UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CONTROLADOR DE PROCESOS INDUSTRIALES

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N :

ISMAEL ROMERO OCARANZA

RUBEN VALDES GARCIA

DIRECTOR DE TESIS:

ING. FELIPE RAUDA GARCIA

MEXICO, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

RESUMEN.

Esta tesis plantea el diseño y construcción de una tarjeta con la capacidad de monitorear y controlar un proceso industrial específico o de laboratorio, con la ayuda de una computadora personal del tipo IBM PC/XT/AT. La tarjeta se conecta en una de las ranuras de expansión de la computadora, y cuenta con las siguientes características:

a) Capacidad de monitorear cuatro sensores diferentes, conectados a un convertidor analógico digital con resolución de doce bits.

b) Un convertidor digital analógico de doce bits.

c) Ocho señales de control para activar y desactivar los equipos encargados del control del proceso industrial o de laboratorio (relevadores o etapas de potencia).

d) Los circuitos necesarios para el monitoreo del proceso, en forma aislada (optoacopladores).

Además la programación de los parámetros requeridos por la tarjeta se determinan a través del programa de aplicación escrito en un lenguaje de alto nivel.

Los bloques que integran la tarjeta se describen a continuación:

a) Decodificador. Las computadoras del tipo IBM PC XT/AT cuentan con ranuras para conectar tarjetas de expansión (una ranura de expansión es un conector donde se tiene acceso al bus de direcciones, bus de datos y bus de control). La tarjeta se conecta en una de las ranuras de expansión de la computadora; el decodificador recibe señales del bus de direcciones y control.

activando la tarjeta cuando la PC direcciona los puertos comprendidos entre las direcciones $300 y $31F (reservadas para tarjetas prototipo), además se encarga de seleccionar adecuadamente los diferentes dispositivos con que cuenta la tarjeta (convertidores, flip-flops, etc.). El decodificador se implementa con un arreglo lógico programable (PAL) y un decodificador de 3 X 8 (74LS138).

b) Sistema de adquisición. Por medio del sistema de adquisición la tarjeta tiene la capacidad de monitorear 4 sensores diferentes y 8 circuitos para el monitoreo del proceso industrial (o de laboratorio) en forma aislada. Los 4 sensores se conectan a un multiplexor analógico de 4 a 1, el cual proporciona la señal analógica (seleccionada de entre las cuatro entradas por medio de dos líneas de control) que se conecta a la entrada del convertidor analógico digital (ADC); el ADC recibe una señal analógica y nos entrega, en doce líneas de salida, la combinación digital asociada a dicha señal.

c) Bloque de salidas de control. La tarjeta cuenta con ocho señales de salida para activar y desactivar el equipo encargado del control del proceso industrial (o de laboratorio); las señales de control se proporcionan por medio de un circuito que cuenta con 8 Flip-flops tipo D (74LS264). Además, la tarjeta entrega una señal analógica generada por un convertidor digital-analógico (DAC-800).

El programa de aplicación está escrito en un lenguaje de alto nivel y para su estudio podemos dividirlo en dos partes principales:

a)Rutinas encargadas del monitoreo del proceso industrial o de laboratorio. Estas rutinas se implementan utilizando un esquema de interrupciones, esto es, periódicamente se interrumpe la operación normal de la computadora para actualizar el valor de los sensores y circuitos encargados del monitoreo aislado del proceso industrial. Una vez que se toman los nuevos valores del bloque de adquisición se regresa el control al programa que se estaba ejecutando en el momento de presentarse la interrupción.

b)Rutinas de control. Estas rutinas se encargan de activar o desactivar el equipo encargado del

control del proceso industrial o de laboratorio de acuerdo al valor que presenten los sensores y los circuitos de monitoreo en forma aislada.

Además el programa principal cuenta con rutinas para actualizar el estado del equipo encargado del control del proceso, sensores y optoaisladores en el monitor de la computadora.

El trabajo se presenta con una solución práctica, en la adquisición de imágenes digitalizadas de señales ultrasónicas, en la cual se comprueba el correcto funcionamiento de la tarjeta desarrollada. Así mismo se hace incapie en la gran variedad de aplicaciones que se le puede dar en la solución de problemas en el campo de la instrumentación electrónica.

INDICE.

I. INTRODUCCIÓN.

1.1 Sistemas de Expansión de la PC.

Un factor importante en una máquina es la disponibilidad de expansión del bus estándar de la PC. El bus de expansión no únicamente proporciona el principal medio de adaptación de la máquina para un rango particular de aplicaciones si no también proporciona un medio de adhesión para un ancho rango de sistemas electrónicos externos. Por lo cual, un gran de número de manufacturas tienen reconocido este hęcho y desarrollan productos de expansión específicamente para control, adquisición de datos y aplicaciones de instrumentación.

Una expansión de bus deberá proporcionar acceso completo a un amplio rango de las señales del sistema de bus. Estas señales pueden ser divididas en las siguientes categorías:

1.- Bus de líneas de direcciones.
2.- Bus de líneas de datos.
3.- Señales de control de lectura y escritura.
4.- Señales de petición de interrupción.
5.- Señales de conocimiento de DMA y de petición de DMA.
6.- Señales de control en general.
7.- Señales de reloj.

8.- Líneas de alimentación.

La expansión se realiza por medio de tarjetas conectadas al bus de la PC (usando conectores que se adaptan a la tarjeta principal), como lo muestra la figura 1-1, o por medio de un sistema de cable plano (como el *STE* o *VME*).

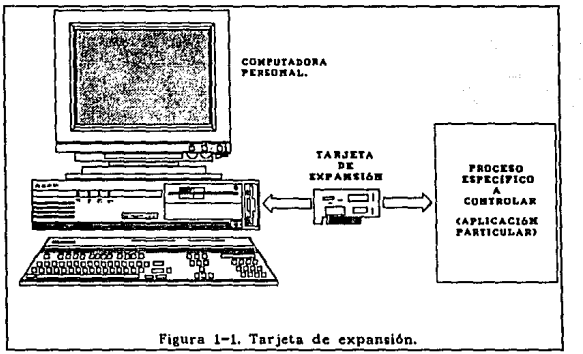


Figura 1-1. Tarjeta de expansión.

El método para expander el bus de la PC es simple, se proporciona un número de puntos de acceso al sistema de la tarjeta principal. Este acercamiento fué seguido por la IBM (y las incontables manufactureras compatibles) como un medio de conexión escencial de sistemas electrónicos periféricos (así como manejadores de disco y controladores de video) vía controladores a tarjetas adaptadas.

Este método de conexión puede ser empleado para aplicaciones especializadas así como

adquisición de datos analógicos, bus de control de IEEE-488, etc.

Dos estándares fundamentales son empleados en el esquema del bus de expansión de la PC convencional. El original, y más extensamente usado, se basa en el *Industry Standard Architecture* (*ISA*) pero es más comúnmente conocido simplemente como el Bus de Expansión de la PC. Este esquema utiliza entre uno y dos conectores directos.

El primer conector (62 líneas) proporciona acceso al bus de 8 bits y la mayoría de las señales de control y líneas de alimentación, mientras que el segundo conector (36 líneas) da acceso al bus de datos restante, junto con algunas señales de control adicionales. Aplicaciones que requieren únicamente 8 bits de datos y el estandar de señales de control pueden usar únicamente el primer conector. En aplicaciones que requieren acceso a 16 bits (no disponible en las máquinas PC antiguas y PC-XT) pueden ser usados ambos conectores.

El segundo método surge con la llegada de la PS/2. Este estandar, conocido como *Micro Channel Architecture* (*MCA*), proporciona acceso a 16 bits del bus de datos en la IBM PS/2 Modelo 50 y 60, y acceso a 32 bits del bus de datos en el Modelo 80 (apto con el CPU 80386).

Una importante ventaja del *MCA* es que permite transferir datos a una razón más rápida que un ISA. Aunque es bueno el incrementar la razón de transferencia de datos, quizá no es tan importante en muchas aplicaciones como cuidar las diferencias de una máquina a otra. Un ejemplo de cambio brusco se tiene cuando una AT estándar es comparada con una PS/2 Modelo 50; puede ser esperado un incremento alrededor del 25 porciento con la memoria convencional y un 100 porciento o más con el DMA en la razón de transferencia de datos.

Algunos ejemplos de tarjetas diseñadas para diferentes propósitos y que apoyan a la computadora en tareas especiales son las siguientes[1]:

-Tarjetas de Entrada/Salida analógicas.
-Tarjetas de Entrada/Salida digitales compatibles con niveles TTL.
-Tarjetas de Entrada/Salida digitales opto-aisladas.
-Tarjetas con líneas de salida adaptadas con relevadores.
-Tarjetas con líneas de salida adaptadas con relevadores para el control de potencia de a.c./c.d. o con triacs para el control de potencia de a.c.
-Tarjetas de expansión de memoria.
-Tarjetas manejadoras de disco duro.
-Programadores de *EPROM*.
-Tarjetas de interface para *IEEE-488*.
-Tarjetas para redes.
-Tarjetas Modem (para permitir la transferencia de información a través de la línea telefónica convencional).
-Tarjetas prototipo.
-Tarjetas para comunicación serial con los puertos *RS-232*, *RS-422*, *RS-423* o *RS-485*.
-Controladores de motores de pasos.
-Tarjetas de Entrada/Salida multi-función (proporciona líneas digitales y analógicas).
-Tarjetas de adquisición de datos de alta velocidad.
-Tarjetas de expansión del bus.
-Manejadores de discos basados en *RAM/EPROM* de estado sólido.
-Tarjetas de instrumentación especializada (multímetros digitales, contadores digitales de frecuencia,etc.).

Los sistemas construidos deben ser capaces de seleccionar de entre un gran rango de señales, para acondicionar tarjetas que proporcionan los circuitos necesarios de interfase para un amplio número de sensores populares y dispositivos de salida. Esto hace posible la construcción de un sistema de control de procesos basado en la PC seleccionando adecuadamente

módulos (tarjetas de expansión). Solo cuando se tiene una aplicación muy especializada, es necesario construir una tarjeta de Entrada/Salida dedicada y/o tarjetas para condicionar señales externas.

El propósito de esta tesis es el diseño y construcción de una tarjeta prototipo de Entrada/Salida multifunción (proporciona líneas digitales y analógicas) de próposito general, para el control de procesos industriales o de laboratorio (como son el despliegue de imágenes ultrasónicas, el proceso para medir el dioxido de carbono en agua salada, etc). El diagrama de bloques de la tarjeta se muestra en la figura 1-2.

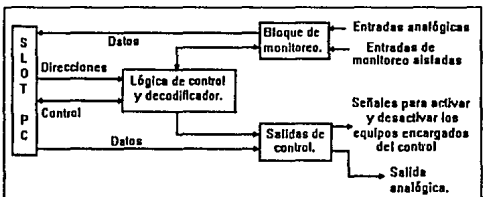


Figura 1-2. Diagrama de bloques de la tarjeta.

La tarjeta de propósito general, diseñada y construida, cuenta con las siguientes características:

a) Convertidor A/D de 12 bits de resolución con un tiempo máximo de adquisición de 25 μs y 4 entradas analógicas.

b) Salidas digitales para la activación de circuitos de potencia o relevadores para proporcionar señales adecuadas para los equipos encargados del control del proceso.

c) Convertidor D/A para la reproducción de señales analógicas (recibidas por el convertidor A/D, o bien, generadas por la computadora).

d) Entrada digitales controladas por optoaisladores para el monitoreo de variables del proceso a controlar.

Además cuenta con toda la circuitería necesaria para interactuar directamente con uno de los slots internos de expansión de la *PC* ocupando las direccines reservadas para la tarjeta prototipo ($300-$31f).

1.2 Panorama de Tesis.

Este trabajo de tesis consta de 6 capitulos, de los cuales los primeros presentan conceptos teóricos acerca de la arquitectura de la PC y el desarrollo que ha tenido el control de procesos vía computadora, en los capitulos restantes se describe el proceso de diseño y construción de la tarjeta controladora de procesos industriales.

El capitulo I presenta una introducción al sistema de expansión de la computadora IBM PC/XT/AT, mencionando las tarjetas que existen en el mercado, así como una breve descripción de la tarjeta diseñada y construida en estes trabajo de tesis.

En el capitulo II se muestra la evolución que ha tenido el control de procesos industriales

por medio de una computadora, mostrando diferentes configuraciones para el mismo.

El capítulo III describe el *hardware* de la tarjeta diseñada, (interfase de la IBM PC/XT/AT con tarjetas de expansión, conversión A/D y D/A, optoaisladores, relevadores, etc.).

El capítulo IV presenta el análisis del programa de adquisición y control, el cual esta hecho en el lenguaje de alto nivel Turbo Pascal 6.0, utilizando el soporte para la programación orientada a objetos, proporcionada por su utilería llamada Turbo Visión.

En el capítulo V se presenta una descripción de la aplicación implementada, describiendo las rutinas y el *hardware* necesario para el control del proceso de la aplicación específica de la tarjeta.

En el capítulo VI se presentan las conclusiones (y la evaluación de las mismas) a las que se llegaron al término de este trabajo.

Finalmente, en el apéndice A, se muestra el listado del programa (principal y unidades) empleado para probar el adecuado funcionamiento de la tarjeta y realizar la implementación de la aplicación presentada.

II ANTECEDENTES.

2.1 Control por computadora.

La utilización de computadoras digitales en el control automático de procesos industriales ha aumentado espectacularmente desde el comienzo de los años setenta. Estas computadoras no sólo permiten implantar sistemas de control de mayores prestaciones que las obtenidas con controladores analógicos, a un menor precio sino que, además pueden realizarse muy diversas tareas de tratamiento de datos y supervisión.

Entre las primeras pueden incluirse la recepción y filtrado de datos, selección de subconjuntos significativos de información, procesamiento para efectuar cálculos (comprobaciones, tendencias, promedios), presentación gráfica, almacenamiento en memoria auxiliar, etc. Estas funciones son de gran interés para el posterior análisis del proceso de fabricación o transformación, posibilitando su mejora.

Las funciones de supervisión incluyen la comprobación de límites de tolerancia de variables del proceso con activación, si procede, de alarmas y las asistencias que facilitan las acciones manuales del operador sobre el proceso.

Por lo que respecta a las funciones específicas de control, hay que señalar que, en estas tareas, a diferencia de las anteriormente mencionadas, la computadora no sólo suministra mensajes de información, sino que, además, genera automáticamente señales de control sobre el proceso con objeto de que el funcionamiento de éste sea el requerido. Al contrario que en las tareas de tratamiento de datos y supervisión, en este caso, el flujo de información proceso-computadora es normalmente bidireccional.

