecta un nivel bajo en la terminal #10 (pulso de **STROBE** generado por la IEPP), pone en alto la señal de **BUSY** (terminal #14) durante la lectura, almacenamiento y procesamiento del dato recibido.

c) Al final de la rutina se genera el pulso bajo por la terminal #1 (señal de **ACKNLG** para la IEPP) para indicar que el caracter fue recibido con éxito.

Las pruebas realizadas para transmitir información desde la PC maestra hacia una PC esclava conectada a uno de los puertos de salida de la IEPP, consistieron en enviar una determinada cadena de caracteres en código ASCII. Cada uno de los bytes recibidos por la PC esclava se desplegaron en la pantalla y se pudo verificar la integridad de la información transmitida. El listado completo del programa desarrollado en lenguaje ensamblador para la elaboración de esta prueba se encuentra en el Apéndice A.

5.3 Transmisión de información hacia dispositivos que sólo reciben el byte de datos.

Es común encontrar sistemas con motores de pasos que son controlados directamente por una computadora personal. Normalmente están conectados al puerto paralelo de la PC, únicamente a través de las líneas de datos D0-D7 (terminales 2-9) y tierra.

Para manipular por lo tanto estos sistemas, se transmite el dato correspondiente directamente por el puerto 378H. Pero en el caso en que un sistema de este tipo se conecte a uno de los puertos de la IEPP, la información enviada desde la PC maestra

deberá transmitirse siguiendo el protocolo de comunicación estándar Centronics Parallel Interface, ya que la IEPP recibe información siguiendo las normas de este protocolo. La Int 17h con servicio 0 se puede utilizar en este caso para transmitir los datos desde la PC maestra hacia la IEPP.

Las líneas correspondientes a las señales de **PE** y **ERROR** quedan sin conectar y por lo tanto el microcontrolador registra **PE**=0 y **ERROR**=1 en el puerto de salida de la IEPP donde esté conectado este dispositivo. La rutina ejecutada en la IEPP para transmitir un caracter hacia cualquiera de los dispositivos conectados a los puertos de salida es la misma descrita en párrafos anteriores.

Para realizar pruebas con este tipo de dispositivos se utilizó un brazo mecánico de Electrónica Veneta de 6 grados de libertad, conectado en uno de los puertos de salida de la IEPP. El código del programa que se ejecuta en la PC para mover el brazo mecánico se lista en el Apéndice A. Una vez más se comprobó que el sistema transmite íntegramente los datos recibidos por la PC maestra ya que el brazo mecánico sigue la rutina de movimiento programada.

5.4 Más de dos dispositivos periféricos conectados en forma simultánea.

Para realizar este tipo de pruebas se dispuso de un brazo mecánico de Electrónica Veneta, un plotter a colores marca Graphtec MP4300, una impresora láser HP LaserJet IIP Plus, una impresora de matriz de puntos Citizen HSP-550 y una computadora PC marca DELL 316SX. Estos dispositivos se conectaron en forma aleatoria a 5 de los puertos de salida de la IEPP.

Para seleccionar en forma automática cualquiera de los puertos de la IEPP se utilizó el programa instalado en la PC, el cual puede ser ejecutado desde Windows sin necesidad de hacer un Shell a Dos. En todos los casos la transmisión de información se llevó a cabo con éxito.

CAPÍTULO 6

RESULTADOS Y CONCLUSIONES.

El objetivo planteado originalmente para el diseño de este sistema fue diseñar un dispositivo electrónico digital basado en el microcontrolador M68HC11 de Motorola, que permitiera la interconexión automática de 8 dispositivos terminales (impresoras, plotters, PC's, etc., o combinación de estos) con una PC, utilizando el puerto paralelo.

La IEPP satisface plenamente dicho objetivo ya que en las pruebas realizadas se probaron los 8 puertos de salida, conectando a cada uno de ellos dispositivos que reciben datos a través del puerto paralelo. La conmutación entre los diferentes puertos de salida se lleva a cabo en forma automática, es decir, el usuario indica desde la PC maestra hacia cual de los puertos de salida de la IEPP va dirigida la información que se transmita a través del puerto paralelo.

Se diseñó un software para ser ejecutado en la PC maestra que le permite al usuario, a través de un menú, elegir el puerto de salida de la IEPP y conocer el estado en el que opera cada uno de los 8 puertos. La palabra de control (secuencia específica de 8 caracteres) también puede transmitirse directamente desde cualquier editor que transmita íntegramente el código ASCII correspondiente a cada caracter de la palabra de control, por ejemplo el Editor del DOS.

Esta interface es completamente portable en el sentido de que puede conectarse directamente en el puerto paralelo de cualquier computadora personal.

De las pruebas realizadas con diferentes tipos de dispositivos periféricos se puede concluir que la IEPP recibe íntegramente la información transmitida desde la PC y que a su vez la transmite correctamente por los respectivos puertos de salida, ya que los dispositivos periféricos conectados a la IEPP reproducen la información fielmente.

El software empleado para transmitir la información desde la PC maestra hacia la IEPP puede ser cualquiera (Orcad, Chi-Writer, Word Perfect, Autocad, Word para Windows, Editor del DOS, etc.) siempre y cuando éstos utilicen el protocolo de comunicación estándar para puerto paralelo Centronics Parallel Interface.

Para mejorar el diseño se pueden hacer algunos cambios en el hardware de la IEPP, estos son los siguientes:

1) Para el desarrollo del prototipo se utilizaron circuitos buffer como medida de protección a posibles sobrecargas en los demás elementos del circuito. Lo cual no resulta estrictamente necesario dado que se están empleando circuitos con tecnología CMOS los cuales tienen en promedio un fanout de 8.

2) El módulo de transmisión de señales de control hacia los dispositivos periféricos se diseñó utilizando compuertas OR para realizar el demultiplexaje de las señales **STROBE**; **SLCT IN**, **INIT** y **AUTO FEED XT** como está indicado en el diagrama de la figura 3.9. Sin embargo resulta más óptimo utilizar circuitos 74HC138 como demultiplexores, dado que en vez de utilizar los 8 circuitos 74HC32 se utilizarían sólo 4 circuitos 74HC138. El diagrama de conexiones para el módulo de transmisión de señales de control hacia los dispositivos periféricos utilizando circuitos 74HC138 se muestra en la figura 6.1.

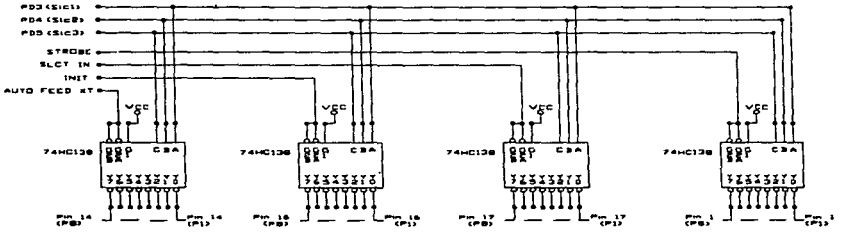


Figura 6.1.- Demultiplexaje de las señales de estado para los puertos de salida de la IEPP.

3) Puede agregarse memoria RAM a la IEPP para tener buffer's más grandes, por ejemplo de 32 Kbytes en vez de los 4 Kbytes disponibles en este prototipo. Esto implica disponer de 3 terminales de salida del microcontrolador para direccionar las 8 RAM's adecuadamente, estas tres terminales pueden ser PD5, PD4 y PD3 y generar las señales SLC1, SLC2 y SLC3 a través de un puerto de salida externo. El diagrama de conexiones para direccionar 8 circuitos RAM de 32 Kbytes se presenta en la figura 6.2.

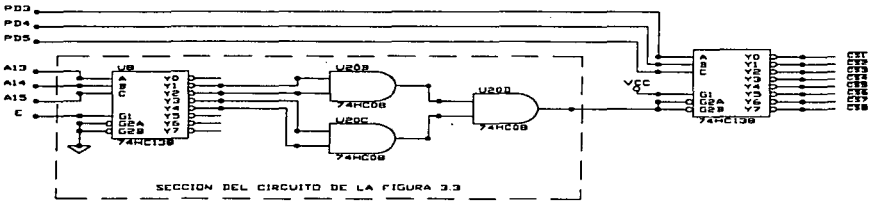


Figura 6.2..- Generación de las líneas de habilitación para direccionar 8 circuitos RAM de 32 Kbytes.

Para direccionar circuitos RAM de capacidad superior a 32 Kbytes es necesario replantear el diseño utilizando otro microcontrolador, por ejemplo el M68HC11F1, el cual tiene una mayor cantidad de puertos de entrada/salida de propósito general al operar en modo bootstrap, y esto permite direccionar directamente memorias de más de 64 Kbytes.

El prototipo de la IEPP tiene un costo material aproximado de $800.00, el cual puede reducirse al implementar las mejoras al circuito antes mencionadas.