Los primeros estudios de viabilidad del empleo de computadoras digitales en el control de procesos industriales datan de los años cincuenta. En 1959 un grupo de ingenieros de las empresas *Thomson Ramo Woolridge y Texaco* diseñaron un sistema de control por computadora de una unidad de polimerización, en la cual se controlaban 26 flujos, 72 temperaturas, 3 presiones y 3 composiciones. Se trataba de determinar una distribución óptima de las alimentaciones de 5 reactores, minimizando la presión de los reactores, controlando los suministros de agua caliente a partir de medidas de la actividad de los catalizadores y determinando la circulación óptima. Esta optimización motivó el interés de las industrias de procesos, de los fabricantes de computadoras, y de grupos de investigación. Se efectuaron entonces otras realizaciones en las cuales las computadoras digitales se empleaban para supervisar el funcionamiento de controladores analógicos con objeto de mejorar las condiciones de trabajo, optimizar la producción, y realizar diversas tareas de tratamiento de datos y supervisión como las mencionada anteriormente. Aunque la tecnología de las computadoras de aquella época suponía una importante limitación, en 1962 existían ya 159 aplicaciones fundamentales en industrias del acero, químicas, y de generación de energía eléctrica. En 1962 la firma inglesa *Imperial Chemical Industries* introdujo en sus instalaciones un nuevo avance consistente en sustituir los controladores analógicos de un proceso por una computadora digital que realizaba sus mismas funciones actuando directamente sobre el proceso. La computadora medía 224 variables y controlaba directamente 129 válvulas. En años posteriores el número de estas aplicaciones fué aumentando, y se acuño la denominación de -Control Digital Directo- para

hacer énfasis en que era la computadora la que actuaba directamente sobre el proceso.

A finales de los años sesenta y comienzos de los setenta las minicomputadoras encuentran una importante acogida en aplicaciones industriales. Fruto de ello es que el número de computadoras de procesos pasa a ser de 5000 en 1970 a unos 50000 en 1975. Se utilizan entonces mayoritariamente minicomputadoras de 16 bits de palabra, con capacidad de memoria principal desde 8KB hasta 124KB y provistos de unidad de disco flexible para almacenamiento secundario.

La aparición en 1972 del microprocesador suministra un impulso decisivo al control de la computadora, haciendo rentables numerosas aplicaciones. Los avances con microelectrónica de los años ochenta con la tecnología de los circuitos de gran escala de integración acentúan esta tendencia, permitiendo que incluso pueda ser rentable la utilización de microprocesadores como elementos de control de un único bucle. Los avances en el desarrollo de recursos de programación para tiempo real han contribuido también en forma importante a disminuir el costo del proyecto e implantación de los sistemas de control por computadora, aumentando su modificabilidad y seguridad de funcionamiento.

En la figura 2-1 tomada de *Astrõm y Wittenmark* se muestra la evolución del número de computadoras, utilizadas para el control de procesos industriales, desde el comienzo de los sesenta[2].

Según estimaciones de la sociedad norteamericana *Venture Development Corporation* el mercado de las computadoras destinadas a las aplicaciones de medida de control y supervisión experimentaba en 1983 un crecimiento anual superior al 34% en unidades y de cerca del 16% en valor.

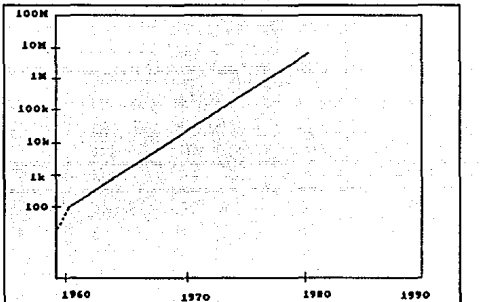


Figura 2-1. Crecimiento del número de computadoras en control de procesos industriales.

En los últimos años han experimentado un importante auge los sistemas de control descentralizado y jerarquizado con implantación mediante redes de computadoras.

La computadora constituye en la actualidad otro campo de aplicación de enormes posibilidades. En particular, el diseño de sistemas de control de robots presenta importantes problemas de estabilidad, tiempo de respuesta y precisión que captan la atención de un número creciente de grupos de investigación, proporcionando un impulso considerable en el desarrollo de la Teoría de Control.

Existe una importante cantidad de computadoras en tareas de control secuencial y lógico. Estas tareas suelen efectuarlas microcomputadoras, que actúan mediante un programa que realiza las funciones de un circuito lógico secuencial. En este caso, tanto los actuadores como los

captadores de señal del proceso son dispositivos de dos estados, tales como válvulas abiertas o cerradas, etc. Las ventajas que conduce el empleo de la computadora en lugar de un circuito secuencial son las derivadas del empleo de lógica programada sobre la cableada es decir, la flexibilidad en el cambio de estrategia de control, menor número de averías, etc. Los inconvenientes son la mayor influencia de una avería y los problemas de mantenimiento y modificación por personal no especializado. Hay que señalar que, en este tipo de aplicaciones, está muy extendido el empleo de autómatas programables, a los que se ha definido como instrumentos electrónicos programables por personal no especializado en informática destinados a cumplir, en ambiente industrial y en tiempo real, funciones escencialmente de automatismos lógicos, combinacionales y secuenciales. No obstante, existen autómatas programables de la gama alta que incorporan la posibilidad de cálculos relativamente importantes. En cualquier caso, la mayor parte de los autómatas programables pueden ser considerados como computadoras especializadas.

En la figura 2-2 se muestra un bucle típico de control analógico en el cual el controlador se diseña para que la salida y(t) siga a la referencia r(t), aunque se presenten perturbaciones externas sobre el proceso w(t) y errores, representados por la acción de v(t), en las medidas de los sensores.

En esta aplicaciones, las variables asociadas al proceso suelen evolucionar de forma continua en el tiempo. Por consiguiente, la utilización de un controlador digital en sustitución del analógico hace necesario el empleo de convertidores analógicos-digitales para suministrar señales al controlador. Asimismo, si los actuadores son también analógicos será necesario el empleo de convertidores digital-analógico. De esta forma se llega al esquema de control digital de la figura 2-3. No obstante, hay que señalar que la digitalización puede extenderse a los propios captadores y actuadores eliminando la necesidad de los convertidores. Por otra parte, en muchos casos no existen señales de referencia analógica, sino que éstas se introducen como consignas en el propio

controlador digital. Por último, se nota que la actuación de la computadora esta sincronizada con el proceso. Para ello es necesario el empleo de un reloj en tiempo real. Típicamente el convertidor analógico-digital suministra un número a la computadora cada vez que llega un pulso de reloj. Los números se suministran en un período T (período de muestreo). No obstante en la práctica, existen sistemas de control digital con períodos de muestreo variable y distintas señales muestreadas con diferentes períodos de muestreo. La señal de entrada a la computadora puede representarse mediante una variable e(kT) que toma valores a intervalos discretos de tiempo k=0,1..... La ejecución del algoritmo de control da como resultado una secuencia de números representada mediante la variable u(kT) que se aplican al proceso, a través de un convertidor digital-analógica, con el mismo período constante T. La señal analógica suministra al actuador se representa mediante la variable u(t) que mantiene un valor constante u(t)=u(kT) para $kT<t<(k+1)T$ [3].

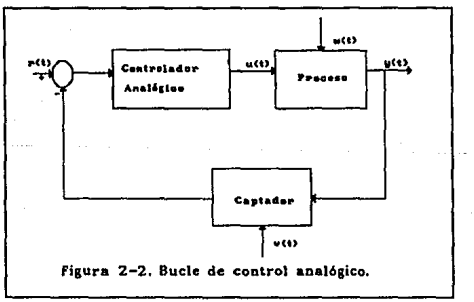


Figura 2-2. Bucle de control analógico.

El empleo de la computadora como elemento de control ofrece, con respecto al control analógico, la ventaja de la posibilidad de cambio en la estrategia de control con la simple

modificación del programa, la menor limitación en la complejidad de esta estrategia para conseguir mejores prestaciones, la posibilidad de configurar esquemas de control distribuido con interconexión entre controladores, y la mayor precisión a igualdad de costo. La computadora puede actuar controlando varios bucles simultáneamente, como se ilustra en la figura 2-4 y, dependiendo de la potencia, puede realizar a la vez las funciones de tratamiento de datos y supervisión.

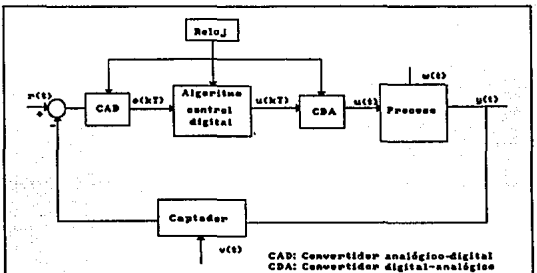


Figura 2-3. Bucle de control digital.

En la figura 2-5 se muestra otro esquema de control que incluye una computadora digital. Es el denominado control analógico-digital o control de supervisión . En este esquema la computadora actua asignando valores de consigna a bucles de control analógico, con objeto de supervisar y controlar acciones de los distintos controladores analógicos locales, según criterios de eficiencia local. En la figura 2-6 se representa el control digital de un bucle de regulación analógica de temperatura.

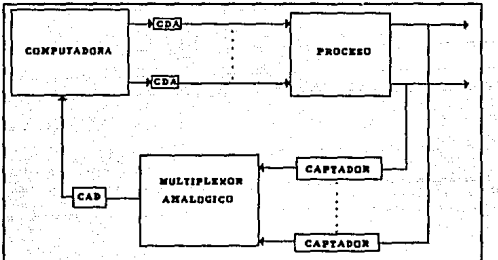


Figura 2-4. Computadora Digital controlando varios bucles de control.

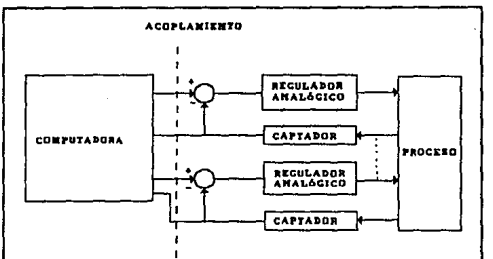


Figura 2-5. Control Analógico-Digital.

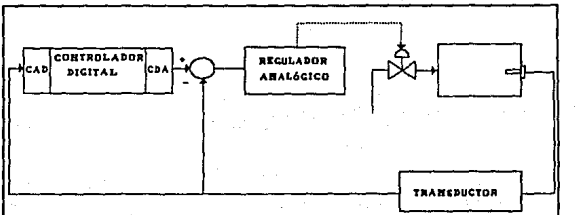


Figura 2-6. Control Analógico-Digital de temperatura.

El control analógico-digital resulta fácil de implantar sobre un proceso que disponga de un conjunto de reguladores analógicos convencionales previamente instalado. De hecho, los primeros sistemas de control analógico-digital se implantaron antes que los sistemas de control digital directo, los cuales recibieron esta denominación por actuar directamente sobre el proceso eliminando los controladores analógicos. Un control analógico-digital ahorra tiempo de cálculo a la computadora, lo cual era especialmente importante en las computadoras primitivas, ya que los cambios de referencia de los reguladores analógicos pueden realizarse a intervalos de tiempo más grandes que los períodos de muestreo de regulación digital directa. Asi mismo puede ofrecer más seguridad de funcionamiento ya que en caso de avería de la computadora, el regulador analógico mantendría la última consigna. Sin embargo, tiene los inconvenientes derivados del empleo de reguladores analógicos, tales como la impresión, constantes limitadas, menor flexibilidad, y la limitación de la complejidad de la estrategia de control a un precio razonable.

En el control analógico-digital existen implícitamente dos niveles de control, con la computadora, en el nivel superior, actuando como coordinador. Si los controladores del nivel

inferior son también digitales se tiene una estructura de control digital jerarquizado como la que se ilustra en la figura 2-7. Asimismo, el esquema puede generalizarse para incluir múltiples niveles, cada uno de lo cuales tiene una función asignada. En la figura 2-8 se muestra un esquema de control jerarquizado de varios procesos de una misma instalación. En el primer nivel se dispone un controlador para cada uno de los procesos. Los bucles de control son supervisados en un segundo nivel. En el tercer nivel se considera la coordinación entre los procesos y el control de la producción. Finalmente, se considera un cuarto nivel de gestión en el cual se podrían tener en cuenta criterios, tales como consideraciones de mercado, materias primas, personal, etc.

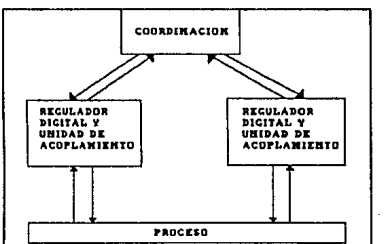


Figura 2-7. Control Digital a dos niveles.

Hay que señalar que el término control jerarquizado no implica que existan múltiples procesadores. En la práctica es frecuente que una sola computadora ejecute, de forma concurrente, las tareas de dos o más niveles.

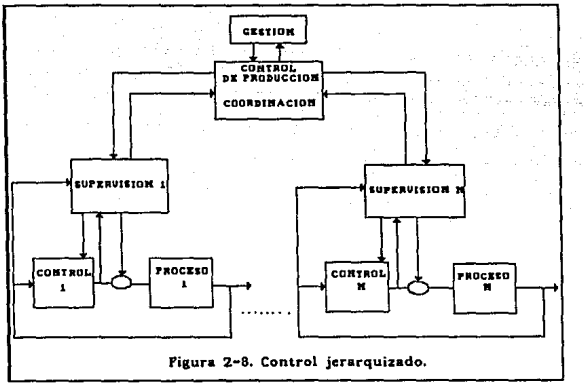


Figura 2-8. Control jerarquizado.

III DESCRIPCIÓN GENERAL DEL SISTEMA.

3.1 Sistemas de microcomputadoras.

Los principales elementos de un sistema de microcomputadora consiste de una Unidad de Procesos Central (*CPU*), Memoria de lectura y escritura (*RAM*), Memoria de solo lectura (*ROM*), junto con dispositivos de entrada y salida (*I/O*). Estos elementos están conectados por un sistema de bus, el cual lleva señales de datos, de control, de direcciones como se muestra en la figura 3-1.

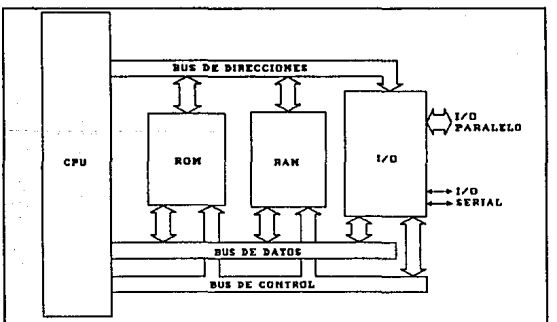


Figura 3-1. Elementos de un sistema básico de microcomputadora.

El *CPU* es el microprocesador (como el 8088,8086,80286), mientras que las memorias de lectura/escritura y de sólo lectura son implementadas usando dispositivos semiconductores de memoria (*RAM* y *ROM* respectivamente). Los semiconductores *ROM* proporcionan almacenamiento no volátil para el código del sistema operativo (El código permanece intacto cuando la fuente de alimentación es desconectada), mientras que el semiconductor *RAM* proporciona almacenamiento del resto del código del sistema operativo, programas de aplicaciones y transición de datos. Es importante notar que esta memoria es volátil y cualquier programa o dato guardado se pierde cuando se desconecta la fuente de alimentación.

El sistema operativo es una colección de programas de utilidades los cuales proporcionan un ambiente de software de aplicaciones que pueden interactuar con el hardware. El sistema operativo proporciona al usuario las principales tareas de la computadora, así como dar formato a los disco flexibles, copiando discos, etc. También proporciona en orden las principales interacción con el usuario (vía el teclado con comandos y en la pantalla con mensajes).

Parte de la memoria *RAM* es reservada para que sea usada por el sistema operativo y para guardar los textos y gráficas por desplegar (como es apropiado). Para optimizar el uso de la memoria disponible el más moderno sistema operativo, que emplea técnicas de memoria magnética, coloca en memoria programas en transición y libera la memoria cuando el programa es terminado. Un tipo de programa especial (llamado programa residente) puede sin embargo quedar en memoria de inmediata ejecución y más tarde guardarse cuando otra aplicación se este ejecutando.

Los dispositivos de entrada y salida (*I/O*) proporcionan la principal conexión con el hardware externo como son: el teclado, los controladores de disco, la pantalla. Los dispositivo de entrada y salida son manejados por un número de dispositivos especializados *VLSI*, cada uno dedicado a una función especial (como el controlador de disco, el controlador de gráficos, etc.).

Estos dispositivos son complejos y programables (requieren de configuración mediante software durante la inicialización del sistema).

Los elementos dentro de un sistema de microcomputadora como se muestra en la figura 3-2 (*CPU*, *RAM*, *ROM*, Dispositivos de Entrada/Salida) son conectados por tres distintos sistemas de bus:

-Bus de direcciones. En él se llevan las señales de direcciones.

-Bus de datos. En él se llevan las señales de datos.

-Bus de control. En él se llevan las señales de control.

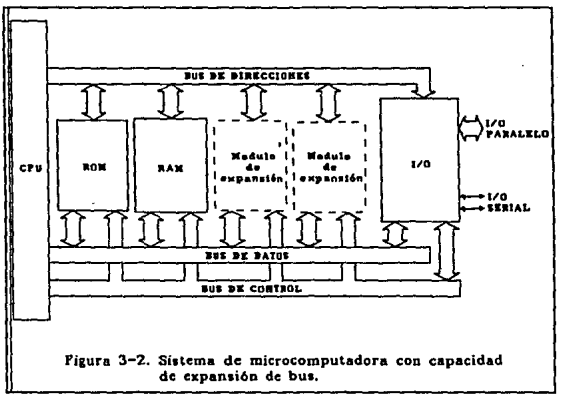


Figura 3-2. Sistema de microcomputadora con capacidad de expansión de bus.

Las señales presentadas en las líneas del bus son digitales y tienen únicamente dos estados,

uno lógico (o alto) y el cero lógico (o bajo). Las direcciones y los datos son codificados en binario, con el bit más significativo (*MSB*) siendo la línea de dirección o dato más alta y el bit menos significativo en la línea de dirección o dato más baja (etiquetado con A0 y D0 respectivamente).

Las líneas del bus se consideran como el cuarto elemento del sistema. Los datos pasan en las líneas del bus en grupos de 8 ó 16 bits. Un grupo de 8 datos es comúnmente conocido como *byte* mientras que un grupo de datos de 16 es referido generalmente como *word* (palabra).

Como un ejemplo, se asume que el estado de 8 datos de las líneas del bus en un sistema, para un instante en particular de tiempo, es como se muestra:

	(MSB)							(LSB)
BUS DE DATOS :	D7	D6	D5	D4	D3	D2	D1	D0
VALOR :	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	(=128)	(=64)	(=32)	(=16)	(=8)	(=4)	(=2)	(=1)
ESTADO LOGICO :	1	0	1	0	0	1	1	1
HEX. :			A			7		

El valor binario es 10100111 y su valor decimal (se encuentra por la suma del decimal equivalente donde este presente un '1' de acuerdo a la posición) es 167. Es más conveniente expresarlo en sistema hexadecimal (base 16). El valor del byte en hexadecimal es A7 (se agrupan los bits en grupos de 4 llamados *nibbles*, y cada *nibble* se convierte a su equivalente carácter hexadecimal); el valor del byte en hexadecimal se puede expresar de la siguiente manera: A7H, &HA7 o $A7_{16}$.

El bus de datos contiene 8 (ó 16) líneas separadas y etiquetadas como D0 a D7 (o D0 a D16), el bus de direcciones en la PC, PC-XT, PC-AT y compatibles, contiene 20 líneas etiquetadas

etiquetadas de A0-A19.

El sistema mostrado en la figura 3-1 puede ser expandido haciendo que los tres sistemas de bus sean accesibles a un números de módulos de expansión como se muestra en la figura 3-2. Estos módulos (que varían de acuerdo a las dimensiones de la tarjeta del circuito impreso) pueden tomar la forma de memoria adicional (memoria expandida), dispositivos de entrada y salida, o pueden proveer funciones adicionales asociadas con gráficos o control de disco. Las tarjetas de expansión se refieren a tarjetas de opción o tarjetas de adaptación que proporcionan una configuración al sistema básico de la microcomputadora para una aplicación en particular.

Operación del microprocesador.

La mayoría de las operaciones del microprocesador implican el movimiento de datos. Claro que el código del programa (el conjunto de instrucciones guardados en la memoria *ROM* y *RAM*), debe ser almacenado antes de ser ejecutado. El microprocesador de esta manera mejora la secuencia continua de los ciclos de recuperación, decodificación y ejecución de las instrucciones. El ciclo de recuperación y decodificación del código de instrucción (valor del operando o dato) de la memoria comprende una operación de lectura mientras que el acto de ejecución de datos del microprocesador a una localidad de memoria comprende una operación de escritura.

El microprocesador determina la fuente del dato (cuando esta siendo leído) y el destino del dato (cuando esta siendo escrito) en un lugar de una dirección única indicada por el bus de direcciones. La dirección en la cual el dato ha sido ubicado (durante la operación de escritura) o en la cual ha sido recuperado y decodificado (durante la operación de lectura) puede constituir, uno o el otro, parte de la memoria (*RAM* o *ROM*) del sistema; también puede ser considerado como Entrada o Salida (*I/O*).

Desde que el bus de datos ha sido conectado a los diferentes dispositivos *VLSI*, un esencial requerimiento es que sus salidas sean capaces de aislarse del bus cuando sea necesario (con la memoria *RAM* o *ROM*). Los dispositivos *VLSI* son habilitados con una entrada llamada selección o habilitador, las que son manejadas por las direcciones de la lógica de decodificación. Esta lógica asegura que la *ROM*, la *RAM* y los dispositivos de entrada/salida no utilicen el bus de datos a la vez.

Las entradas de la lógica de decodificación son derivadas de una o más líneas del bus de direcciones. El decodificador de direcciones divide la memoria disponible en bloques, los cuales corresponden a uno o más dispositivos *VLSI*. Entonces mientras el microprocesador esta leyendo o escribiendo a la memoria *RAM* por ejemplo , las direcciones de la lógica de decodificación aseguran que únicamente la *RAM* sea seleccionada, mientras que la *ROM* y los dispositivos de entrada/salida permanecen aislados del bus de datos (alta impedancia).

Arquitectura del sistema PC.

La PC contiene tres unidades: Unidad del sistema, teclado y monitor. A su vez la unidad del sistema contiene tres partes que son: La tarjeta del sistema, la fuente de alimentación, y las unidades de disco.

La tarjeta original del sistema de la IBM PC emplea aproximadamente 100 circuitos integrados incluyendo el *CPU* 8088 (U3), un controlador de interrupciones 8259A (U2), un coprocesador 8087 (U4), un controlador de bus 8288 (U6), un generador de reloj 8284A (U11) y *timer/contador* (U34), un controlador de *DMA* 8237A y una interfase paralela 8255 (U36) junto con un *host* de lógica discreta (incluye en el bus dispositivos de memoria, de tres estados - bidireccionales- y seguidores)[4].

La IBM PC-AT estándar emplea un *CPU* 80286 (U74) y un coprocesador 80287 (U76). Dos controladores *DMA* (U111 y U122) adaptados, un *timer* programable (U103), un generador de reloj 8284A, dos controladores de interrupción 8259A (U114 y U125) y mapa de memorias 74LS612n (U124).

Muchas de la microcomputadoras de hoy en día XT y AT son basadas en un número pequeño de dispositivos (muchos de los cuales utilizan tecnología superficial). Esto no solo favorece la confianza en lo instrumental, si no también en la reducción de costos de manufactura.

El número total de dispositivos en las modernas microcomputadoras XT y AT compatibles ha sido reducido significativamente por la integración de varias de las funciones asociadas con el original chip de la PC con un singular dispositivos *VLSI* o con el propio *CPU*. Como un ejemplo, el chip controlador 82C100 XT proporciona por lo menos 6 funciones del original chip XT y efectivamente reemplaza los siguientes dispositivos:

-8237 Controlador *DMA*.
-8253 *Contador/Timer*.
-8255 Interfase Paralela.
-8259 Controlador de interrupciones.
-8284 Generador de Reloj.
-8288 Controlador de bus.

3.2 Bus de expansión de la PC.

El bus de expansión de la PC esta basado en un número de slots de expansión, cada uno es

es adaptado en forma directa con un conector de 62 líneas de los tres buses (figura 3-3), junto con un conector opcional para las 36 líneas (figura 3-4) restantes. La tarjeta de expansión puede ser diseñada conectándola únicamente con el conector de 62 líneas o puede hacer pareja con el conector de 32 líneas. Antes solo se tenía adaptado el conector de 62 líneas en las máquinas antiguas (que tenían el bus de 8 bits); las tarjetas diseñadas con este conector son conocidas como *tarjetas de expansión de 8 bits* o *Tarjetas de expansión PC/XT*. La máquina AT, sin embargo proporciona acceso completo a los 16 bits del bus de datos junto con las señales de control, por lo que se requiere del conector de 36 líneas. Las tarjetas que son diseñadas para hacer uso de los dos conectores son conocidas generalmente como *Tarjetas de expansión de 16 bits* o *Tarjetas de expansión AT*.

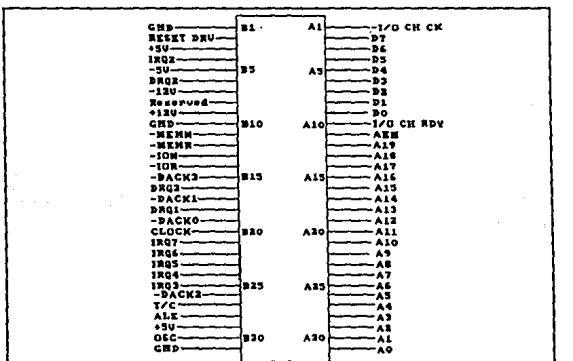


Figura 3-3. Bus del canal de 62 terminales de entrada/salida de la IBM PC, incluye 20 líneas de direcciones, 8 líneas bidireccionales de datos, señales de control y alimentación.

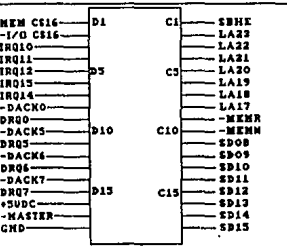


Figura 3-4. Bus de direcciones de entrada/salida auxiliar de 36 terminales de la IBM PC. Incluye 7 líneas de direcciones más significativos para 16 MB de espacio de direccionamiento, 8 líneas de datos del byte alto, señales de control y alimentación.

La original PC fué adaptada con 5 slots (espacio de separación aproximadamente de 25 mm). La estándar XT proporciona tres slots de más para tener un total de 8 (espacio aproximadamente de 19 mm de separación). Algunas tarjetas, particularmente las de los disco duros, requieren el doble de espacio o sea que ocupan dos slots en la XT. Esto es desafortunado, particularmente en donde el número de slots libres es escaso.

Todos los slots de las XT proporcionan las mismas señales con una notable excepción; el slot más cercano a la fuente de alimentación fué empleado particularmente por la configuración de la IBM (la *IBM 3270 PC*) aceptando un adaptador teclado/timer. Esta particular configuración emplea una señal dedicada a la "selección de tarjeta" (B8 en el conector) la cual es requerida por la tarjeta principal. Otra tarjeta que opera en esta posición es la tarjeta de comunicaciones asíncrona de la IBM 3270.

Como la XT, la estándar AT también proporciona 8 slots de expansión. Seis de estos slots son adaptados con los dos conectores (62 líneas y 36 líneas) mientras que los slots que están en la posición 1 y 7 tienen el conector de 62 líneas; los slots en las posiciones 1 y 7 están diseñados para aceptar tarjetas de expansión de 8 bits, que puede usar el máximo disponible de altura a todo lo largo.

Finalmente se deberá notar que las tarjetas diseñadas para la AT (están específicamente diseñadas para tomar ventaja de la disponibilidad del bus de datos de 16 bits) ofrecen una considerable ventaja en la velocidad sobre las que están basadas en el bus de datos de 8 bits proporcionados por el conector de expansión de la original XT. En algunas aplicaciones , esta ventaja en la velocidad puede ser crítica.

Tarjetas de Expansión de la PC.

Las tarjetas de expansión del sistema de la PC tienden ha ser pequeñas en sus contornos y dimensiones (ver figura 3-5). Sin embargo, el máximo disponible del adaptador y del conector del equipo de la PC (y PS/2) son las siguientes[5]:

ESTANDAR	BUS	ALTURA		LARGO		ANCHO	
		plg	mm	plg	mm	plg	mm
ISA	XT	4.2	107	13.3	3.35	0.5	12.7
ISA	AT	4.8	122	13.2	3.35	0.5	12.7
MCA	PS/2	3.8	96	13.2	3.35	0.5	12.7

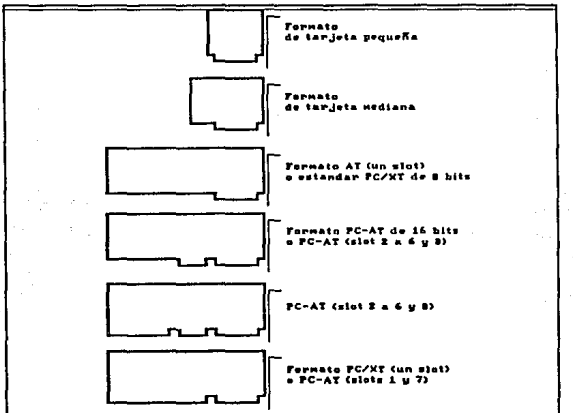


Figura 3-5. Contornos de diferentes tipos de tarjetas de expansión.

Es importante notar que aunque la XT-286 esta basada en una tarjeta principal AT, sus expansiones están sujetas a las restricciones de altura generales impuestas en las tarjetas XT (4.2 pulgadas máximo).

Otra dificultad es que algunas tarjetas XT pueden fallar al operar en el equipo de AT debido a la interrupción del circuito DMA.

El conector del bus de expansión de 62 líneas.