APÉNDICE A
LISTADOS DE LOS PROGRAMAS

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                                                                        %%%%
%%%%                              IEPP.ASM                                  %%%%
%%%%      PROGRAMA EJECUTADO POR EL MICROCONTROLADOR M68HC11                %%%%
%%%%      DE LA  INTERFACE ELECTRÓNICA PARA PUERTO PARALELO                 %%%%
%%%%                                                                        %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

STACK       EQU  $00FF   ;256 BYTES PARA LA PILA
MEM         EQU  $C000   ;PUERTO DE SALIDA DE DATOS
PORTA       EQU  $00     ;PUERTO A ($1000)
PORTD       EQU  $08     ;PUERTO D ($1008)
PORTE       EQU  $0A     ;PUERTO E ($100A)
DDRD        EQU  $09     ;DIRECCIONAR PUERTO D
PACTL       EQU  $26     ;DIRECCIONAR PA3 Y PA7
CFORC       EQU  $0B     ;REGISTRO DE LA FUNCIÓN OUTPUT COMPARE
TCNT        EQU  $0E     ;REGISTRO CONTADOR INTERNO DE 16 BITS
TOC2        EQU  $18     ;REGISTRO DE LA FUNCIÓN OUTPUT COMPARE
TOC3        EQU  $1A     ;REGISTRO DE LA FUNCIÓN OUTPUT COMPARE
TOC4        EQU  $1C     ;REGISTRO DE LA FUNCIÓN OUTPUT COMPARE
TCTL1       EQU  $20     ;PARA CONFIGURAR LA FUNC OUTPUT COMPARE
TCTL2       EQU  $21     ;PARA CONFIGURAR  LA FUNCIÓN INPUT CAPTURE
TMSK1       EQU  $22     ;HABILITA INTERRUPCIONES EN FUNCS IC Y OC
TFLG1       EQU  $23     ;REGISTRO DE BANDERAS DE FUNCIONES IC Y OC
OPTION      EQU  $39     ;RECONFIGURACIÓN DEL BIT IRQE

            ORG  $0100   ;DECLARACIÓN DE VARIABLES (RAM INTERNA)
APUNn       RMB  $2      ;APUNTA AL ULTIMO CARÁCTER RECIBIDO
APUN1       RMB  $2      ;ULTIMO CARÁCTER ALMACENADO EN BUFFER1
APUN2       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER2
APUN3       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER3
APUN4       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER4
APUN5       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER5
APUN6       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER6
APUN7       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER7
APUN8       RMB  2       ;ULTIMO CARÁCTER ALMACENADO EN BUFFER8
APUNM1      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P1
APUNM2      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P2
APUNM3      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P3
APUNM4      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P4
APUNM5      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P5
APUNM6      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P6
APUNM7      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P7
APUNM8      RMB  2       ;ULTIMO CARÁCTER TRANSMITIDO POR P8
BUFFER      RMB  8       ;BUFFER DE RECEPCIÓN
BUFFERn     RMB  2       ;VARIABLE DE USO TEMPORAL
DATO        RMB  1       ;ALMACENA EL DATO A RECIBIDO
DATO-OUT    RMB  1       ;ALMACENA EL DATO A TRANSMITIR
STATUS      RMB  1       ;REGISTRA LLEGADA DE CADENAS DE CONTROL
                         ;DE SOLICITUD DE ESTATUS DE LOS PUERTOS Pn
PTO-PRED    RMB  1       ;CONTIENE EL NUM DE PUERTO PREDETERMINADO
BYCAD       RMB  1       ;# BYTES DE ALMACENADOS EN BUFFER
```

```
CONT      RMB  1          ;# BYTES DE BYTES EXTRAÍDOS DE BUFFER
ACK       RMB  1          ;INDICA SI EL PERIF CONEC A Pn ES DE IMPRESIÓN
TEMP      RMB  1          ;VARIABLE DE USO TEMPORAL
TRANS-OK  RMB  1          ;INDICA SI LA TRANSMISIÓN POR Pn TUBO ÉXITO
LLENO     RMB  1          ;INDICA CUANDO BUFFERn ESTA SATURADO

          ORG  $2000      ;DISTRIBUCIÓN DE RAM EXTERNA
BUFFER1   RMB  $1000      ;2000-2FFF -> ALMACENA INFO. DESTINADA A P1
BUFFER2   RMB  $1000      ;3000-3FFF -> ALMACENA INFO. DESTINADA A P2
BUFFER3   RMB  $1000      ;4000-4FFF -> ALMACENA INFO. DESTINADA A P3
BUFFER5   RMB  $1000      ;5000-5FFF -> ALMACENA INFO. DESTINADA A P4
BUFFER4   RMB  $1000      ;6000-6FFF -> ALMACENA INFO. DESTINADA A P5
BUFFER6   RMB  $1000      ;7000-7FFF -> ALMACENA INFO. DESTINADA A P6
BUFFER7   RMB  $1000      ;8000-8FFF -> ALMACENA INFO. DESTINADA A P7
BUFFER8   RMB  $1000      ;9000-9FFF -> ALMACENA INFO. DESTINADA A P8

          ORG  $E000              ;COMIENZA CÓDIGO DE PROGRAMA
CADENA    FCB  $5E,$3C,$2A        ;CADENA CONTROL: ^<*&%$#
          FCB  $26,$25,$24,$23,$40


INICIALIZACION:
          LDS  #STACK
          LDX  #$1000             ;MODO DE DIRECCIONAMIENTO INDEXADO
          BSET OPTION,X,$20       ;IRQE CONFIGURADO POR FLANCO
          LDAA #$3F
          STAA DDRD,X             ;PD0-PD5 COMO PUERTOS DE SALIDA
          BSET PACTL,X,$08        ;PA3 COMO PUERTO DE SALIDA
          LDAA #$3                ;PD5=PD4=PD3=0 (P1 SELECCIONADO)
          STAA PORTD,X            ;SEÑALES HACIA PC: PD2=PE=0 Y PD1=ERROR=1
          BSET TFLG1,X,$79        ;LIMPIA BANDERAS:IC3,OC5,OC4,OC3,OC2
          LDAA #0                 ;ACCIÓN "POLLED OPERATION"
          STAA TMSK1,X            ;PARA FUNCIÓN INPUT CAPTURE Y OUTPUT
COM
          LDAA #$BE               ;CONFIGURACIÓN DE OUTPUT COMPARE
          STAA TCTL1,X            ;PA6/OC2=0 PA5/OC3=1 PA4/OC4=1 PA3/OC5=0
          BSET CFORC,X,$78        ;FORZA LA ACCIÓN INDICADA EN TCTL1
          LDAA #$2                ;CONFIGURACIÓN INPUT CAPTURE DE PA0
          STAA TCTL2,X            ;PARA DETECTAR FLANCO DE BAJADA DE
                                  ;LA SEÑAL ACKNLG DE LOS PERIFÉRICOS
          JSR  INIVAR             ;INICIALIZA EN CERO A TODAS LAS VARIABLES
          LDAA #1
          STAA PTO-PRED           ;PUERTO P1 COMO PREDETERMINADO
          TPA
          ANDA #$EF               ;SE HABILITAN INTERRUPCIONES (I=0)
          TAP


START:
          JSR  TRANS-DATOS        ;TRANSMITE DATOS A DISPS. PERIFÉRICOS
          JSR  ESTATUS            ;TRANS ESTATUS DE DISP PERIF HACIA PC
```

```
            BRA         START
;-------------------------------------------------------------------------------
; TRASMITE LOS DATOS QUE SE ENCUENTRAN ALMACENADOS EN LOS BUFFERS DE 4
; KBYTES ASOCIADOS A CADA UNO DE LOS PUERTOS DE SALIDA HACIA LOS
; DISPOSITIVOS PERIFÉRICOS CONECTADOS EN DICHOS PUERTOS.

TRANS-DATOS:
            JSR         TRANS-P1
            JSR         TRANS-P2
            JSR         TRANS-P3
            JSR         TRANS-P4
            JSR         TRANS-P5
            JSR         TRANS-P6
            JSR         TRANS-P7
            JSR         TRANS-P8
            RTS

;-------------------------------------------------------------------------------
;SI STATUS ≠ 0, LA PC SOLICITO EL ESTADO DE LOS PUERTOS DE SALIDA Pn DE LA IEPP.
;SE UTILIZAN LAS SEÑALES PE (PD2) Y ERROR (PD1) PARA TRANSMITIR HACIA LA PC EL
;ESTADO DE CADA UNO DE LOS PUERTOS Pn DE LA IEPP.

ESTATUS:
            LDX         #$1000
            LDY         #STATUS
            LDAA        #0
            CMPA        STATUS
            BEQ         FIN-EST             ;NO HA LLEGADO CADENA ^<*&%$#A
            BCLR        PORTD,X,$38         ;HABILITA P1
            JSR         T-EST-Pn
W-EST2:
            BRCLR       00,Y,$2,W-EST2      ;ESPERA CADENA DE ESTATUS ^<*&%$#B
            BSET        PORTD,X,$08         ;HABILITA P2
            JSR         T-EST-Pn
W-EST3:
            BRCLR       00,Y,$4,W-EST3      ;ESPERA CADENA DE ESTATUS ^<*&%$#C
            BSET        PORTD,X,$10         ;HABILITA P3
            BCLR        PORTD,X,$08
            JSR         T-EST-Pn
W-EST4:
            BRCLR       00,Y,$8,W-EST4      ;ESPERA CADENA DE ESTATUS ^<*&%$#D
            BSET        PORTD,X,$08         ;HABILITA P4
            JSR         T-EST-Pn
W-EST5:
            BRCLR       00,Y,$10,W-EST5     ;ESPERA CADENA DE ESTATUS ^<*&%$#E
            BCLR        PORTD,X,$18         ;HABILITA P5
            BSET        PORTD,X,$20
            JSR         T-EST-Pn
W-EST6:
            BRCLR       00,Y,$20,W-EST6     ;ESPERA CADENA DE ESTATUS ^<*&%$#F
            BSET        PORTD,X,$08         ;HABILITA P6
```

```
            JSR         T-EST-Pn
W-EST7:
            BRCLR       00,Y,$40,W-EST7     ;ESPERA CADENA DE ESTATUS ^<*&%$#G
            BCLR        PORTD,X,$08         ;HABILITA P7
            BSET        PORTD,X,$10
            JSR         T-EST-Pn
W-EST8:
            BRCLR       00,Y,$80,W-EST8     ;ESPERA CADENA DE ESTATUS ^<*&%$#H
            BSET        PORTD,X,$08         ;HABILITA P8
            JSR         T-EST-Pn
            CLR         STATUS
FIN-EST:
            RTS

;-------------------------------------------------------------------------------
;TRANSMITE EL ESTATUS DEL PUERTO Pn  HACIA LA PC

T-EST-Pn:
            BRSET       PORTA,X,2,NO-OK     ;PE=1 =>PUERTO NO O.K.
            BRCLR       PORTA,X,4,NO-OK     ;ERROR=0 =>PUERTO NO O.K.
            BCLR        PORTD,X,6           ;PUERTO O.K. =>ERROR=0, PE=0
            BRA         F-EST-Pn
NO-OK:
            BCLR        PORTD,X,2           ;ERROR=0
            BSET        PORTD,X,4           ;PE=1
F-EST-Pn:
            JSR         RETARDO
            BSET        PORTD,X,2           ;RESTABLECE  CONDICIONES INICIALES
            BCLR        PORTD,X,4           ;ERROR=1 Y PE=0
            RTS

RETARDO:                                    ;RETARDO DE 65 MILISEGUNDOS
            PSHX                            ;PARA SINCRONIZARSE CON LA PC
            LDX         #$3FFF
RET1:       DEX
            CPX         #0
            BNE         RET1
            PULX
            RTS

;-------------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER1 POR EL PUERTO P1

TRANS-P1:
            LDX         #$1000
            BCLR        PORTD,X,$38         ;HABILITA EL PUERTO P1
            LDX         APUNM1
MAS1:
            LDD         #BUFFER1
            CPX         APUN1               ;HAY DATOS EN BUFFER1 PENDIENTES
            BEQ         FIN-MAN1            ;POR TRANSMITIR?
```

```
            ADDD    APUNM1
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA
            LDAA    #1
            CMPA    TRANS-OK            ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN1            ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM1              ;ACTUALIZA APUNM1
            BRA     MAS1
FIN-MAN1:
            CPX     #$1000              ;BUFFER1 ESTA LLENO?
            BNE     FIN1                ;NO => TERMINA TRANSMISIÓN POR P1
            JSR     LIMPIA              ;RESTABLECE CONDICIONES INICIALES
            STX     APUN1
            STX     APUNM1
FIN1:
            RTS

;-------------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER2 POR EL PUERTO P2.