El conector del bus de expansión de 62 líneas es adaptado a la tarjeta principal. Un lado del

conector es referido como A (líneas enumeradas de A1 a A31) mientras el otro lado es referido como B (líneas enumeradas de B0 a B31). El bus de datos y direcciones son agrupados en el lado A del conector mientras el bus de control y líneas de alimentación ocupan el lado B (ver figura 3-6).

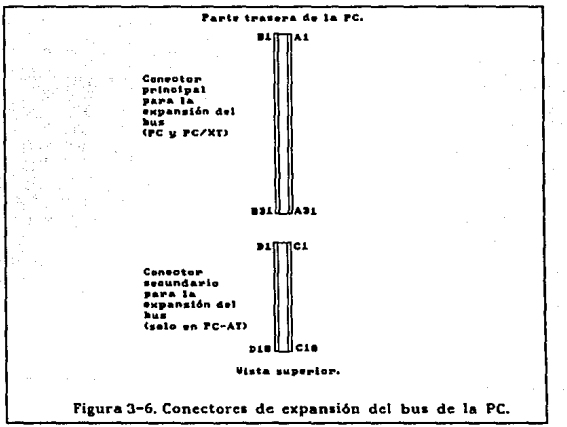


Figura 3-6. Conectores de expansión del bus de la PC.

Es importante estar consciente que algunas PC antiguas no utilizaban las letras A y B en el conector de expansión, distinguiendo los dos lados del conector enumerando con números nones del 1 al 61 de un lado y del otro lado con números pares del 2 al 62.

La siguiente tabla describe cada una de las señales presentes en el conector bus de expansión[6].

Terminal no.	Abreviatura	Dirección	Señal	Función
A1	IOCHK	I	Checa canal I/O	Nivel bajo indica error de paridad en la memoria o en un dispositivo de I/O
A2	D7	I/O	Línea de Dato 7	Línea de Bus de Datos
A3	D6	I/O	Línea de Dato 6	Línea de Bus de Datos
A4	D5	I/O	Línea de Dato 5	Línea de Bus de Datos
A5	D4	I/O	Línea de Dato 4	Línea de Bus de Datos
A6	D3	I/O	Línea de Dato 3	Línea de Bus de Datos
A7	D2	I/O	Línea de Dato 2	Línea de Bus de Datos
A8	D1	I/O	Línea de Dato 1	Línea de Bus de Datos
A9	D0	I/O	Línea de Dato 0	Línea de Bus de Datos
A10	IOCHRDY	I	I/O Canal de aviso	Pulso bajo de una memoria lenta o de dispositivo de I/O le indica que no esta preparada la transferencia de datos.

A11	AEN	O	Habilitador de direcciones	Señal dada por el controlador del DMA indica que el ciclo del DMA esta en progreso, deshabilita los puertos de I/O durante la operación del DMA
A12	A19	I/O	Línea de dirección 19	Línea del Bus de direcciones
A13	A18	I/O	Línea de dirección 18	Línea del Bus de direcciones
A14	A17	I/O	Línea de dirección 17	Línea del Bus de direcciones
A15	A16	I/O	Línea de dirección 16	Línea del Bus de direcciones
A16	A15	I/O	Línea de dirección 15	Línea del Bus de direcciones
A17	A14	I/O	Línea de dirección 14	Línea del Bus de direcciones
A18	A13	I/O	Línea de dirección 13	Línea del Bus de direcciones

A19	A12	I/O	Línea de dirección 12	Línea del Bus de direcciones
A20	A11	I/O	Línea de dirección 11	Línea del Bus de direcciones
A21	A10	I/O	Línea de dirección 10	Línea del Bus de direcciones
A22	A9	I/O	Línea de dirección 9	Línea del Bus de direcciones
A23	A8	I/O	Línea de dirección 8	Línea del Bus de direcciones
A24	A7	I/O	Línea de dirección 7	Línea del Bus de direcciones
A25	A6	I/O	Línea de dirección 6	Línea del Bus de direcciones
A26	A5	I/O	Línea de dirección 5	Línea del Bus de direcciones
A27	A4	I/O	Línea de dirección 4	Línea del Bus de direcciones
A28	A3	I/O	Línea de dirección 3	Línea del Bus de direcciones

A29	A2	I/O	Línea de dirección 2	Línea del Bus de direcciones
A30	A1	I/O	Línea de dirección 1	Línea del Bus de direcciones
A31	A0	I/O	Línea de dirección 0	Línea del Bus de direcciones
B1	GND	n.a.	Tierra	Tierra común , 0 Volts.
B2	RESET	O	Reset	Cuando toma nivel alto inicializa toda la tarjeta.
B3	+5V	n.a.	+ 5 Volts de D.C.	Línea de alimentación.
B4	IRQ2	I	Petición de interrupción nivel 2	Petición de interrupción (alta prioridad)
B5	-5V	n.a.	-5 Volts de D.C.	Línea de alimentación.
B6	DRQ2	I	Petición de acceso directo a memoria nivel 2.	Tomando valor alto cuando requiere transferencia el DMA. La señal permanece en nivel alto hasta que la línea DACK va a nivel bajo
B7	-12V	n.a.	-12 Volts de D.C.	Línea de alimentación.

B8	OWS	I	Zero estados de espera.	Indica a el microprocesador que el ciclo de bus puede ser completado sin acertar ningún ciclo de espera.
B9	+12V	n.a.	+12 Volts de D.C.	Línea de alimentación.
B10	GND	n.a.	Tierra	Tierra común 0 Volts.
B11	MEMW	O	Escritura en la memoria.	Tomando un nivel bajo se realiza una operación de escritura en la memoria.
B12	MEMR	O	Lectura en la memoria.	Tomando un nivel bajo se realiza una operación de lectura en la memoria.
B13	IOW	O	Escritura en un dispositivo de entrada y salida.	Tomando un nivel bajo se realiza una operación de escritura en los dispositivos de entrada y salida.

B14	IOR	O	Lectura en un dispositivo de entrada y salida.	Tomando un valor bajo se realiza una operación de lectura en los dispositivos de entrada y salida.
B15	DACK3	O	Acceso directo a memoria de conocimiento nivel 3.	Tomando nivel bajo realiza una petición al DMA de acuerdo al nivel.
B16	DRQ3	I	Petición directo a memoria nivel 3.	Tomando un nivel alto cuando el DMA requiere de transferencia. La señal permanece en alto hasta que DACK vaya a nivel bajo.
B17	DACK1	O	Acceso directo a memoria de conocimiento nivel 1.	Nivel bajo es petición del DMA en el correspondiente nivel.

B18	DRQ1	I	Petición directo a memoria nivel 1.	Tomando un nivel alto cuando el DMA requiere de transferencia. la señal permanece en alto hasta que DACK vaya a nivel bajo.
B19	DACK0	O	Acceso directo a memoria de conocimiento nivel 2.	Nivel bajo es petición del DMA en el correspondiente nivel.
B20	CLK4	O	Reloj de 4.77 MHz	Reloj del CPU dividido por tres ,210 ns del período , 33 porciento del ciclo de trabajo.
B21	IRQ7	I	Petición de interrupción nivel 7.	Línea para petición de interrupción por un dispositivo de Entrada/Salida, prioridad baja.
B22	IRQ6	I	Petición de interupción nivel 6.	Línea para petición de interrupción por un dispositivo de Entrada/Salida.

B23	IRQ5	I	Petición de interupción nivel 5.	Línea para petición de interrupción por un dispositivo de Entrada/Salida.
B24	IRQ4	I	Petición de interupción nivel 4.	Línea para petición de interrupción por un dispositivo de Entrada/Salida.
B25	IRQ3	I	Petición de interupción nivel 3.	Línea para petición de interrupción por un dispositivo de Entrada/Salida.
B26	DACK2	O	Acceso directo a memoria de conocimiento nivel 2.	Nivel bajo es petición del DMA en el correspondiente nivel.
B27	TC	O	Terminal/Count	Esta línea proporciona un pulso cuando la terminal de cuenta del canal del DMA ha sido alcanzado.

B28	ALE	O	Habilitador del latch de direcciones.	Flanco de bajada indica que el latch ha sido habilitado. La señal toma un nivel alto cuando hay una transferencia del DMA.
B29	+5V	n.s.	+5 Volts de D.C.	Línea de alimentación.
B30	OSC	O	Reloj de 14.31818 MHz.	Reloj rápido con un período de 70 ns, un ciclo de trabajo del 50%.
B31	GND	n.s.	Tierra	Tierra común, 0 volts.

Características eléctricas del Bus de la PC.

Todas las líneas de las señales presentes en el conector son compatibles con TTL. En el caso de las señales de salida de la tarjeta principal, el máximo de carga impuesta por una tarjeta de expansión deberá ser limitada a no más de dos dispositivos de *(LS) TTL*.

La línea *IOCHRDY* esta disponible para interfases de memorias o dispositivos de Entrada/Salida lentas. El proceso normal genera ciclos de lectura y escritura usando cuatro ciclos de reloj por byte transferido. La frecuencia de reloj de la PC estándar es de 4.77 MHz con un período 210 ns. Así cada ciclo de lectura y escritura del procesador requiere de 840 ns en un reloj de velocidad estándar. Por otro lado la transferencia del *DMA* en los ciclos de lectura y escritura requiere de 5 ciclos de reloj (1050μs). Cuando la línea *IOCHRDY* es activada, el ciclo

de reloj del procesador es extendido para agragar un número de ciclos de reloj.

Finalmente, cuando un procesador de Entrada/Salida desea tomar control del bus, debe de activar la línea de *MASTER*. Esta señal deberá ser activada por no más de 15μs, de otra manera puede ocasionar daños al refresco de la memoria del sistema.

Diseño de tarjetas de expansión.

Varios factores se necesitan tomar en cuenta cuando se diseña una tarjeta de expansión para la PC. Esto incluye requerimientos de alimentación, la distribución de las líneas de alimentación, el desacoplamiento, y la decodificación de las direcciones. Además puede ser requerido accesar a las señales de control (como *IOCHK, IOCHRDY, DRQ, IRQ*) en caso de tarjetas con dispositivos de Entrada/Salida lentos, que requieran transferencia del *DMA* o necesiten manejar interrupciones (ver figura 3-7)[7].

Líneas de alimentación.

La disponibilidad de la alimentación para la tarjeta de expansión depende sobre el rango de suministro de la fuente de alimentación del sistema; los requerimientos de la tarjeta principal y las demandas de otras tarjetas puede ser adaptado. Cuando se diseña las tarjetas de expansión, los límites recomendados por tarjeta para cada línea de alimentación son las siguientes:

Línea de voltaje	Conexión	Máximo de corriente
+5 Volts	B3 y B29	1.5 A
-5 Volts	B5	100 mA

+12 Volts	B9	500 mA
-12 Volts	B7	100 mA

Cuando varias tarjetas son adaptadas se debe estimar y calcular la corriente y potencia total que demanda cada línea de alimentación. Esto debe ser sin exceder el rango de la fuente de suministro. En algunos casos debe ser menor que 15 Watts.

Como una guía, las siguiente tabla de datos da referencia de la potencia de las líneas de alimentación para los sistemas PC, XT, XT-286 y AT.

Sistema	PC	XT	XT-286	AT
Total de potencia (Watts) Rango de corriente máxima(Ampers)	63.5	130	157	192
+5 Volts	7	15	20	20
-5 Volts	0.3	0.3	0.3	0.3
+12 Volts	2	4.2	4.2	7.3
-12 Volts	0.25	0.25	0.25	0.3

Un sistema principal completo (incluyendo coprocesador matemático) requiere aproximadamente 4A en la línea de +5V y 2A en la línea de +12V. Una tarjeta de video EGA y dos manejadores de unidad de disco flexibles demandarán adicionalmente 2.4A y 1.8A en la línea de +5V y +12V respectivamente. Un ventilador requiere de 0.3A en la línea de

alimentación de +12V. La carga total es de 6.4A en la línea de +5V y 4.1A en la línea de +12V. Con una XT standar la fuente de suministro reserva 9.6A disponibles de la línea de +5V y únicamente 100mA disponibles de la fuente de +12V.

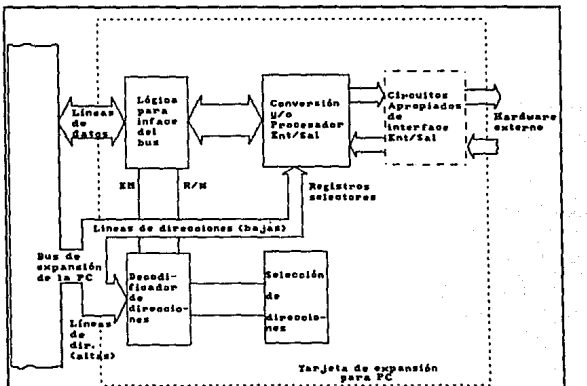


Figura 3-7. Diagrama de bloques generalizado para una tarjeta de expansión.

Diseño con el puerto paralelo Entrada/Salida.

La IBM PC es una máquina poderosa en el procesos de datos, pero se necesita la forma de comunicarse con el mundo exterior. El hardware de Entrada/Salida y el correspondiente software proporcionan la interfase de transferencia de datos entre la computadora y los dispositivos periféricos[6].

Conceptos Básicos.

Hay varias formas de iniciar y controlar la transferencia de datos:

-Programa controlando la Entrada/Salida.

-Rutina de servicios de interrupciones controlando la Entrada/Salida.

-Hardware controlando La Entrada/Salida (*DMA*).

Los puertos pueden ser conectados al microprocesador como si se accesaran direcciones de memoria. En este caso se llama Mapa de Memoria de Entrada y Salida, comúnmente conocida como Mapa de puertos, así se tiene acceso a los comandos especiales del microprocesador. Las instrucciones *IN* y *OUT* del microprocesador de Intel 8088 proporcionan la transferencia de datos para los puertos de Entrada/Salida.

La figura 3-8 muestra un diagrama de bloques de la operación de Entrada/Salida. La lógica de decodificación da el pulso de selección llamado *chip select* recibiendo señales del bus de direcciones y las señales de control *IOR* e *IOW*. El pulso de selección (*chip select*) habilita el puerto de entrada para dar acceso a los datos desde el dispositivo de entrada. El pulso de selección (*chip select*) habilita el puerto de salida para dar acceso a el dato dado por el procesador. El puerto de entrada incluye un circuito integrado tres estados para aislar la entrada del bus excepto durante el ciclo lectura. En la salida se usan dispositivos de memoria (registros) para guardar el dato hasta que la salida relativamente lenta acepte los bits de datos. En la figura 3-8 los interruptores (*DIP switch*) y los *leds* sirven como dispositivos de entrada y salida respectivamente.

Programa controlando la Transferencia de Entrada/Salida.