TRANS-P2:
            LDX     #$1000
            BCLR    PORTD,X,$30         ;HABILITA EL PUERTO P2
            BSET    PORTD,X,$8
            LDX     APUNM2
MAS2:
            LDD     #BUFFER2            ;HAY DATOS EN BUFFER2 PENDIENTES
            CPX     APUN2               ;POR TRANSMITIR?
            BEQ     FIN-MAN2
            ADDD    APUNM2
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA
            LDAA    #1
            CMPA    TRANS-OK            ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN2            ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM2              ;ACTUALIZA APUNM2
            BRA     MAS2
FIN-MAN2:
            CPX     #$1000              ;BUFFER2 ESTA LLENO?
            BNE     FIN2                ;NO => TERMINA TRANSMISIÓN POR P2
            JSR     LIMPIA              ;RESTABLECE CONDICIONES INICIALES
            STX     APUN2
            STX     APUNM2
FIN2:
            RTS
```

```
;-------------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER3

TRANS-P3:
          LDX       #$1000
          BCLR      PORTD,X,$28          ;HABILITA EL PUERTO P3
          BSET      PORTD,X,$10
          LDX       APUNM3
MAS3:
          LDD       #BUFFER3
          CPX       APUN3                ;HAY DATOS EN BUFFER3 PENDIENTES
          BEQ       FIN-MAN3             ;POR TRANSMITIR?
          ADDD      APUNM3
          XGDY
          LDAA      00,Y
          STAA      DATO-OUT
          JSR       ENVÍA                ;TRANSMITE UN DATO POR P3
          LDAA      #1
          CMPA      TRANS-OK             ;TRANSMISIÓN EXITOSA?
          BEQ       FIN-MAN3             ;NO => PERIF. NO PUEDE RECIBIR DATOS
          INX
          STX       APUNM3               ;ACTUALIZA APUNM3
          BRA       MAS3
FIN-MAN3:
          CPX       #$1000               ;BUFFER3 ESTA LLENO?
          BNE       FIN3                 ;NO => TERMINA TRANSMISIÓN POR P3
          JSR       LIMPIA               ;RESTABLECE CONDICIONES INICIALES
          STX       APUN3
          STX       APUNM3
FIN3:
          RTS

;-------------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER4

TRANS-P4:
          LDX       #$1000
          BCLR      PORTD,X,$20          ;HABILITA EL PUERTO P4
          BSET      PORTD,X,$18
          LDX       APUNM4
MAS4:
          LDD       #BUFFER4
          CPX       APUN4                ;HAY DATOS EN BUFFER4 PENDIENTES
          BEQ       FIN-MAN4             ;POR TRANSMITIR?
          ADDD      APUNM4
          XGDY
          LDAA      00,Y
          STAA      DATO-OUT
          JSR       ENVÍA                ;TRANSMITE UN DATO POR P4
          LDAA      #1
          CMPA      TRANS-OK             ;TRANSMISIÓN EXITOSA?
```

```
            BEQ     FIN-MAN4        ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM4          ;ACTUALIZA APUNM4
            BRA     MAS4
FIN-MAN4:
            CPX     #$1000          ;BUFFER4 ESTA LLENO?
            BNE     FIN4            ;NO => TERMINA TRANSMISIÓN POR P4
            JSR     LIMPIA          ;RESTABLECE CONDICIONES INICIALES
            STX     APUN4
            STX     APUNM4
FIN4:
            RTS

;-------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER5

TRANS-P5:
            LDX     #$1000
            BCLR    PORTD,X,$18     ;HABILITA EL PUERTO P5
            BSET    PORTD,X,$20
            LDX     APUNM5
MAS5:
            LDD     #BUFFER5
            CPX     APUN5           ;HAY DATOS EN BUFFER5 PENDIENTES
            BEQ     FIN-MAN5        ;POR TRANSMITIR?
            ADDD    APUNM5
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA           ;TRANSMITE UN DATO POR P5
            LDAA    #1
            CMPA    TRANS-OK        ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN5        ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM5          ;ACTUALIZA APUNM5
            BRA     MAS5
FIN-MAN5:
            CPX     #$1000          ;BUFFER5 ESTA LLENO?
            BNE     FIN5            ;NO => TERMINA TRANSMISIÓN POR P5
            JSR     LIMPIA          ;RESTABLECE CONDICIONES INICIALES
            STX     APUN5
            STX     APUNM5
FIN5:
            RTS

;-------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER6

TRANS-P6:
            LDX     #$1000
            BCLR    PORTD,X,$10     ;HABILITA EL PUERTO P6
```

```
            BSET    PORTD,X,$28
            LDX     APUNM6
MAS6:
            LDD     #BUFFER6
            CPX     APUN6           ;HAY DATOS EN BUFFER6 PENDIENTES
            BEQ     FIN-MAN6        ;POR TRANSMITIR?
            ADDD    APUNM6
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA           ;TRANSMITE UN DATO POR P6
            LDAA    #1
            CMPA    TRANS-OK        ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN6        ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM6          ;ACTUALIZA APUNM6
            BRA     MAS6
FIN-MAN6:
            CPX     #$1000          ;BUFFER6 ESTA LLENO?
            BNE     FIN6            ;NO => TERMINA TRANSMISIÓN POR P6
            JSR     LIMPIA          ;RESTABLECE CONDICIONES INICIALES
            STX     APUN6
            STX     APUNM6
FIN6:
            RTS

;-------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER7

TRANS-P7:
            LDX     #$1000
            BCLR    PORTD,X,$8      ;HABILITA EL PUERTO P7
            BSET    PORTD,X,$30
            LDX     APUNM7
MAS7:
            LDD     #BUFFER7
            CPX     APUN7           ;HAY DATOS EN BUFFER7 PENDIENTES
            BEQ     FIN-MAN7        ;POR TRANSMITIR?
            ADDD    APUNM7
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA           ;TRANSMITE UN DATO POR P7
            LDAA    #1
            CMPA    TRANS-OK        ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN7        ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM7          ;ACTUALIZA APUNM7
            BRA     MAS7
FIN-MAN7:
            CPX     #$1000          ;BUFFER7 ESTA LLENO?
            BNE     FIN7            ;NO => TERMINA TRANSMISIÓN POR P7
```

```
            JSR     LIMPIA              ;RESTABLECE CONDICIONES INICIALES
            STX     APUN7
            STX     APUNM7
FIN7:
            RTS

;-------------------------------------------------------------------------
;SUBRUTINA QUE ENVÍA LOS DATOS CONTENIDOS EN BUFFER8

TRANS-P8:
            LDX     #$1000
            BSET    PORTD,X,$38         ;HABILITA EL PUERTO P1
            LDX     APUNM8
MAS8:
            LDD     #BUFFER8
            CPX     APUN8               ;HAY DATOS EN BUFFER8 PENDIENTES
            BEQ     FIN-MAN             ;POR TRANSMITIR?
            ADDD    APUNM8
            XGDY
            LDAA    00,Y
            STAA    DATO-OUT
            JSR     ENVÍA               ;TRANSMITE UN DATO POR P8
            LDAA    #1
            CMPA    TRANS-OK            ;TRANSMISIÓN EXITOSA?
            BEQ     FIN-MAN8            ;NO => PERIF. NO PUEDE RECIBIR DATOS
            INX
            STX     APUNM8              ;ACTUALIZA APUNM8
            BRA     MAS8
FIN-MAN8:
            CPX     #$1000              ;BUFFER8 ESTA LLENO?
            BNE     FIN8                ;NO => TERMINA TRANSMISIÓN POR P7
            JSR     LIMPIA              ;RESTABLECE CONDICIONES INICIALES
            STX     APUN8
            STX     APUNM8
FIN8:
            RTS

;-------------------------------------------------------------------------
;INICIALIZA APUNTADORES E INDICA A LA PC QUE YA PUEDE ENVIAR MAS DATOS A LA
;IEPP

LIMPIA:
            LDX     #$1000              ;SEÑAL DE BUSY A NIVEL BAJO
            BCLR    TCTL1,X,$40         ;PA6=0 CUANDO TCNT SEA IGUAL TOC2
            LDD     TCNT,X              ;LEE CONTENIDO DE TCNT, 5c
            ADDD    #$20                ;SUMA, 4c
            STD     TOC2,X              ;QUEDA PROGRAMADA FUNCIÓN OC2, 5c
            LDX     #0                  ;3c
            RTS                         ;5c, STX(5c), STX(5c)
```

```
;-------------------------------------------------------------------------------
;TRANSMITE UN DATO HACIA EL PUERTO Pn SELECCIONADO
;TRANS-OK=0 => TRANSMISIÓN CON ÉXITO
;TRANS-OK=1 => DISPOSITIVO NO PUEDE RECIBIR INFORMACIÓN POR EL MOMENTO

ENVIA:
            PSHX
            LDX     #$1000
            CLR     TRANS-OK
            CLR     ACK
            LDAA    #1
            BRSET   PORTA,X,1,VEPE      ;SI ACKNLG=1 ESPERA RECONOCIMIENTO
            STAA    ACK                 ;ACKNLG=0 NO ESPERA
RECONOCIMIENTO
VEPE:
            BRCLR   PORTA,X,2,VEERROR       ;PE=0 => VERIFICA SEÑAL DE
ERROR
            STAA    TRANS-OK                ;PE=1 => NO TRANSMITE DATO
            BRA     FIN-ENV                 ;FIN DE RUTINA
VEERROR:
            BRSET   PORTA,X,4,ENV           ;ERROR=1 => TRANSMITE DATO
            STAA    TRANS-OK                ;ERROR=0 =>NO TRANSMITE DATO
            BRA     FIN-ENV                 ;FIN DE RUTINA
ENV:
            LDAA    DATO-OUT                ;PONE EL DATO EN TERMINALES
DE
            STAA    MEM                     ;SALIDA DEL CIRCUITO U9
(74HC273)
            BSET    TCTL1,X,$3              ;OC5=SET SUBO SEÑAL BUSY DE PC
            BSET    CFORC,X,$8              ;PA3=1
VEBUSY:
            BRSET   PORTA,X,$80,VEBUSY      ;BUSY DE PERIFÉRICO =0?
            BCLR    PORTD,X,$1              ;BUSY=0 => GENERA PULSO BAJO
DE
            PSHX                            ;LA SEÑAL STROBE PARA EL PERIF.
            PULX
            BCLR    TCTL1,X,$1              ;OC5=CLEAR BAJO BUSY DE PC
            BSET    CFORC,X,$8              ;PA3=0
            BSET    PORTD,X,$1              ;SUBE SEÑAL DE STROBE
            LDAA    #1
            CMPA    ACK                     ;NO ESPERA RECONOCIMIENTO
            BEQ     FIN-ENV                 ;TERMINA RUTINA
VEACKNLG:
            BRCLR   TFLG1,X,1,VEACKNLG      ;ESPERA PULSO BAJO DE ACKNLG
            BSET    TFLG1,X,1               ;COMO RECONOCIMIENTO =>IC3F=1
FIN-ENV:
            PULX
            RTS

;-------------------------------------------------------------------------------
;PONE EN ESTADO INICIAL CERO A LAS VARIABLES UTILIZADAS EN EL PROGRAMA
```

```
INIVAR:
        CLRB
CLEANV:
        LDAA    #0
        LDX     #$0100
        ABX
        STAA    00,X
        INCB
        CMPB    #$FF
        BNE     CLEANV
        RTS

;-------------------------------------------------------------------------------
;RUTINA DE ATENCIÓN A INTERRUPCIÓN:
;EL CARÁCTER TRANSMITIDO POR LA PC A TRAVÉS DEL PUERTO PARALELO ES LEÍDO Y
;ALMACENADO EN EL BUFFER DE RECEPCIÓN. SE GENERAN LOS ESTADOS DE LAS
SEÑALES ;BUSY Y ACKCNLG DE ACUERDO AL PROTOCOLO DE COMUNICACIÓN
ESTÁNDAR PARA ;PUERTO PARALELO.

INTERRUPT:
        ORG     INTERRUPT

RECIBE-DATO:
        LDX     #$1000
        LDAA    PORTE,X                 ;LEE DATO DE PUERTO E
        STAA    DATO
        BSET    TCTL1,X,$E0             ;PROGRAMACIÓN DE FUNCIÓN OC
        BCLR    TCTL1,X,$10             ;1110 0000 OC2=SET, OC3=0
        BSET    CFORC,X,$60             ;FORZA PA6=1 Y PA5=0
        BSET    TCTL1,X,$30             ;PULSO EN PA5 PARA U36 (74HC273)
        BSET    CFORC,X,$20             ;FORZA PA5=1
        LDAA    #8
        CMPA    BYCAD                   ;HAY 8 BYTES EN BUFFER?
        BNE     LEEDAT                  ;NO => BYCAD=0
        JSR     VACIABUFF               ;VACÍA EL BUFFER DE RECEPCIÓN
LEEDAT:
        LDAB    BYCAD
        CMPB    #7                      ;HAY 7 BYTES DE CAD DE CNTRL?
        BNE     NOACT                   ;NO => SIGUE PROCESO DE
ANÁLISIS
        JSR     IDEN-CAD                ;SI => IDENTIF POSIBLE CAD CNTRL
        LDAB    #1
        CMPB    TEMP                    ;SE COMPLETO LA CADENA CNTRL?
        BEQ     NOACT                   ;NO => SIGUE PROCESO DE
ANÁLISIS
        CLR     BYCAD                   ;SI => INICIALIZAN BYCAD Y CONT
        CLR     CONT
        BRA     FINALM                  ;TERMINA PROCESO ANÁLISIS
NOACT:
        LDY     #CADENA                 ;OFFSET DE CADENA DE CONTROL
        LDX     #BUFFER                 ;OFFSET DE BUFFER DE RECEPCIÓN
```

```
        LDAB    BYCAD                   ;# BYTES ALMACENADOS EN
BUFFER
        ABY
        ABX
        LDAA    DATO
        STAA    00,X                    ;ALMACENA DATO EN BUFFER
        INC     BYCAD                   ;ACTUALIZA BYCAD
        INC     CONT                    ;ACTUALIZA CONT
        CMPA    00,Y                    ;DATO = BYTE DE CADENA CNTRL
        BEQ     FINALM                  ;TERMINA PROCESO DE ANÁLISIS
        JSR     VACIABUFF               ;ALMACENA DATOS DE BUFFER EN
BUFFERn
        LDAA    #1
        CMPA    LLENO                   ;BUFFERn ESTA LLENO?
        BEQ     F-INT                   ;SI => SEÑAL BUSY DE PC QUEDARA EN 1
FINALM:                                 ;NO => SEÑAL BUSY DE PC EN CERO
        LDX     #$1000                  ;AL FINAL DE LA RUTINA
        BCLR    TCTL1,X,$40             ;PA6=0 CUANDO TCNT=TOC2
        LDD     TCNT,X                  ;LEE CONTENIDO DE TCNT, 5c
        ADDD    #$46                    ;70 CICLOS DE RELOJ DE SIG INST, 4c
        STD     TOC2,X                  ;PROG DE LA FUNCIÓN OC2, 5c
F-INT:
        LDD     TCNT,X                  ;LEE CONTENIDO DE TCNT, 5c
        ADDD    #$38                    ;56 CICLOS DE RELOJ DE SIG INST, 4c
        STD     TOC4,X                  ;PROGRAMADA FUNCIÓN OC4, 5c
        BSET    TCTL1,X,$08             ;7c
        BCLR    TCTL1,X,$04             ;7c
        BSET    CFORC,X,$10             ;FORZA PA4=0 ES DECIR  ACKNL=0 7c
        BSET    TCTL1,X,$4              ;SE PROGRAMA FUNCIÓN OC4=SET 7c
        RTI     ;12

;-------------------------------------------------------------------------
;VACÍA LOS DATOS CONTENIDOS EN EL BUFFER DE RECEPCIÓN "BUFFER" AL BUFFERn
;CORRESPONDIENTE.

VACIABUFF:
        PSHX                            ;PTO-PRED INDICA EL PUERTO Pn
        LDAA    PTO-PRED                ;SELECCIONADO EN ESE MOMENTO
        CMPA    #1                      ;ES P2 EL PUERTO SELECCIONADO?
        BNE     IDEN2                   ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P1                  ;SI => ALMACENA LOS DATOS EN BUFFER1
        BRA     FINIDEN                 ;TERMINA RUTINA DE VACIADO
IDEN2:
        CMPA    #2                      ;ES P2 EL PUERTO SELECCIONADO?
        BNE     IDEN3                   ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P2                  ;SI => ALMACENA LOS DATOS EN BUFFER2
        BRA     FINIDEN                 ;TERMINA RUTINA DE VACIADO
IDEN3:
        CMPA    #3                      ;ES P3 EL PUERTO SELECCIONADO?
        BNE     IDEN4                   ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P3                  ;SI => ALMACENA LOS DATOS EN BUFFER3
        BRA     FINIDEN                 ;TERMINA RUTINA DE VACIADO
```

```
IDEN4:
        CMPA    #4              ;ES P4 EL PUERTO SELECCIONADO?
        BNE     IDEN5           ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P4          ;SI => ALMACENA LOS DATOS EN BUFFER4
        BRA     FINIDEN         ;TERMINA RUTINA DE VACIADO
IDEN5:
        CMPA    #5              ;ES P5 EL PUERTO SELECCIONADO?
        BNE     IDEN6           ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P5          ;SI => ALMACENA LOS DATOS EN BUFFER5
        BRA     FINIDEN         ;TERMINA RUTINA DE VACIADO
IDEN6:
        CMPA    #6              ;ES P6 EL PUERTO SELECCIONADO?
        BNE     IDEN7           ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P6          ;SI => ALMACENA LOS DATOS EN BUFFER6
        BRA     FINIDEN         ;TERMINA RUTINA DE VACIADO
IDEN7:
        CMPA    #7              ;ES P7 EL PUERTO SELECCIONADO?
        BNE     IDEN8           ;NO => SIGUE PROCESO DE IDENTIFIC DE Pn
        JSR     ALM-P7          ;SI => ALMACENA LOS DATOS EN BUFFER7
        BRA     FINIDEN         ;TERMINA RUTINA DE VACIADO
IDEN8:
        JSR     ALM-P8          ;ALMACENA LOS DATOS EN BUFFER7
FINIDEN:
        PULX
        RTS

ALM-P1:                         ;ALMACENA DATOS DE BUFFER EN BUFFER1
        LDD     #BUFFER1
        LDX     APUN1
        JSR     ALM-DATO
        STX     APUN1
        RTS