Hay dos tipos de programas que controlan la Entrada/Salida: el condicional e incondicional. La transferencia incondicional lleva o trae datos del puerto de Entrada/Salida sin determinar si

el puerto envía o recibe datos. Generalmente, la transferencia incondicional maneja comandos de información y estados de información. Para esta transferencia de datos , el procesador asume que el dispositivo de Entrada/Salida esta leyendo o enviando datos. Pueden ocurrir errores en la transferencia de datos si el programador no es cuidadoso. Si el microprocesador envía datos más rápidamente que la capacidad de recepción del puerto de salida, los datos se perderán. También en el puerto de entrada se pueden tener errores si los datos son leídos por el microprocesador con velocidades elevadas.

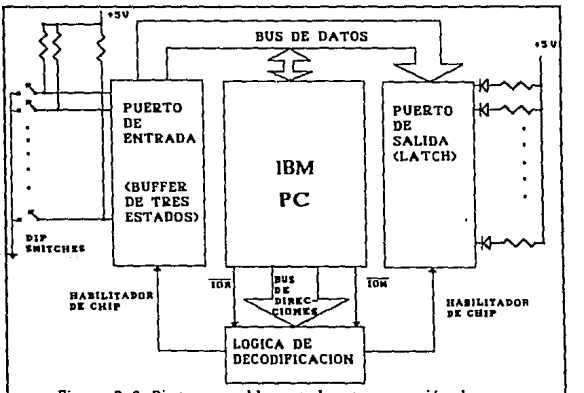


Figura 3-8. Diagrama a bloques de una operación de Entrada/Salida básico.

Para evitar estos problemas, los puertos de entrada son configurados con transferencia de datos condicional usando un protocolo de comunicación (*handshaking*). En este caso, la

transferencia incondicional de información de estado, desde el dispositivo de Entrada/Salida hacia el microprocesador antecede a la actual transferencia de dato. El estado de información en el bit patrón, llamado bandera, indica la condicón de disposición de un puerto de Entrada/Salida. El *software* examina el estado de la bandera incrementando el tiempo asociado con la operación de Entrada/Salida. El tiempo adicional es llamado tiempo elevado de Entrada/Salida. Si varios dispositivos de Entrada/Salida son usados en un sistema con programa de Entrada/Salida, el proceso necesita checar la bandera por cada dispositivo en turno, este proceso es conocido como de poleo.

La figura 3-9 muestra un ejemplo usando *handshaking* en una operación de Entrada/Salida. Un dispositivo de entrada tiene datos disponibles en el puerto 1 para la transmisión a la microcomputadora. Para indicar la disponibilidad de datos, el dispositivo de entrada envía un bit de bandera en el puerto de entrada 0. La microcomputadora periódicamente accesa la bandera de estado en el puerto de entrada 0 y la examina. Si es 1, el dato esta disponible y la microcomputadora acepta el dato en el puerto de entrada 1. La señal de selección (*chip select*) del puerto 1 habilita el dato hacia el puerto 1 y también inicializa la bandera.

Mapa de Entrada en la IBM PC.

Para generar correctamente la señal de selección (*chip select*), se necesita conocer las direcciones del puerto de Entrada/Salida y el mapa de asignación del puerto. El diseño de la PC proporciona 10 bits para las direcciones de los puertos, desde A0 hasta A9, para direccionar un total de 1024 puertos. El mapa de direcciones de puertos esta dividido en dos partes. Las direcciones desde 0000H a 01FFH están asignadas para la tarjeta principal. El espacio de las direcciones de 0200H a 03FFH se reservan para las tarjeta de expansión. La tabla siguiente muestra las direcciones de los puertos que están destinados con anterioridad a las características de las tarjetas IBM. Se tienen únicamente 32 direcciones (300H a 31FH) designadas a puertos

para las tarjetas prototipos.

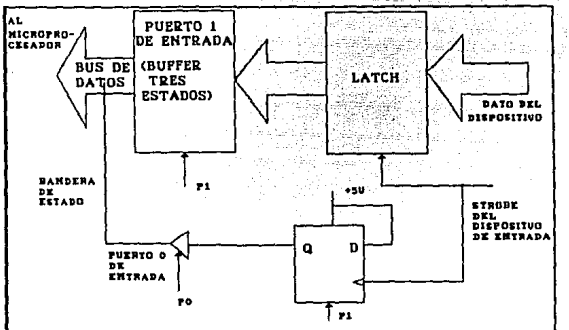


Figura 3-9. Handshaking usando una bandera de estado.

Rango-hexadecimal	Uso	
000-00F	DMA chip 8237A-5	Asignado para los
020-021	Interrupción 8259A	
040-043	Timer 8253-5	
060-063	PPI 8255A-5	
080-083	Registro de página del DMA	

0Ax	Registro de mascaras NMI	componentes de la
0Cx	Reservado	tarjeta principal
0Ex	Reservado	
100-1FF	No usado	
200-20F	Control de juego	
210-217	Unidad de expansión	
220-24F	Reservado	
278-27F	Reservado	
2F0-2F7	Reservado	Asignado para
2F8-2FF	Comunicación asíncrona	las tarjetas de
300-31F	Tarjeta Prototipo	puertos
320-32F	Disco Duro	
378-37F	Impresora	
380-38C	Comunicaciones SLDC	
390-399	Comunicaciones sincrona binaria (2)	
3A0-3A9	Comunicaciones sincrona binaria (1)	

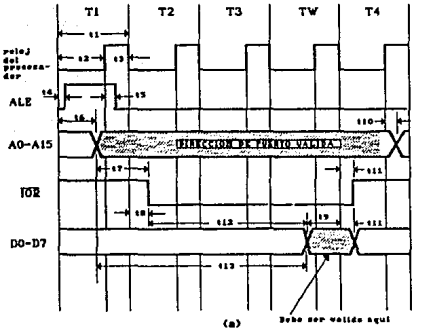
3B0-3BF	Desplegado monocromatico IBM/ impresora.	
3C0-3CF	Reservado	
3D0-3DF	Color/gráficos	
3E0-3E7	Reservado	
3F0-3F7	Discos	
3F8-3FF	Comunicaciones asíncrona	

Tiempos

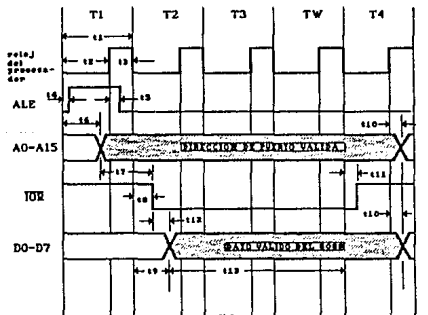
La llave de los diseños de alguna interfase en el sistema es el establecimiento de tiempos compatibles con el bus del sistema. La figura 3-10 presentan información de los tiempos en los ciclos de lectura y escritura en los puertos de Entrada/Salida que existen en el bus del sistema.

Estos ciclos de bus son normalmente de 4 ciclos de reloj, pero el diseño de la PC inserta tiempos adicionales que constituyen el llamado ciclo de espera. De este modo en la PC todos los ciclos de reloj son un mínimo de 5 (aproximadamente de 1.05 microsegundos). El ciclo de bus puede ser de gran ayuda para controlar la señal de listo (*IOCHRDY*) en el bus del sistema.

El ciclo de lectura del bus se inicia cada tiempo que el procesador ejecuta una instrucción *IN*. Durante T1, la señal *ALE* es activada, indicando con su flanco de bajada que los bits A0 hasta A15 del bus de direcciones contienen una dirección válida. Durante T2, la señal *IOR* del bus de control es activada, lo cual indica que se está direccionando al puerto de entrada para poder utilizar el bus de datos para la transferencia de datos. En el comienzo de T4, el procesador muestra el dato en el bus de datos y la señal de *IOR* es desactivada.



Simbolo	Max	Min
t1	-	209.5
t2	-	124.5
t3	-	71.8
t4	15	-
t5	15	-
t6	128	16
t7	-	91.5
t8	35	10
t9	-	42
t10	35	10
t11	-	10
t12	-	551.5
t13	-	668



Simbolo	Max	Min
t1	-	209.5
t2	-	124.5
t3	-	71.8
t4	15	-
t5	15	-
t6	128	16
t7	-	91.5
t8	35	10
t9	122	14
t10	-	10
t11	35	10
t12	112	-
t13	-	506.5

* todos los tiempos están en ns.

Figura 3-10. Tiempos para los puertos de Entrada/Salida.
(a) Tiempos del ciclo de lectura del puerto de entrada.
(b) Tiempos del ciclo de escritura del puerto de salida.

El ciclo de escritura del bus es iniciado cada tiempo que el procesador 8088 ejecuta una instrucción de salida. La señal *ALE* es activada, lo cual indica en el flanco de bajada que el bus de direcciones contiene una dirección válida de puertos durante T1. Después, la señal *IOW* del bus de control se activa durante T2, para seleccionar al puerto de salida que deberá tomar el dato del bus. Más tarde durante T2, el procesador 8088 colocará en el bus el dato para el puerto de salida. En el comienzo de T4, la señal *IOW* es desactivada.

3.3 Estructura de Interrupciones.

Una característica importante de una computadora es su estructura de interrupciones; la principal tarea de la estructura de interrupciones es proporcionar una forma eficiente para que el microprocesador responda rápidamente de eventos inciertos. El proceso de interrupciones incrementa el trabajo útil de la computadora, autorizando a los dispositivos periféricos el servicio de petición a el microprocesador cuando ellos lo necesiten, en lugar del proceso de poleo. Adicionalmente algunos microprocesadores tiene internamente un chip que genera las interrupciones.

Proceso de interrupción.

Una computadora es una máquina de estados finitos que tiene muchos estados únicos. La ejecución de un programa puede ser descrita como una secuencia de estados de máquina $S_0, S_1, \ldots\ldots, S_n$. Los elementos del estado de un programa son el (1) espacio de instrucción, (2) espacio de dato, (3) contador del programa y (4) estado del procesador.

El proceso de interrupción temporal cambia el estado de la máquina mientras ejecuta la rutina de servicio de interrupción. Para regresar a el estado previo a la petición de interrupción se requiere que dicho estado de la máquina haya sido guardado. El microprocesador

automáticamente guarda el contador del programa y el estado del procesador cuando la interrupción es conocida. La rutina de servicio de interrupción debe guardar todos los registros del procesador que pueden ser cambiados y restaurarlos antes de regresar el contador del programa y estado del programa a su estado previo. La pila (*stack*) es comúnmente usada para salvar el contador del programa, el estado del programa y los registros. Esta pila es direccionada por un registro (*stack pointer*) del microprocesador de Intel 8088.

La transferencia de control de la rutina principal a la rutina de servicio de interrupción toma lugar después que el contador del programa y estado del procesador son guardados. Las direcciones de la rutina del servicio de interrupción son guardadas en una tabla de vectores de interrupción. Un vector es una dirección de una rutina de servicio de interrupción. Cada dispositivo usa una entrada separada en la tabla del vectores de interrupción señalando su propia rutina de servicio de interrupción. La figura 3-11 muestra los conceptos en forma de diagramas de flujo, y la secuencia que siguen las interrupciones:

A) Una petición de interrupción ocurre durante las ejecución de estado S_{i+2}.

B) El microprocesador comienza el servicio de petición después de la ejecución del estado S_{i+2} y terminando en S_j con el estado S_{j+2} salvado en el Stack.

C) Las direcciones de la rutina del servicio de interrupción son encontradas en la tabla de vector de interrupción durante el estado S_j.

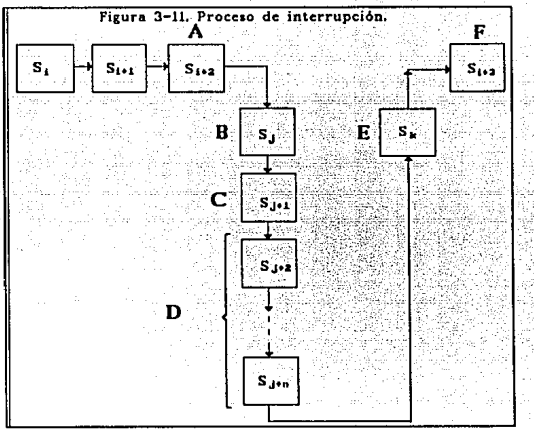
D) La rutina de servicio de interrupción se ejecuta en los estados S_{j+2} a S_{j+n}.

E) El estado S_{j+2} es guardado por el estado S_k.

F) Continua ejecutando la rutina principal en el estado S_{i+3}.

El servicio de rutina de interrupción pueden ser anidado en varios sistemas de computadoras. De tal manera que un dispositivo puede interrumpir la ejecución de una rutina de servicio de interrupción. La figura 3-12 muestra la transferencia de control durante el proceso de

anidamiento de petición de interrupción.



Típicas trampas encontradas usando interrupciones.

Las interrupciones deben ser usadas con precaución desde el programa normal debido a que su ejecución es temporalmente suspendida cuando una interrupción es atendida. Los cuatro principales temas a considerar son: (1)Interrupción de proceso crítico, (2) Pérdida de interrupciones, (3) Niveles de prioridad, y (4)Tiempo latente de interrupción. La confianza del desempeño del sistema depende propiamente del manejo de estos problemas potenciales.

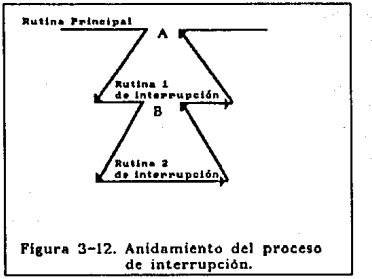


Figura 3-12. Anidamiento del proceso de interrupción.

Proceso crítico.

Este problema complica el proceso específico o tareas causados por el comportamiento incierto o indeseable del sistema si es interrumpido. Un ejemplo de esto ocurre cuando los registros de la memoria segmentada están siendo cambiados. Todos los programas usan registros de memoria segmentada para controlar las direcciones físicas de estructuras de datos críticos como el stack y el espacio de código. Si una interrupción ocurre mientras el stack esta siendo relocalizado, la ejecución puede no continuar propiamente después de la rutina de interrupción. Las interrupciones durante rutinas que tienen restricciones firmes pueden causar efectos indeseables.

Para asegurar el funcionamiento correcto, alguna tarea puede ser protegida de la interrupción deshabilitando las interrupciones mascarables durante las partes críticas de la rutina. Las interrupciones que ocurran cuando estén deshabilitadas pueden ser atendidas cuando las interrupciones sean habilitadas. Si un dispositivo intenta múltiple petición de interrupción mientras las interrupciones están deshabilitadas, las peticiones se perderán.

Interrupciones deshabilitadas.

El problema de pérdida de interrupciones puede ocurrir si las interrupciones están deshabilitadas por un tiempo largo, que puede ser en una parte crítica de la rutina. El resultado puede ser desfavorable en el sentido de que un dispositivo puede estar interrumpiendo y el microprocesador no lo hace caso, lo cual implica una pérdida de las interrupciones.

Prioridades de interrupciones.

Puesto que dispositivos diferentes pueden pedir el servicio de interrupción simultáneamente, un sistema de niveles de prioridades determina el orden del proceso de interrupciones. El dispositivo de máxima prioridad es atendido primero, seguido sucesivamente de los dispositivos de baja prioridad. Un sistema de almacenamiento de prioridades es usado cuando un dispositivo puede interrumpir y existe una rutina de servicio de interrupción en progreso. Típicamente, una rutina de servicio puede ser interrumpida por un dispositivo con alta prioridad. Si un dispositivo con igual o menor prioridad pide servicio, la petición es salvada hasta que la presente rutina de interrupción es completada.