ALM-P2:                         ;ALMACENA DATOS DE BUFFER EN BUFFER2
        LDD     #BUFFER2
        LDX     APUN2
        JSR     ALM-DATO
        STX     APUN2
        RTS

ALM-P3:                         ;ALMACENA DATOS DE BUFFER EN BUFFER3
        LDD     #BUFFER3
        LDX     APUN3
        JSR     ALM-DATO
        STX     APUN3
        RTS

ALM-P4:                         ;ALMACENA DATOS DE BUFFER EN BUFFER4
        LDD     #BUFFER4
        LDX     APUN4
        JSR     ALM-DATO
        STX     APUN4
        RTS
```

```
ALM-P5:                                 ;ALMACENA DATOS DE BUFFER EN BUFFER5
        LDD     #BUFFER5
        LDX     APUN5
        JSR     ALM-DATO
        STX     APUN5
        RTS

ALM-P6:                                 ;ALMACENA DATOS DE BUFFER EN BUFFER6
        LDD     #BUFFER6
        LDX     APUN6
        JSR     ALM-DATO
        STX     APUN6
        RTS

ALM-P7:                                 ;ALMACENA DATOS DE BUFFER EN BUFFER7
        LDD     #BUFFER7
        LDX     APUN7
        JSR     ALM-DATO
        STX     APUN7
        RTS

ALM-P8:                                 ;ALMACENA DATOS DE BUFFER EN BUFFER8
        LDD     #BUFFER8
        LDX     APUN8
        JSR     ALM-DATO
        STX     APUN8
        RTS

;-------------------------------------------------------------------------------
;VACÍA LOS DATOS CONTENIDOS EN BUFFER AL BUFFERn CORRESPONDIENTE.

ALM-DATO:
        STD     BUFFERn
        STX     APUNn
        CLR     LLENO
        LDAB    BYCAD           ;BYCAD-CONT=NUM DE BYTES EN BUFFER
        SUBB    CONT
VACÍA:
        CPX     #$1000          ;BUFFERn ESTA LLENO?
        BNE     NOLLEN          ;NO => CONTINUA VACIADO
        LDAA    #1              ;SI => TERMINA VACIADO Y SE INDICA QUE
        STAA    LLENO           ;BUFFERn ESTA SATURADO CON LLENO=1
        BRA     FINVACIA
NOLLEN:                         ;CICLO DE VACIADO
        PSHB
        LDD     BUFFERn
        ADDD    APUNn
        XGDY                    ;D=BUFFERn+APUNn
        PULB
        LDX     #BUFFER
        ABX                     ;X=BUFFER+BYCAD
```

```
            LDAA    00,X            ;EXTRAE DATO DE BUFFER+BYCAD
            STAA    00,Y            ;ALMACENA DATO EN BUFFER+APUNn
            LDX     APUNn
            INX
            STX     APUNn           ;APUNn=APUNn+1
            INCB
            DEC     CONT            ;CONT=CONT-1
            CMPB    BYCAD           ;SE VACIARON TODOS LOS DATOS?
            BNE     VACÍA           ;NO => CONTINUA PROCESO
            CLR     BYCAD           ;SI => INICIALIZA BYCAD=0
FINVACIA:                           ;TERMINA RUTINA DE VACIADO
            RTS

;-------------------------------------------------------------------------
;IDENTIFICA EL TIPO DE CADENA DE CONTROL INDICADO CON EL ULTIMO BYTE  DE LA
;SECUENCIA.

IDEN-CAD:
            PSHX
            CLR     TEMP
            LDAA    DATO
            CMPA    #$30
            BLS     FIN-IDEN        ;EL DATO ES MENOR A $30 =>TERMINA
            CMPA    #$39
            BHS     ESTATUS?        ;MAYOR A $39 => ES CADENA DE ESTATUS?
            ANDA    #$0F            ;QUITA EL NIBBLE ALTO
            STAA    PTO-PRED        ;ALMACENA EL NUEVO NUM DE PTO-PRED
            BRA     FIN-IDEN        ;TERMINA IDENTIFICACIÓN DE CADENA
ESTATUS?:
            LDX     #STATUS
            LDAA    DATO
            CMPA    #$41            ;SOLICITUD DE ESTATUS DEL PUERTO P1?
            BNE     STAT2           ;NO => CONTINUA PROCESO DE IDENTIF
            BSET    00,X,$1         ;PONE EN 1 EL BIT0 DE VARIABLE STATUS
            BRA     FIN-IDEN        ;TERMINA IDENTIFICACIÓN DE CADENA
STAT2:      CMPA    #$42            ;SOLICITUD DE ESTATUS DEL PUERTO P2?
            BNE     STAT3
            BSET    00,X,$2         ;PONE EN 1 EL BIT1 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT3:      CMPA    #$43            ;SOLICITUD DE ESTATUS DEL PUERTO P3?
            BNE     STAT4
            BSET    00,X,$4         ;PONE EN 1 EL BIT2 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT4:      CMPA    #$44            ;SOLICITUD DE ESTATUS DEL PUERTO P4?
            BNE     STAT5
            BSET    00,X,$8         ;PONE EN 1 EL BIT3 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT5:      CMPA    #$45            ;SOLICITUD DE ESTATUS DEL PUERTO P5?
            BNE     STAT6
            BSET    00,X,$10        ;PONE EN 1 EL BIT4 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT6:      CMPA    #$46            ;SOLICITUD DE ESTATUS DEL PUERTO P6?
```

```
            BNE     STAT7
            BSET    00,X,$20        ;PONE EN 1 EL BIT5 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT7:      CMPA    #$47            ;SOLICITUD DE ESTATUS DEL PUERTO P7?
            BNE     STAT8
            BSET    00,X,$40        ;PONE EN 1 EL BIT6 DE VARIABLE STATUS
            BRA     FIN-IDEN
STAT8:      CMPA    #$48            ;SOLICITUD DE ESTATUS DEL PUERTO P8?
            BNE     FIN-STAT
            BSET    00,X,$80        ;PONE EN 1 EL BIT7 DE VARIABLE STATUS
            BRA     FIN-IDEN
FIN-STAT:
            LDAB    #1              ;NO SE IDENTIFICO CADENA DE CONTROL
            STAB    TEMP
FIN-IDEN:
            PULX
            RTS

;-------------------------------------------------------------------------
;ALMACENA LA DIRECCIÓN DE INTERRUPT EN EL VECTOR $FFF2

            ORG     $FFF2           ;VECTOR DE LA INTERRUPCIÓN IRQ
IRQ         FDB     INTERRUPT

;ALMACENA LA DIRECCIÓN DE INICIALIZACIÓN EN EL VECTOR $FFFE

            ORG     $FFFE           ;VECTOR DEL RESET
RESET       FDB     INICIALIZACION
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%&%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                                          %%%
%%%                                    MENU.ASM                                              %%%
%%%     PROGRAMA EJECUTADO POR LA PC QUE PERMITE SELECCIONAR UNO                             %%%
%%%     DE LOS PUERTOS DE LA IEPP (P1-P8) COMO PUERTO PREDETERMINADO                         %%%
%%%                                                                                          %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%&%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


grupo1          group  sega,segb,segc,segd
assume          cs:grupo1,ds:segb,ss:segc,es:segd


sega segment word public 'code'
start:

        mov     ax,segb                 ;inicializa el ds
        mov     ds,ax
        mov     ax,segd                 ;inicializa el es
        mov     es,ax
        mov     ax,segc                 ;inicializa el ss
        mov     ss,ax
        mov     sp,100h                 ;inicializa el sp
inicio:
        call    menuprin                ;Despliega menú principal
        call    sensatec                ;Verifica si se sensó una tecla
        cmp     dato,1bh                ;Fue ESC la tecla presionada?
        jz      termina                 ;Termina programa
        call    opcionp                 ;Verifica si se eligió una opción del menú principal
        jmp     inicio
termina:
        mov     atributo,07h            ;Propiedades: Fondo negro y caracteres blancos
        call    clscrin                 ;Limpia la pantalla
        mov     ah,4ch                  ;Termina sesión
        int     21h


;------------------------------------------------------------------------------------------
;Verifica si la opción elegida es válida

opcionp:
        cmp     dato,41h                ;La tecla presionada fue "A" o "a"?
        jz      opc1                    ;Si => opción 1 elegida
        cmp     dato,61h
        jnz     opc2?                   ;No => pregunta si la opción 2 fue elegida
opc1:
        call    menupuerto              ;Despliega el menú para seleccionar un puerto de salida de la IEPP
opc1a:
        call    sensatec                ;Espera a que se presione una tecla
        cmp     dato,1bh                ; Si es ESC regresa a menú principal
```

```
        jz      finopc
        call    idpuerto        ;Identifica cual de los puertos fue seleccionado
        cmp     seleccion,0     ;seleccion=0, indica que la tecla no corresponde con ninguna opción
        jz      opc1a           ;Regresa para esperar a que se presione una tecla válida.
        call    retar           ;Genera un retardo para que el mensaje desplegado pueda ser leído.
        jmp     finopc          ;Termina rutina.
opc2?:                          ;Verifica si la opción 2 fue seleccionada.
        cmp     dato,42h        ;La tecla fue "B" o "b"?
        jz      opc2            ;Si => La opción de solicitud de estatus fue requerida por el usuario.
        cmp     dato,62h
        jnz     finopc          ;termina rutina.
opc2:
        call    menuestado      ;Despliega el menú de estatus
        call    stpuertos       ;Solicita es estado de cada Pn a la IEPP
opc2a:
        call    sensatec        ;Espera a que se presione la tecla ESC
        cmp     dato,1bh
        jnz
finopc:
        ret


;-------------------------------------------------------------------------------
;Identifica cual de los puertos Pn fue seleccionado para transmitir a la IEPP la cadena de control
;correspondiente.