Tiempo latente de interrupción.

El tiempo latente de interrupción es el intervalo entre el tiempo cuando la petición de interrupción es puesta y el tiempo cuando el rutina de servicio de interrupción es insertada. El tiempo esta compuesto por el tiempo requerido para la petición de interrupción del proceso después de que ha sido reconocido, más el tiempo necesario para completar la ejecución de la instrucción cuando la interrupción fue insertada. El tiempo de proceso de interrupción es la cantidad de tiempo necesario para que el microprocesador conozca la interrupción y pueda salvar el estado de la máquina. El tiempo necesario para completar la siguiente instrucción depende de

la instrucción. Las instrucciones que ejecutan varios desplazamientos tiene el potencial de ampliar el tiempo de completar la instrucción.

Tipos de interrupción.

Las peticiones de interrupciones generalmente son asociadas con los dispositivos externos del microprocesador como las unidades de disco, teclados, impresoras, y temporizadores que son interrupciones de hardware. La peticiones de interrupción generadas por el propio microprocesador son interrupciones de software.

Una petición de interrupción que puede ser ignorada por el microprocesador se conoce como interrupción mascarable. La interrupción mascarable es habilitada y deshabilitada por lenguaje de la máquina. Las líneas de petición de interrupción no mascarable siempre causan interrupción cuando son activadas, y puede ser usada como línea de petición mascarable si la operación mascarable es implementada por hardware.

Interrupción de *Hardware*.

Los dispositivos externos usan nivel de voltaje o un cambio de nivel voltaje para realizar una petición de interrupción. El método (por nivel o por flanco en la línea de interrupción) que el dispositivo debe de usar depende del ambiente del microprocesador. Si el microprocesador tiene un nivel de interrupción en la línea de interrupción, el dispositivo debe de mantener la petición de interrupción activa hasta que el microprocesador reconozca la petición. El microprocesador reconoce la interrupción activándose una o más líneas de estado. El microprocesador que tiene interrupción por flanco usa flip-flops internos para la captura de la petición de interrupción. El microprocesador debe tener alguna forma de identificar cual el es el dispositivo que realiza el servicio de petición, así como la propia rutina de servicio de interrupción que será ejecutada.

La identificación de información es proporcionada por el dispositivo durante el reconocimiento de la secuencia de interrupción, o es definida por la línea que es activada. Por ejemplo el microprocesador 8085 tiene 4 líneas de interrupción de modo que el vector es por medio de posiciones fijas y una línea de petición interrupción causa que el microprocesador lea el dato del bus para información de identificación del dispositivo. La mayor parte de los dispositivos no responden a las señales de conocimiento de interrupción poniendo un código en el bus de datos, los circuitos integrados controladores de interrupción son usados para generar el código apropiado.

Las numerosas fuentes de interrupción de *hardware* pertenecen a una función de entrada o de salida. Otras dos fuentes de interrupción consideradas son la del circuito para la detección de fallas y el sistema de *reset*. Muchos de los sistemas contienen circuitos extra que aseguran la integridad del sistema. El desempeño del sistema puede ser afectado adversamente por los problemas de la fuente de suministro. Cuando el nivel de voltaje se acerca a un limite bajo del rango de operación del circuito integrado, el comportamiento puede resultar erróneo. Un daño durante la falla de alimentación es indeseable en el acceso al disco. Una computadora que tiene un sistema detección de fallas de potencia detecta pérdidas imprevistas de potencia de alimentación de corriente alterna. Esto es cuando la alimentación falla, los filtros (capacitores) que suministran potencia pueden mantenerse en un nivel de voltaje válido por un tiempo corto. Este tiempo es utilizado por el sistema de falla de alimentación emitiendo una petición de interrupción en la línea de interrupción no mascarable. La rutina de servicio de interrupción de falla de alimentación puede asegurar el paro del microprocesador.

Un problema de *hardware* en el arreglo de la memoria causará un comportamiento imprescindible. Un esquema de protección comunmente utilizado es checar la paridad de la *RAM*. En este caso un error en la paridad durante la operación de lectura dispara una interrupción. La rutina de servicio de interrupción reporta el error al usuario y entonces detiene

al microprocesador. Si se utiliza este método, un programa debe inicializar la memoria antes de hacer la lectura porque los valores presentados en la memoria *RAM* pueden no tener correctamente la paridad después de encender a la computadora. Este es la razón de que la PC IBM gasta algún tiempo en administración de memoria cada vez que el sistema es encendido.

La línea de *reset* en un microprocesador puede ser vista como un tipo de interrupción. La diferencia del *reset* con respecto a las otras interrupciones es que el estado previo de la máquina al activarse la señal de reset no es salvado. La interrupción del *reset* aborta al programa que se esta ejecutando y forza a la ejecución del inicio de las direcciones pre-establecidas. El usuario que hace un reset manual a la computadora da lugar a que el microprocesador tomo un estado conocido.

Interrupción por *Software*.

Las interrupciones de software son generadas desde el microprocesador. Las interrupciones lógicas ocurren cuando en una operación aritmética resulta un sobreflujo o una división sobre cero. Las interrupciones pueden también ser generadas por las instrucciones de la máquina. Los dos principales usos de las interrupciones de *software* son para examinar las rutinas de interrupción y para proporcionar una conexión transparente utilizando rutinas. La idea de la conexión transparente es que la rutina de propósito general (que es desconocida para el programador), puede ser accesada a través de la tabla de vectores de interrupción emitiendo una instrucción de *software* de interrupción. Una interesante implementación de esta técnica es la *ROM BIOS* (Sistema Básico de Entrada y Salida) de la IBM PC. La *BIOS* contiene las rutinas que la PC usa para la operación del sistema. La potencia de estas rutinas son parte permanente de la computadora y son accesadas a través de instrucciones de *software* de interrupción. Si las direcciones de alguna rutina de *BIOS* cambia en un nuevo modelo de la computadora, La rutina también será accesada a través de una misma interrupción y todo el *software* será compatible con

la computadora en sentido ascendente.

Interrupciones del Microprocesador del 8088.

La estructura de interrupciones de PC IBM esta basada directamente sobre el microprocesador 8088. La figura 3-13 muestra las fuentes de interrupción de el 8088. Es evidente que el 8088 soporta el proceso de interrupción interno y externo.

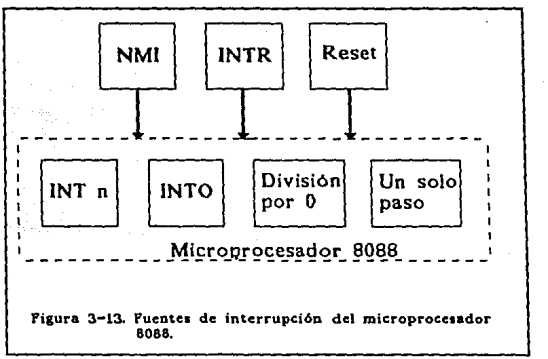


Figura 3-13. Fuentes de interrupción del microprocesador 8088.

3.4 Construcción Física de la tarjeta.

3.4.1 Tarjeta de acceso al slot de la PC.

Las primeras pruebas para la implementación de la tarjeta para el control de procesos industriales se realizaron en tabletas de experimentación por lo cual fué necesario elaborar una tarjeta de extensión para algunas señales del canal de entrada/salida de la PC proporcionadas por los *slots*. Esta tarjeta permite tener acceso a 40 terminales de uso frecuente en tarjetas prototipo; para realizar dicha tarjeta, se seleccionaron terminales del bus de datos, algunas del bus de direcciones (las empleadas para el direccionamiento de puertos) y señales de control (terminales del estado de los canales de datos y direcciones, de lectura y escritura en puertos y memoria, una señal de reloj, señales de petición de interrupciones, etc.); la figura 3-14 muestra el diagrama esquemático de la tarjeta de extensión.

El diseño del circuito impreso de la tarjeta se realizó con el programa de computadora *SMART WORK*, una vez terminado dicho diseño se elaboró el circuito impreso de la tarjeta por medio un proceso fotográfico que consiste en lo siguiente: se imprime en papel albanene el diagrama del circuito impreso de la tarjeta (por medio de un plotter) con un factor de escala de 2:1, se obtiene una fotografía, con reducción del 50%, al diagrama de la tarjeta, se revela el negativo, se prepara la tableta de vaquelita (se corta con las dimensiones adecuadas, se elimina el polvo, la grasa y se barniza con una solución sensible a la luz UV), se coloca el negativo de la tarjeta sobre la tableta de vaquelita y se expone a una fuente de luz UV intensa durante algunos minutos, posteriormente se procede a introducir la tarjeta en una solución corrosiva que ataca

las areas que no fueron expuestas a la luz UV, después se limpia con agua, se realizan las perforaciones necesarias, se colocan y se soldan los componentes que lleva la tarjeta (conector de cuarenta terminales y capacitores).

Las figuras 3-15, 3-16 y 3-17 muestran los diagramas del diseño de la tarjeta (lado de componentes, lado de soldadura, distribución de componentes y asignación de terminales en el conector de salida).

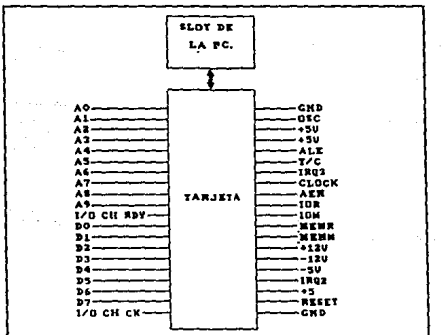


Figura 3-14. Diagrama de la tarjeta de acceso al bus del canal I/O de la IBM PC.

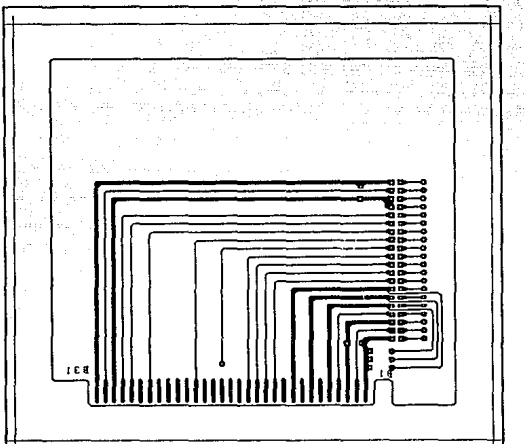


Figura 3-15. Circuito impreso de la tarjeta de extensión del SLOT (lado de soldadura).

3.4.2 Sistema con resolución de 8 bits.

El primer sistema implementado (8 bits de resolución) esta integrado por un convertidor analógico digital de ocho bits de resolución, con 8 entradas analógicas conectadas a un multiplexor (incluido en el convertidor) con líneas de control para la selección de la entrada a monitorear, un convertidor digital analógico de 8 bits, señales de salida para el manejo de relevadores y líneas de entrada para opto-acopladores.

Los bloques que constituyen la tarjeta se describen a continuación.

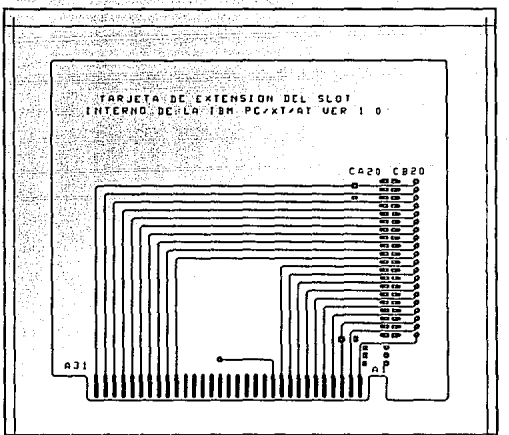


Figura 3-16. Circuito impreso de la tarjeta de extensión del SLOT (lado de componentes).

Bloque de decodificación.

El decodificador se diseño considerando el mapa de puertos de la tabla siguiente:

A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	DISPOSITVO
1	1	0	0	0	0	0	*	*	*	OPTOACOPLADORES

1	1	0	0	0	0	1	*	*	*	RELEVADORES
1	1	0	0	0	1	0	0	0	0	CANAL 1 (ADC)
1	1	0	0	0	1	0	0	0	1	CANAL 2 (ADC)
1	1	0	0	0	1	0	0	1	0	CANAL 3 (ADC)
1	1	0	0	0	1	0	0	1	1	CANAL 4 (ADC)
1	1	0	0	0	1	0	1	0	0	CANAL 5 (ADC)
1	1	0	0	0	1	0	1	0	1	CANAL 6 (ADC)
1	1	0	0	0	1	0	1	1	0	CANAL 7 (ADC)
1	1	0	0	0	1	0	1	1	1	CANAL 8 (ADC)
1	1	0	0	0	1	1	*	*	*	CONVERTIDOR D/A

La figura 3-18 muestra el diagrama de bloques; la función lógica elemental se implementó con el circuito *TIBPAL16L8* (arreglo lógico programable) y el circuito *74LS139* (decodificador 2x4). Como se observa en la figura el decodificador recibe señales que permiten programar (por medio de interruptores) el bloque de direcciones en la cual se encuentran localizados los diferentes dispositivos que integran la tarjeta (permitiendo ubicar la tarjeta una posición diferente cuando las direcciones $300-$31F se encuentran ocupadas por otros dispositivos).

Conversión analógica-digital.

La etapa de conversión A/D se obtiene con el convertidor *ADC0809* (con resolución de 8 *bits*), que recibe 8 señales analógicas conectadas a un multiplexor analógico interno, con 3 líneas

(para la selección de una de las 8 señales) proporcionadas por bus de direcciones, se cuenta además con una referencia de voltaje de +5V para minimizar el ruido proveniente de la fuente de alimentación.

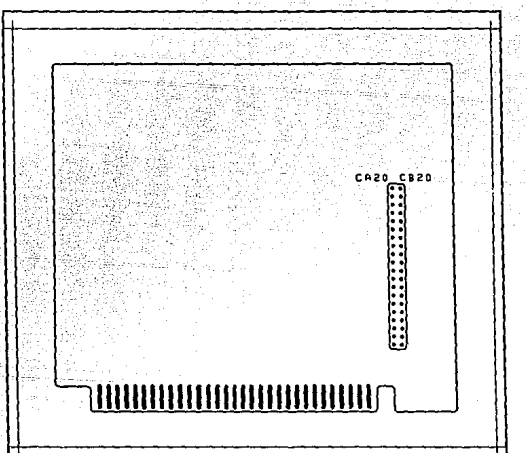


Figura 3-17. Circuito impreso de la tarjeta de extensión del SLOT (distribución de componentes).

El bloque de conversión A/D permite monitorear señales analógicas comprendidas entre los valores de 0 a 10V. El diagrama a bloques se muestra en la figura 3-19.