idpuerto:
        mov     seleccion,1
        cmp     dato,31h        ;El puerto P1 fue seleccionado
        jnz     op2
        call    envcad          ;Se transmite la cadena ^<*&%$#1
        jmp     ptook?
op2:    cmp     dato,32h        ;El puerto P1 fue seleccionado
        jnz     op3
        call    envcad          ;Se transmite la cadena ^<*&%$#2
        jmp     ptook?
op3:    cmp     dato,33h        ;El puerto P1 fue seleccionado
        jnz     op4
        call    envcad          ;Se transmite la cadena ^<*&%$#3
        jmp     ptook?
op4:    cmp     dato,34h        ;El puerto P1 fue seleccionado
        jnz     op5
        call    envcad          ;Se transmite la cadena ^<*&%$#4
        jmp     ptook?
op5:    cmp     dato,35h        ;El puerto P1 fue seleccionado
        jnz     op6
        call    envcad          ;Se transmite la cadena ^<*&%$#5
        jmp     ptook?
op6:    cmp     dato,36h        ;El puerto P1 fue seleccionado
        jnz     op7
        call    envcad          ;Se transmite la cadena ^<*&%$#6
        jmp     ptook?
op7:    cmp     dato,37h        ;El puerto P1 fue seleccionado
```

```
        jnz     op8
        call    envcad              ;Se transmite la cadena ^<*&%$#7
        jmp     ptook?
op8:    cmp     dato,38h            ;El puerto P1 fue seleccionado
        jnz     actban              ;Pone la variable seleccion=0
        call    envcad              ;Se transmite la cadena ^<*&%$#8
ptook?:
        cmp     comunic,1           ;comunic=1 indica que no se transmitió la cadena
        jnz     transok
        mov     bp,offset mena1     ;mensaje DESCONECTADA O EN SATURACIÓN
        mov     cx,mena1end-mena1
        call    escaviso
        jmp     finiden
transok:
        mov     bp,offset mena2     ;mensaje CADENA DE CONTROL TRANSMITIDA....
        mov     cx,mena2end-mena2
        call    escaviso
        jmp     finiden
actban:
        mov     seleccion,0         ;La tecla no correspondió con ninguna de las opciones
finiden:
        ret


;-------------------------------------------------------------------------------------------------------------
;Solicita el estado de cada puerto Pn a la IEPP mediante la cadena de control correspondiente.

stpuertos:
        mov     posicion,0929h
        mov     dato,41h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#A
        mov     posicion,0a29h
        mov     dato,42h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#B
        mov     posicion,0b29h
        mov     dato,43h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#C
        mov     posicion,0c29h
        mov     dato,44h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#D
        mov     posicion,0d29h
        mov     dato,45h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#E
        mov     posicion,0e29h
        mov     dato,46h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#F
        mov     posicion,0f29h
        mov     dato,47h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#G
        mov     posicion,1029h
        mov     dato,48h
        call    pidestatus          ;Pide estado de P1 mediante la cadena ^<*&%$#H
        ret
```

```
;-------------------------------------------------------------------------------
;Pide el estado de cada uno de los puertos Pn a la IEPP en forma consecutiva desde P1 hasta P8

pidestatus:
        call    envcad                  ;Envía la cadena correspondiente
        cmp     comunic,1               ;comunic=1 => transmisión no tuvo éxito y lo reintenta
        jz      pidestatus
        mov     dx,0379h                ;Registro de estatus del LPT1
stat1:  in      al,dx                   ;Espera a que Error=0
        or      al,0f7h
        cmp     al,0f7h
        jnz     stat1
        call    time                    ;Retardo para leer el valor de PE
        in      al,dx
        and     al,20h                  ;lee estado de PE
        cmp     al,20h
        jnz     statok                  ;PE=0 =>Dispositivo O.K.
        mov     dx,posicion             ;PE=1 => Dispositivo en estado de Error o sin papel
        mov     cx,menb2end-menb2
        mov     bp,offset menb2
        call    escestado               ;mensaje ERROR O SIN PAPEL
        jmp     fpidest
statok:
        mov     dx,posicion
        mov     bp,offset menb1
        mov     cx,menb1end-menb1
        call    escestado               ;mensaje EN LINEA
fpidest:
        ret


time:
        push    cx
        mov     cx,0ffh
t1:     dec     cx
        cmp     cx,0
        jnz     t1
        pop     cx
        ret


;-------------------------------------------------------------------------------
;Envía cadena de control por el puerto paralelo utilizando la int 17h del BIOS

envcad:
        push    ax
        push    dx
        push    di
        mov     comunic,0
        mov     dx,0379h                ;Hay periférico conectado al Pto. Paralelo?
        In      al,dx                   ;Puerto de entrada del LPT1
        cmp     al,0dfh                 ;Busy=0, Ack=1, Slct=1
        jz      env1                    ;envía cadena
```

```
        mov   comunic,1                ; IEPP está en saturación o esta desconectada de la PC
        jmp   fin
 env1:  mov   di,offset cadena         ;Primeros 7 bytes de la cadena de control
        mov   dx,0
 env2:  mov   al,[di]                  ;INT 17h para enviar un carácter por el puerto paralelo
        cmp   al,40h
        jz    finenv
        mov   ah,0
        int   17h
        inc   di
        jmp   env2
 finenv:
        mov   al,dato                  ;Transmite el último byte de la cadena de control
        mov   ah,0
        int   17h
 fin:   pop   di
        pop   dx
        pop   ax
        ret

;-------------------------------------------------------------------------------------------------------
;Espera a que se presiona una tecla

sensatec:
        mov   ah,01h
        int   16h
        jz    sensatec
        mov   ah,0                     ;Lee carácter de buffer de teclado
        int   16h
        mov   dato,al                  ;Almacena el carácter en la variable dato
        ret

;-------------------------------------------------------------------------------------------------------
;Retardo para que el usuario alcance a leer el mensaje desplegado en la pantalla

retar:
        push  ax
        push  bx
        push  cx
        mov   cx,0ffh
ret1:
        push  cx
        mov   cx,0ffffh
ret3:   mov   ax,bx
        mov   bx,ax
        loop  ret3
        pop   cx
        loop  ret1
        pop   cx
        pop   bx
        pop   ax
        ret
```

```
;-------------------------------------------------------------------------------
;Despliega en pantalla las opciones del menú principal

menuprin:
        mov     atributo,1ah
        call    clscrin
        push    es
        mov     ax,ds
        mov     es,ax
        mov     bp,offset menp
        mov     ah,13h
        mov     al,01
        mov     bh,0
        mov     bl,1eH
        mov     cx,menpend-menp
        mov     dx,030dh
        int     10h
        pop     es
        ret

;-------------------------------------------------------------------------------
;Despliega en pantalla información sobre el estado de operación de los puertos Pn de la IEPP

menuestado:
        mov     atributo,1ah
        call    clscrin
        push    es
        mov     ax,ds
        mov     es,ax
        mov     bp,offset menb
        mov     ah,13h
        mov     al,01
        mov     bh,0
        mov     bl,1eH
        mov     cx,menbend-menb
        mov     dx,0311h
        int     10h
        pop     es
        ret


;-------------------------------------------------------------------------------
;Despliega en pantalla el menú que le permite al usuario seleccionar uno de los 8 puertos Pn de la IEPP

menupuerto:
        mov     atributo,1ah
        call    clscrin
        push    es
        mov     ax,ds
        mov     es,ax
        mov     bp,offset mena
        mov     ah,13h
```

```
        mov   al,01
        mov   bh,0
        mov   bl,1eH
        mov   cx,menaend-mena
        mov   dx,0307h
        int   10h
        pop   es
        ret

;-------------------------------------------------------------------------------
;Escribe el mensaje correspondiente que indica el estado en el que opera cada Pn

escestado:
        push  es
        mov   ax,ds
        mov   es,ax
        mov   ah,13h
        mov   al,01
        mov   bh,0
        mov   bl,1eH
        int   10h
        pop   es
        ret

;-------------------------------------------------------------------------------
;Escribe un mensaje de aviso en la pantalla

escaviso:
        push  es
        mov   ax,ds
        mov   es,ax
        mov   ah,13h
        mov   al,01
        mov   bh,0
        mov   bl,9eH
        mov   dx,1107h
        int   10h
        pop   es
        ret


;-------------------------------------------------------------------------------
;Limpia la pantalla

clscrin:
        push  ax
        push  bx
        push  cx
        push  dx
        mov   ah,06h
        mov   al,00h
        mov   bh,atributo
        mov   ch,00h
```

```
        mov   cl,00h
        mov   dh,18h
        mov   dl,4fh
        int   10h
        pop   dx
        pop   cx
        pop   bx
        pop   ax
        ret

sega    ends

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

segb    segment byte public 'data'

menp  db      "INTERFACE ELECTRONICA PARA PUERTO PARALELO (IEPP)",0dh,0ah
      db      "            -----------------------------------------------"
      db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
      db      "        A) SELECCIONAR UNO DE LOS OCHO PUERTOS DE LA IEPP"
      db      0dh,0ah,0dh,0ah
      db      "        B) DESPLEGAR EL ESTADO DE LOS PUERTOS DE LA IEPP"
      db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
      db      0dh,0ah,"                PRESIONE ESC PARA ABORTAR"
menpend       db    0

mena  db      "SELECCIONE EL PUERTO POR EL QUE DESEA TRANSMITIR SU
      db      INFORMACION"
      db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
      db      "                 1) PUERTO DE SALIDA P1",0dh,0ah
      db      "                 2) PUERTO DE SALIDA P2",0dh,0ah
      db      "                 3) PUERTO DE SALIDA P3",0dh,0ah
      db      "                 4) PUERTO DE SALIDA P4",0dh,0ah
      db      "                 5) PUERTO DE SALIDA P5",0dh,0ah
      db      "                 6) PUERTO DE SALIDA P6",0dh,0ah
      db      "                 7) PUERTO DE SALIDA P7",0dh,0ah
      db      "                 8) PUERTO DE SALIDA P8",0dh,0ah
      db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
      db      "        PRESIONE ESC PARA REGRESAR AL MENÚ PRINCIPAL"
menaend       db    0
mena1         db    "     IEPP NO RESPONDE (DESCONECTADA O EN SATURACION)"
mena1end      db    0
mena2         db    "       CADENA DE CONTROL TRANSMITIDA CON EXITO"
mena2end      db    0

menb  db      "ESTADO DE LOS PUERTOS DE SALIDA DE LA IEPP"
      db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
      db      "            PUERTO              ESTADO"
      db      0dh,0ah,0dh,0ah
      db      "              P1",0dh,0ah
      db      "              P2",0dh,0ah
      db      "              P3",0dh,0ah
      db      "              P4",0dh,0ah
```