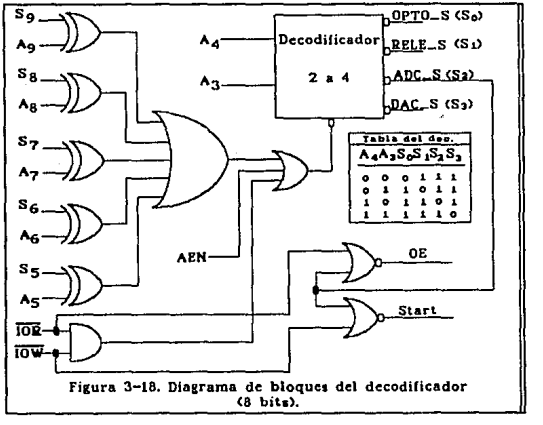


Figura 3-18. Diagrama de bloques del decodificador (8 bits).

Conversión digital-analógica.

Esta etapa se integra con el convertidor digital/analógico *DAC08* (con resolución de 8 bits), que maneja una señal analógica de salida dentro del rango de 0 a 10V. El diagrama a bloques se muestra en la figura 3-20.

Líneas de entrada digitales.

Las líneas de entrada digitales se emplean para el monitoreo del proceso industrial en forma aislada. El estado de estas señales se obtiene verificando el puerto de la tarjeta correspondiente

a los optoacopladores.

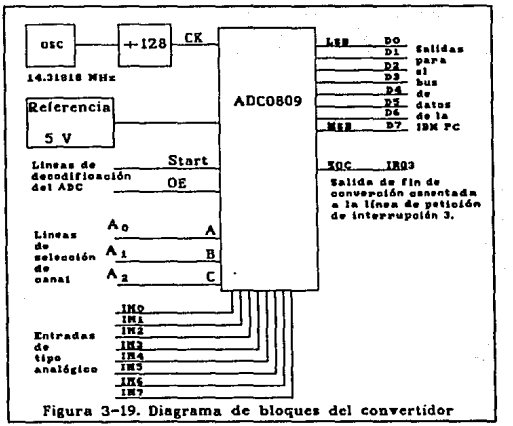


Figura 3-19. Diagrama de bloques del convertidor

Líneas digitales de salida.

Estas líneas se encargan de proporcionar las señales para activar y desactivar los equipos encargados del control del proceso industrial. Las señales se proporciona por medio del puerto asociado a los relevadores de la tarjeta.

Para probar el sistema se elaboraron programas que permitían monitorear, almacenar y

desplegar el valor de las señales analógicas conectadas al convertidor A/D; programas para monitorear las líneas de los opto-acopladores, ajustar el valor de los relevadores y generar señales analógicas con ayuda del convertidor D/A.

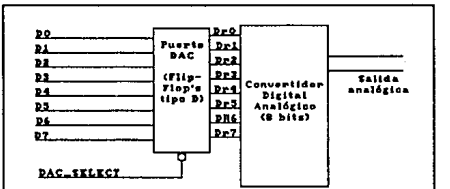


Figura 3-20. Diagrama de bloques del convertidor digital/analógico (8 bits).

Las figuras 3-21, 3-22 y 3-23 muestran el circuito electrónico de la tarjeta de 8 bits de resolución, las figuras 3-24, 3-25 y 3-26 muestran los diagramas elaborados en *SMART WORK* para la fabricación de la tarjeta.

Una vez que se construyó la tarjeta con 8 bits de resolución se procedió al diseño y fabricación de la tarjeta de control de procesos con 12 bits de resolución.

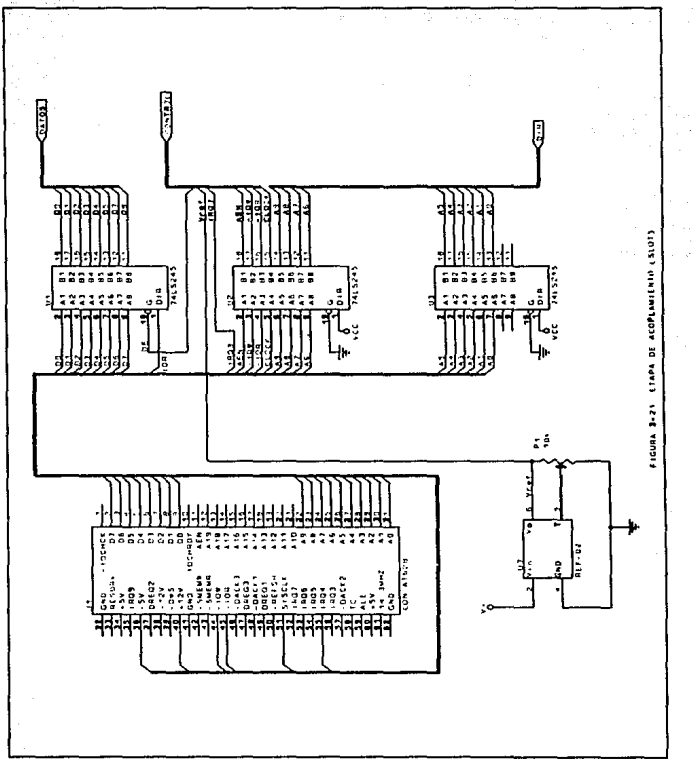


FIGURA 3-21 ETAPA DE ACOPLAMIENTO (SLOT)

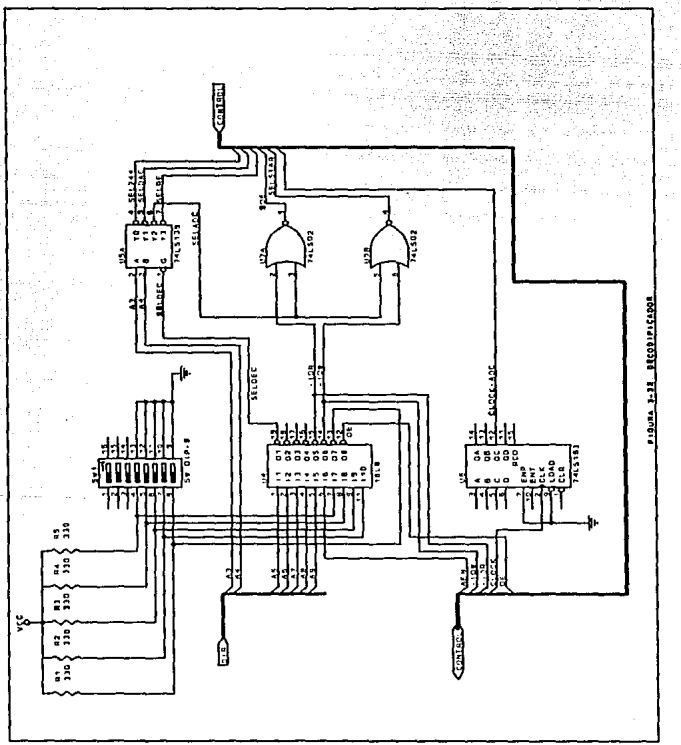


FIGURA 3-28 DECODIFICADOR

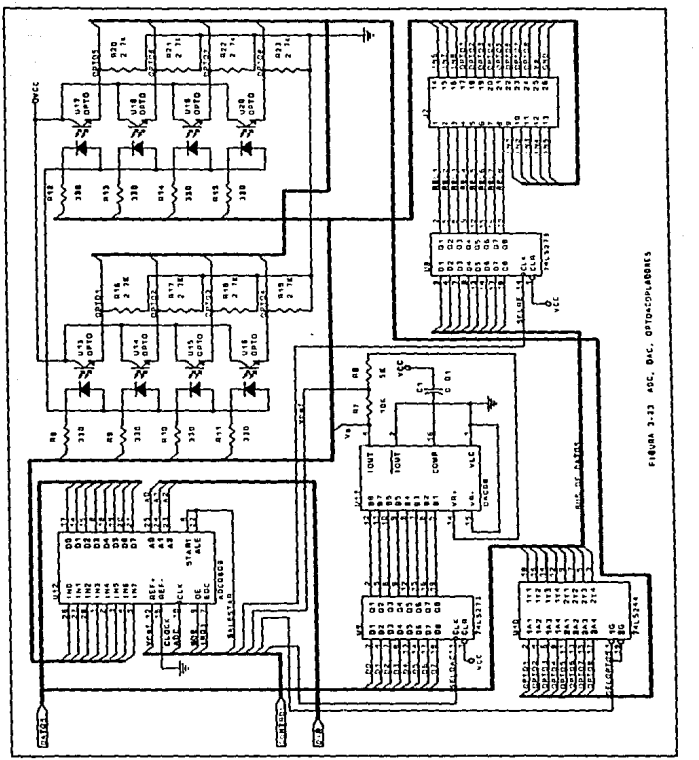


FIGURA 3-23 ADC, DAC, OPTOACOPLADORES

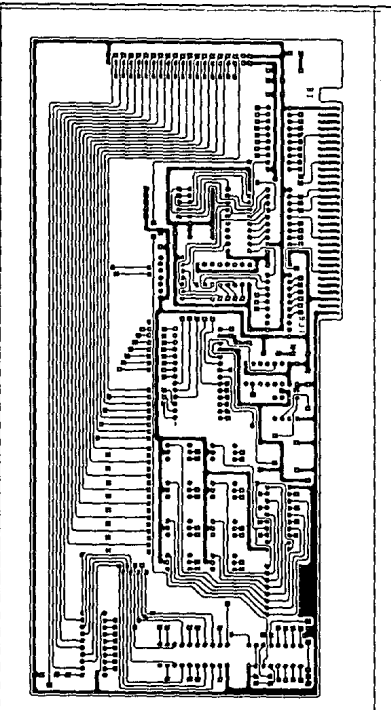


Figura 3-2 . Circuito impreso de la tarjeta de 8 bits de resolución (lado de soldadura).

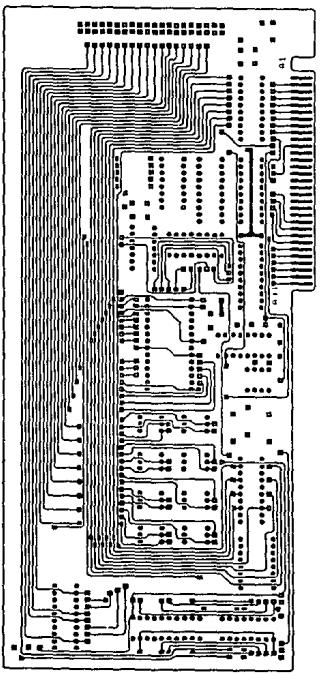


Figura 3-16. Circuito impreso de la tarjeta de 8 bits de resolución (lado de componentes).

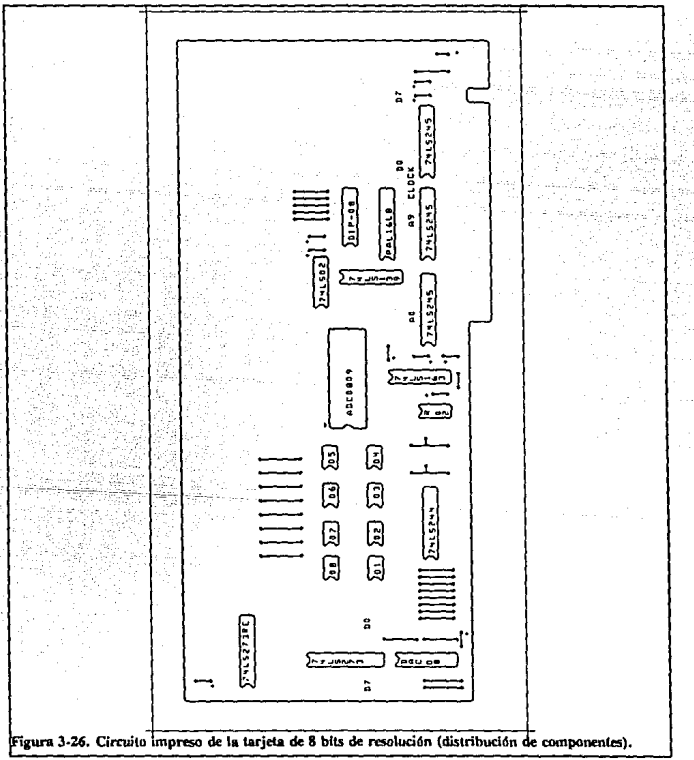


Figura 3-26. Circuito impreso de la tarjeta de 8 bits de resolución (distribución de componentes).

3.4.3 Tarjeta con resolución de 12 bits.

Los circuitos que integran esta tarjeta se describen a continuación y el diagrama de bloques se muestra en la figura 3-27.

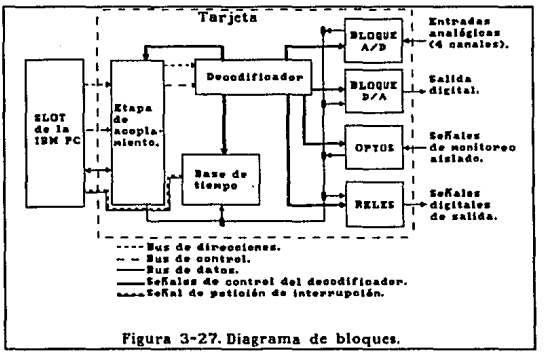


Figura 3-27. Diagrama de bloques.

Decodificador de la tarjeta.

La tarjeta se conecta en una de las ranuras de expansión de una computadora compatible con IBM PC/XT/AT. Para tener acceso a las señales de la ranura de expansión se colocó una etapa de aislamiento por medio de circuitos tres estados bidireccionales (*74LS245*); los circuitos para las líneas de direcciones y control se encuentran activos permanentemente, mientras que el circuito del Bus de Datos requiere una línea de habilitación proporcionada por el decodificador, cuando la computadora presenta las direcciones comprendidas en el intervalo $300-$31F (espacio

reservado para tarjetas prototipo).

El decodificador toma señales de control y direcciones activando los distintos dispositivos cuando la PC direcciona los puertos comprendidos entre las direcciones $300-$31F. El decodificador se encarga de activar todos los elementos de la tarjeta de acuerdo con el siguiente mapa de puertos:

A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	DISPOSITIVO
1	1	0	0	0	0	0	*	*	*	Convertidor A/D (4 canales)
1	1	0	0	0	0	1	0	0	0	Relevadores
1	1	0	0	0	0	1	0	0	1	Optoacopladores
1	1	0	0	0	0	1	0	1	0	Libre
1	1	0	0	0	0	1	0	1	1	Control
1	1	0	0	0	0	1	1	0	0	Convertidor D/A (LSB)
1	1	0	0	0	0	1	1	0	1	Convertidor D/A (MSB)
1	1	0	0	0	0	1	1	1	0	Libre
1	1	0	0	0	0	1	1	1	1	libre

Del mapa de puertos anterior se observa que las líneas de direcciones que no cambian son A_9-A_5, por lo tanto estas líneas se utilizan para implementar el decodificador de la tarjeta. La línea de selección de la tarjeta esta dada por la siguiente función lógica:

$$TS = A_9' + A_8' + A_7 + A_6 + A_5 + AEN \cdot IOW + IOR$$

Para implementar la función lógica anterior se incluyeron las señales de control *AEN*, *IOR* e *IOW* para asegurar que se tenga una dirección válida de puertos de entrada/salida. Esta función se puede implementar con circuitos inversores y una compuerta *OR*.