```
        db      "                P5",0dh,0ah
        db      "                P6",0dh,0ah
        db      "                P7",0dh,0ah
        db      "                P8",0dh,0ah
        db      0dh,0ah,0dh,0ah,0dh,0ah,0dh,0ah
        db      "            PRESIONE ESC PARA REGRESAR A MENU PRINCIPAL"
menbend         db      0
menb1           db      "   EN LINEA"
menb1end        db      0
menb2           db      "ERROR O SIN PAPEL"
menb2end        db      0
cadena          db      "^<*&%$#@"
dato            db      0
comunic         db      0
posicion        dw      0
atributo        db      0
selección       db      0
segb            ends
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

segc segment word stack 'stack'

segc            ends
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

segd segment byte public 'data'

segd            ends
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

end             start
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                                                                          %%%%
%%%%                              ROBOT.ASM                                   %%%%
%%%%   TRANSMITE LA SECUENCIA ADECUADA DE DATOS AL BRAZO ROBOT                %%%%
%%%%   PARA QUE REALICE EN FORMA CÍCLICA UNA RUTINA DE MOVIMIENTO             %%%%
%%%%                                                                          %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

grupo1      group  sega,segb,segc,segd
assume      cs:grupo1,ds:segb,ss:segc,es:segd

sega segment word public 'code'
start:
        mov     ax,segb                 ;inicializa el ds
        mov     ds,ax
        mov     ax,segd
        mov     es,ax
        mov     ax,segc
        mov     ss,ax
        mov     sp,100h                 ;inicializa el sp
inicia:
        mov     ax,20h
vuelta1:
        call    rutanor
        dec     ax
        jne     vuelta1
        call    retar1
        mov     ax,20h
vuelta2:
        call    rutainv
        dec     ax
        jne     vuelta2
        callL   retar1
status:
        mov     ah,01h                  ;Espera a que se presione una tecla
        int     16h
        jz      inicia
        mov     ah,0                    ;Lee carácter de buffer de teclado
        int     16h
        cmp     al,1bh
        jnz     inicia
        mov     ah,4ch                  ;Termina sesión
        int     21h

;-----------------------------------------------------------------------------------
;Mueve el brazo robot en sentido horario
rutanor:
        push    ax
        push    dx
        push    di
        mov     dx,0
        mov     di,offset tabla1
```

```
rut1:
        mov   al,[di]
        mov   ah,0
        int   17h
        call  retar2
        dec   ax
        mov   ah,0
        int   17h
        inc   di
        call  retar2
        cmp   al,96h
        jne   rut1
        pop   di
        pop   dx
        pop   ax
        ret

;-------------------------------------------------------------------------------
;Mueve el brazo en sentido antihorario
rutainv:
        push  ax
        push  dx
        push  di
        mov   dx,0
        mov   di,offset tabla2
rut2:
        mov   al,[di]
        mov   ah,0
        int   17h
        out   dx,al
        call  retar2
        dec   ax
        mov   ah,0
        int   17h
        inc   di
        call  retar2
        cmp   al,12h
        jne   rut2
        pop   di
        pop   dx
        pop   ax
        ret

;-------------------------------------------------------------------------------
;Retardo para iniciar el recorrido
retar1:
        push  ax
        push  bx
        push  cx
        mov   cx,06
ret1:
        push  cx
        mov   cx,0fffh
```

```
ret3:
        mov   ax,bx
        mov   bx,ax
        loop  ret3
        pop   cx
        loop  ret1
        pop   cx
        pop   bx
        pop   ax
        ret

;-------------------------------------------------------------------------------
;Retardo para enviar el siguiente dato de la tabla
retar2:
        push  ax
        push  bx
        push  cx
        mov   cx,08fffh
ret2:
        mov   ax,bx
        mov   bx,ax
        loop  ret2
        pop   cx
        pop   bx
        pop   ax
        ret

sega    ends
;-------------------------------------------------------------------------------
segb segment byte public 'data'
TABLA1  db    0F3h,0F5h,0F7h,0F9h,0FBh,0FDh,13h,15h,17h,33h,35h,37h,23h,25h
        db    27h,63h,65h,67h,43h,45h,47h,0C3h,0C5h,0C7h,83h,85h,87h,93h,95h
        db    97h
TABLA2  db    97h,95h,93h,87h,85h,83h,0C7h,0C5h,0C3h,47h, 45h,43h,67h,65h,63h
        db    27h,25h,23h,37h,35h,33h,17h,15h,13h
segb    ends
;-------------------------------------------------------------------------------
segc segment word stack 'stack'

segc    ends
;-------------------------------------------------------------------------------
segd segment byte public 'data'

segd    ends
;-------------------------------------------------------------------------------
end     start
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                         %%%
%%%                              ESCLAVA.ASM                                %%%
%%%      PROGRAMA QUE LE PERMITE A UNA COMPUTADORA TIPO PC RECIBIR          %%%
%%%               DATOS A TRAVÉS DEL PUERTO PARALELO                        %%%
%%%                                                                         %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

grupo1      group sega,segb,segc,segd
assume      cs:grupo1,ds:segb,ss:segc,es:segd

sega segment word public 'code'
start:
        mov     ax,segb             ;inicializa el ds
        mov     ds,ax
        mov     ax,segd
        mov     es,ax
        mov     ax,segc
        mov     ss,ax
        mov     sp,100h             ;inicializa el sp
        mov     atributo,07h
        call    limpia
        mov     pos,0101h
        call    poscur
        mov     dx,37ah
        mov     al,02h              ;inicializa busy=0 y Acknlg=1
        out     dx,al

inicia:
        call    sensapto            ;verifica si llegó un dato por el puerto paralelo
        mov     ah,0
        int     16h                 ;Interrupción para saber si se presionó una tecla
        cmp     al,1bh              ;Si fue ESC termina el programa
        jnz     inicia
        mov     atributo,07h        ;fondo negro y caracteres blancos
        call    limpia              ;limpia la pantalla
        mov     ah,4ch              ;Termina sesión
        int     21h

;------------------------------------------------------------------------------
;Lee el byte de control del LPT1 para determinar si hay un dato listo en el puerto 378H

sensapto:
        mov     dx,379h             ;byte de control del LPT1
        in      al,dx
        or      al,0bfh             ;lee terminal 10  (STROBE generado por IEPP)
        cmp     al,0bfh             ;Si el bit correspondiente vale cero significa que hay dato
        jnz     finsenpto
        call    leedato             ;lee el dato por el puerto 378H
finsenpto:
        ret
;El dato transmitido por la IEPP es leído por la PC esclava a través del puerto 378h del LPT1
```

```
leedato:
        push    dx
        mov     dx,37ah         ;byte de estatus del LPT1
        mov     al,0            ;busy=1 Acknlg=1
        out     dx,al           ;Indica a la IEPP que la PC está ocupada recibiendo un dato
        mov     dx,378h         ;lee el dato por el 378h
        in      al,dx
        mov     dato,al
        mov     al,01h
        mov     dx,37ah
        out     dx,al           ;Pone en bajo la señal de Acknlg
        call    pincar          ;Despliega el carácter en la pantalla de la PC esclava.
        mov     al,02h          ;Pone en nivel alto la señal de Acknlg y en bajo la señal de Busy
        out     dx,al
        pop     dx
        ret

;-------------------------------------------------------------------------------
;Despliega el carácter recibido por el puerto paralelo en la pantalla de la PC, el despliegue es en forma
;continua.

pincar:
        push    ax
        push    bx
        push    cx
        mov     ah,09h
        mov     al,dato
        mov     bh,0
        mov     bl,atributo
        mov     cx,1
        int     10h
        call    poscur
        pop     cx
        pop     bx
        pop     ax
        ret

;-------------------------------------------------------------------------------
;Posiciona el cursor adecuadamente haciendo el cambio de línea de ser necesario

poscur:
        push    ax
        push    bx
        push    dx
        mov     bx,0
        mov     dx,pos
        inc     dl
        cmp     dl,4eh
        jne     actpos
        mov     dl,2
        inc     dh
        cmp     dh,18h
```

```
        jne     actpos
        mov     dh,1
        call    limpia
actpos:
        mov     pos,dx
        mov     ah,02h
        int     10h
        pop     dx
        pop     bx
        pop     ax
        ret

;-------------------------------------------------------------------------------
;Limpia la pantalla

limpia:
        push    ax
        push    bx
        push    cx
        push    dx
        mov     ah,06h
        mov     al,00h
        mov     bh,atributo             ;0000 0111 fondo negro caracteres blancos
        mov     ch,00h
        mov     cl,00h
        mov     dh,18h
        mov     dl,4fh
        int     10h
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        ret

sega    ends
;-------------------------------------------------------------------------------
segb segment byte public 'data'
atributo        db      0
dato            db      0
pos             dw      0
segb            ends
;-------------------------------------------------------------------------------
segc segment word stack 'stack'
segc    ends
;-------------------------------------------------------------------------------
segd segment byte public 'data'
segd    ends
;-------------------------------------------------------------------------------
end     start
```

APÉNDICE B
MANUAL DE USUARIO

El manejo de la interface electrónica para puerto paralelo (IEPP) es demasiado sencillo, sin embargo es mejor que el usuario disponga de las instrucciones necesarias para realizar las conexiones debidamente y así evitar cualquier problema. A continuación se describen en orden consecutivo, cada uno de los pasos que se llevarán a cabo para operar la interface.

1) Conectar el puerto de entrada de la IEPP al puerto paralelo de la PC maestra, utilizando un cable de 25 hilos con un conector DB25 macho en cada uno de los extremos del cable. Las terminales de los conectores DB25 deberán estar conectadas una a una, es decir, terminal #1 con terminal #1, terminal #2 con terminal #2, y así sucesivamente.
2) Conectar los dispositivos periféricos en los puertos de salida de la IEPP, tomando en cuenta lo siguiente:
   a) Si el periférico es un dispositivo de impresión, se utiliza un cable normal para impresora. El extremo del cable con el conector DB25 macho se conecta al puerto de salida seleccionado de la IEPP y el extremo con el conector Amphenol 57-30360 de 36 conectores en el puerto del dispositivo de impresión.
   b) Si el periférico es una PC esclava, se utiliza un cable construido como se describe en la Tabla 5.1.
   c) Si el periférico es un dispositivo que sólo recibe datos, como es el caso de sistemas que operan con motores de pasos, se utilizará el cable que normalmente utiliza dicho dispositivo cuando es conectado directamente al puerto paralelo de la computadora. Obviamente en este caso se conectará en uno de los puertos de salida de la IEPP.
3) Encender la PC maestra.
4) Encender cada uno de los dispositivos periféricos.
5) Energizar la IEPP, conectando un rectificador de 9 VDC en el borne de polarización.
6) Ejecutar en la PC el programa MENU.EXE, el cual permite al usuario seleccionar el puerto de salida de la IEPP que desee como puerto predeterminado, esto significa que la información que la PC transmita a través del puerto paralelo, será recibida por la

IEPP y a su vez ésta la transmitirá íntegramente al periférico que se encuentre conectado en el puerto Pn seleccionado por el usuario.

7) Si se desea ejecutar el programa MENU.EXE desde el sistema operativo WINDOWS, realizar lo siguiente.

   a) Estando en el Administrador de programas, seleccionar Archivo y a continuación Nuevo.

   b) Dentro de la ventana Nuevo elemento, seleccionar Grupo de programas y presionar Aceptar.

   c) Dentro de la ventana Propiedades del grupo de programas se elige el nombre del grupo de programas. En Descripción poner IEPP. En Archivo de grupo también poner IEPP y presionar Aceptar.

   d) En este momento queda abierta la ventana IEPP. Para agregar un elemento al grupo de programas IEPP, se repite nuevamente el paso del inciso a). En este caso dentro de la ventana Nuevo elemento se selecciona la opción Elemento de programa y se presiona Aceptar.

   e) Dentro de la ventana Propiedades del elemento de programa se elige el nombre del elemento y sus propiedades. En Descripción poner el nombre del elemento MENU. En Línea de comando poner la dirección donde se encuentra MENU, por ejemplo c:\iepp\menu.exe. En Directorio de trabajo poner nuevamente la dirección del directorio donde se encuentra MENU, en este caso, c:\iepp. Finalmente seleccionar la opción Cambiar icono.

   f) Aparecerá la ventana Cambiar el icono para avisar que no hay icono disponible para el programa y que se podrá seleccionar uno del Administrador de programas, presionar Aceptar. Una vez que se seleccionó el icono deseado presionar Aceptar. Presionar Aceptar nuevamente dentro de la ventana Propiedades del elemento de programa. A partir de este momento se podrá ejecutar el programa MENU.EXE dentro de WINDOWS sin necesidad de hacer un SHELL al sistema operativo DOS.

8) En cualquier momento se puede realizar un reset manual al sistema, para lo cual el usuario deberá tomar en cuenta que la información que se encuentre almacenada en la memoria del sistema se perderá.
9) Para la desconexión de los dispositivos, deberá primero desenergizar la IEPP, a continuación los dispositivos periféricos y finalmente la PC maestra.

APÉNDICE C

DIAGRAMAS DE CONEXIONES DEL SISTEMA

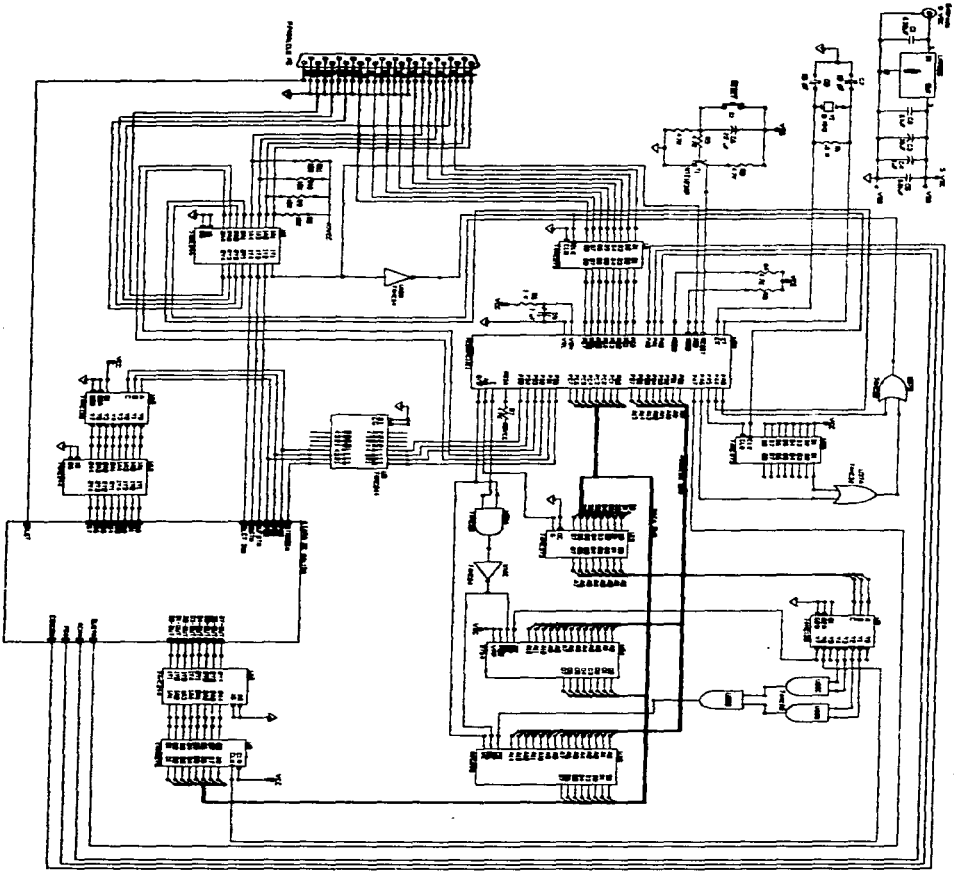


DIAGRAMA DE CONEXIONES PARTE 1

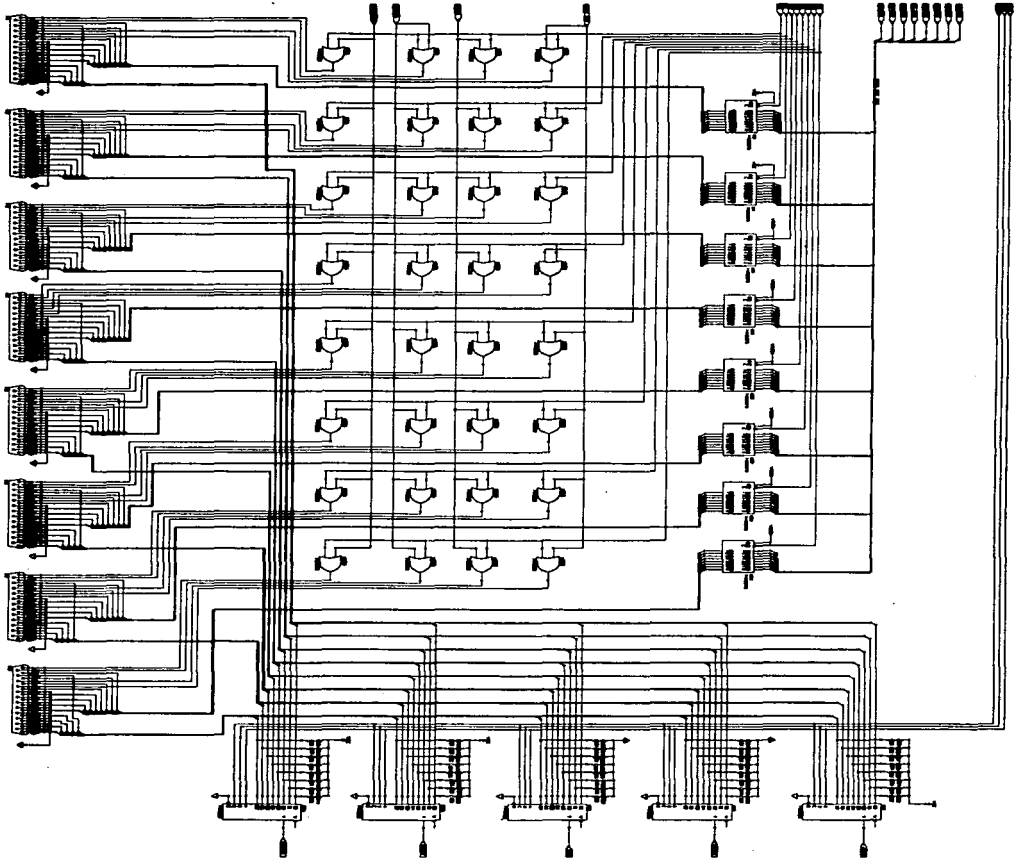


DIAGRAMA DE CONEXIONES PARTE 2

BIBLIOGRAFIA:

M68HC11 Reference Manual. Motorola Inc. 1991.

M68HC11 E Series Technical Data. Motorola. 1993.

CMOS Logic Databook. National Semiconductor. 1988.

The Programmer's PC Sourcebook.
Thom Hogan.
Microsoft Press. 1988.

Data Acquisition and Process Control with the M68HC11 Microcontroler.
Driscoll. Coughlin. Villanucci.
Macmillan Publishing Company. 1994.

The Programmer's PC Sourcebook.
Thom Hogan.
Microsoft Press. 1988.

Assembly Language for the IBM-PC.
Kip R. Irvine.
Macmillan Publishing Company. 1990.

Microprocessors and Peripherals.
Brey.
Macmillan Publishing Company.

Microprocessors and Microcomputer, Based System Design.
Mohamed.

Microcomputer Journal. "From Printer Port to Enhanced Parallel Port".
Página 37. Julio-Agosto 1995.

Computercraft. "An Automatic Printer Selector".
Página 26. Octubre 1991.