Con el fin de ahorrar espacio en la tarjeta se procedió a implementar el decodificador con un Arreglo Lógico Programable (*PAL16L8*) y un decodificador de 3 x 8.

Para generar la tabla de programación del PAL se utilizó el programa *CUPL* que minimiza funciones, genera archivos con el formato adecuado para programar el *PAL* y archivos de documentación del *PAL*.

El diagrama de bloques del decodificador se muestra en la figura 3-28.

Conversión analógico-digital.

La computadora, que directamente sólo puede procesar señales de tipo digital, para el control del proceso debe interactuar con dispositivos que tratan señales de tipo analógico o continuo, tales como temperatura, presión, humedad, etc. Para poder interactuar con este tipo de señales se requiere una interfase que emplea un sensor (transductor) para convertir una variable física (analógica no eléctrica) en una señal proporcional eléctrica analógica; la última se convierte en una señal digital por medio de un *ADC* (figura 3-29).

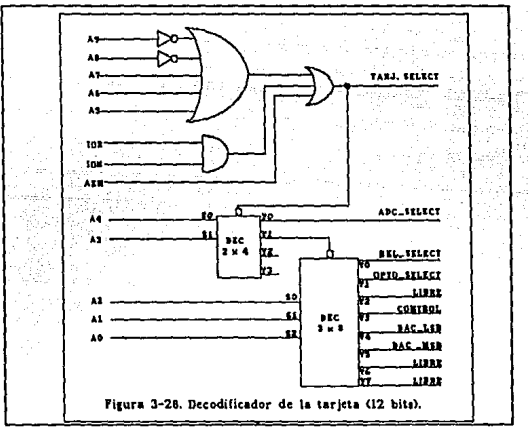


Figura 3-28. Decodificador de la tarjeta (12 bits).

Convertidor Analógico Digital 574 AKD.

El convertidor A/D esta asignado a las direcciones $300-$307 del mapa de puertos de la tarjeta. Recibe la señal analógica de las líneas de salida de un multiplexor analógico de 2 x 4.

Características del convertidor *HI-574A*©.

-Convertidor Analógico-Digital de 12 Bits con referencia y reloj.

-Interfase con el Bus del Microprocesador de 8, 12, 0 16 bits.

-Tiempo de acceso 150ns.

-Tiempo máximo de conversión 25μs.

-Inmune al ruido, vía modo sencillo en la transmisión entre circuitos.

-Opera con $\pm$12 a $\pm$15V.

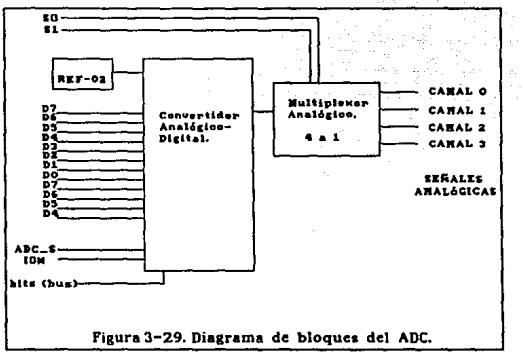


Figura 3-29. Diagrama de bloques del ADC.

Aplicaciones.

-Sistemas de adquisición de datos en cuestiones militares y en la industria.

-Instrumentos Electrónicos y Científicos.

-Sistemas de control de procesos.

Descripción

El HI-574A es un convertidor Analógico Digital de 12 Bits de resolución, incluye una referencia de 10 Volts y reloj, salida tres estados y una interfase digital para el bus de control de microprocesadores. La conversión es de aproximaciones sucesivas.

Error de Linealidad.

La linealidad es una especificación de la máxima desviación de la salida medida con respecto a la línea recta que se extiende en todo el margen de la forma de onda. Puede ser expresada como porcentaje del voltaje total de la escala o como una fracción del voltaje equivalente del LSB y debe ser menor que ½ LSB.

El HI-574AK, AL, AT y AU se garantiza para una no linealidad de ±½ LSB. El rango del HI-574AJ y AS esta garantizado a ±½ LSB máximo de error.

Voltaje analógico de entrada. Es el máximo margen permisible de voltaje de entrada. Los valores típicos son 0 a 10V, de 0 a 20V, ±5V ó ±10V.

Impedancia de entrada. Los valores están comprendidos entre 5kΩ y 10kΩ.

Sensibilidad a la temperatura. Para una entrada digital fija dada, la salida analógica varía con la temperatura a causa de que las fuentes de voltaje de la referencia y de que las resistencias son sensibles a la temperatura. Los coeficientes de temperatura son de 10 a 45*ppm*/°C dependiendo del modelo.

Tiempo de conversión. El tiempo máximo de conversión es de 25 μs .

Aplicando el HI-574A.

Para cada aplicación de este convertidor, las conexiones de tierras, las desviaciones de las fuentes, sincronización digital y rutinas de señales en la tarjeta del circuito deben ser optimizadas para asegurar un máximo desempeño.

Consideraciones físicas de montaje y presentación.

Al diseñar el circuito se debe tratar de no tener circuitos parásitos (L, R y C), que puedan hacer imposibles las mediciones de 12 bits, al nivel del convertidor A/D. La mejor política es eliminar o minimizar estos circuitos parásitos a través de un apropiado diseño del circuito antes de cuantificar sus efectos.

La recomendación en la construcción es imprimir a doble tamaño la tarjeta del circuito con un plano de tierras de acuerdo al tamaño del circuito.

En general, las señales analógicas deberán ser trazadas entre líneas de tierra y guardando otra forma las líneas digitales. Si las líneas analógicas y las líneas digitales deben cruzarse, estas deberán estar en ángulos rectos.

Fuentes de suministro.

Los voltajes de suministro en el HI-574A (+15V, - 15V y +5V) deben ser bien regulados. Los picos de voltaje puede afectar la exactitud, causando que los bit menos significativos oscilen cuando la entrada aplicada es constante. El ruido digital y el ruido de las fuentes de conmutación son especialmente molestos. Si las fuentes conmutadas son necesarias, las salidas deberán ser cuidadosamente filtradas para asegurar un voltaje de DC constante en las terminales del convertidor.

Adicionalmente, un par de capacitores de paso deben colocarse en cada terminal de suministro de voltaje, para restar el efecto de la variación de la corriente de suministro. Se recomienda conectar un par de capacitores entre las treminales 1 y 15 (*Vlogic supply*), otro entre las terminales 7 y 9 (*Vcc*) y otro entre las terminales 11 y 9 (*VEE*). Cada par de capacitores se integra con uno de tantalio de 10μF en paralelo con un cerámico de 0.1μF.

Conexiones a tierra.

Las corrientes a tierra del HI-574A son de 5.5 mA de DC en la terminal 9 (Analógico) y 7 mA a la salida de la terminal 15 (Digital). Estas terminales deben ser unidas para un mejor funcionamiento del convertidor. Además la terminal 9 debe tener la pista cerca de +15V y la terminal 15 junto a +5V. Si el convertidor esta lejos del sistema, se unen las terminales 9 y 15; una vez juntas deben salir con una sola línea hacia la tierra del sistema.

Transductores.

Los transductores son muy diversos y están basados en una gran variedad de fenómenos eléctricos. Una clase importante de transductores depende de la posibilidad de que la variable analógica de interés altere la resistencia del transductor. Hay numerosos transductores basados en resistencias, medida de voltajes y potenciómetros.

Referencia de voltaje.

La referencia que se utilizó es el circuito *REF-02* que es una referencia de voltaje de presición que proporciona una salida estable de +5V, la cual puede ser ajustada sobre un rango ±6% con un mínimo de efecto en la estabilidad por la temperatura. Bajo costo, bajo en ruido y baja potencia hacen que REF-02 sea un excelente dispositivo de referencia estable.

Multiplexor Analógico.

El multiplexor analógico nos permite monitorear distintas variables físicas empleando un solo convertidor A/D. El multiplexor recibe señales asociadas a distintos transductores y cada señal se puede conectar a las líneas de salida del multiplexor por medio de señales de control.

Base de tiempo programable.

Debido a que la tarjeta se puede instalar en una computadora del tipo IBM PC/XT/AT (cada una con diferente velocidad de procesamiento), se cuenta con una base de tiempo programable por software que indica la frecuencia de muestreo de las señales monitoreadas por el convertidor A/D. La frecuencia de muestreo cambia de acuerdo a la computadora empleada, teniendo como límite superior un valor de 0.04MHz.

La base de tiempo se obtiene de un reloj (implementado con un cristal de cuarzo, resistencias, capacitores y compuertas lógicas), contadores binarios de 4 bits y un divisor de frecuencia con 6 líneas de selección (proporcionadas por las terminales D2-D7 del puerto de control) para el ajuste de la frecuencia de muestreo.

Optoaisaladores.

El diagrama de bloques de un optoaislador se muestra en la figura 3-30. Un optoaislador es un dispositivo que contiene un emisor acoplado por medios ópticos con un fotodetector a través de un medio corto de aislamiento. Este arreglo permite pasar información de un circuito, que contiene el emisor, a otro circuito que contienen el receptor.

El paso de la información es por medios ópticos y en un solo sentido, de modo que el emisor no puede realizar la función de recibir información y el receptor no puede transmitir información.

La tarjeta cuenta con circuitos *4N2* (figura 3-31) para el monitoreo aislado del proceso industrial, este circuito cuenta con un diodo emisor de luz y el receptor esta implementado por un transistor. La entrada esta asociada al estado de una variable del proceso a controlar y las

salidas se conectan a un circuito tres estados (*74LS273*) que constituye el puerto de datos con la dirección $300, el estado de cada circuito de monitoreo aislado se asocia a un bit del bus de datos de este puerto.

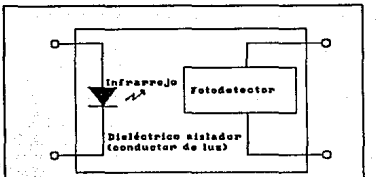


Figura 3-30. Diagrama de bloques del Optoaislador.

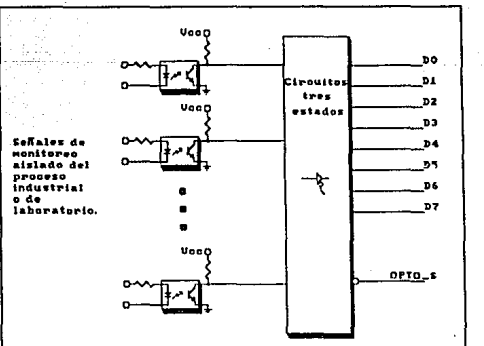


Figura 3-31. Bloque de señales opto-aisladas.

Señales de control.

La tarjeta cuenta con 8 señales para la corrección de algún parámetro del proceso industrial proporcionadas por relevadores o etapas de potencia. Las señales son proporcionadas por el puerto $318, formado por un registro de 8 bits (8 *flip-flops* tipo *D*). El diagrama de bloques de esta etapa se muestra en la figura 3-32.

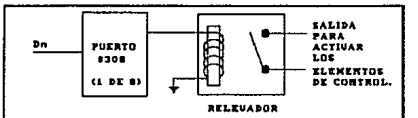


Figura 3-32. Diagrama de bloques para los relevadores.

Convertidor digital analógico.

La tarjeta proporciona una salida analógica para implementar algoritmos de controladores diferentes al de dos estados, o generar señales (senoidal, cuadrada, triangular) cuando el proceso así lo requiera. El diagrama de bloques se muestra en la figura 3-33.

Descripción del *DAC800 CBI.*

El *DAC800* es un circuito integrado de la tercera generación que tiene una equivalencia (terminal a terminal) con el circuito *DAC80* introducido por la empresa *Burr-Brown*. Tiene todas las funciones de sus predecesores, con menor tiempo de establecimiento[10].

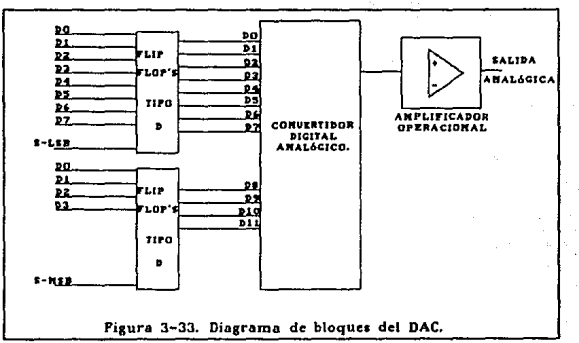


Figura 3-33. Diagrama de bloques del DAC.

El modelo de corriente de salida del *DAC800* se forma con un circuito integrado que contiene un diodo zener de referencia, alta velocidad en el cambio de corriente y resistencias de precisión de película delgada. El *DAC800* provee rangos de salida de voltaje de ±2.5V, ±5V, ±10V, 0 a +5V, 0 a +10V (modelo V) o rangos de corriente de salida de ±1.175mA ó de 0 a -2.35mA (modelo I).

El convertidor tiene un error de no linealidad de ±½LSB, ±30ppm/°C, operando de 0°C a +70°C. En la configuración bipolar, a carga completa se tiene garantizado menos de 25ppm de FSR/°C.

El *DAC800* viene en un circuito de 24 terminales distribuidas en dos líneas, con la misma configuración que el *DAC80*.

Especificaciones.

Resolución. Este término especifica el número de bits que el convertidor puede acomodar y el número correspondiente de voltajes de salida (o corrientes). Por ejemplo, un convertidor que puede aceptar 10 bits de entrada se dice que es un convertidor con resolución de 10 bits, el número de voltaje de salida posible es de 2^{10}= 1024. Por tanto, el menor cambio posible del voltaje de salida es de 1/1024 del alcance o margen de salida a plena escala. Adoptando 1000 como aproximación de 1024 podemos describir la resolución como 1 parte en 1000 o 0.1%. La resolución del *DAC800* es de 12 bits.

Linealidad. Es una especificación de la máxima desviación de la salida medida con respecto a la línea recta que se extiende en todo el margen de la forma de onda. El *DAC800* tiene un error lineal de $\pm$½ *LSB* a una temperatura de 25°C.

Exactitud. La exactitud de un convertidor es una medida de la diferencia entre el voltaje de salida analógica real y la correspondiente en el caso ideal. La falta de linealidad introduce imprecisiones. Contribuyen a ulteriores limitaciones de la precisión, la incertidumbre del voltaje de referencia la ganancia del amplificador, el offset del amplificador, etc.

Tiempo de establecimiento (*settling*). Cuando cambia la entrada digital en un convertidor, los interruptores se abren y cierran y aparecen variaciones bruscas de voltaje. A causa de las inevitables capacidades e inductancias parásitas presentes en los circuitos pasivos, los transistorios así iniciados pueden persistir durante un tiempo apreciable. Se originan transitorios adicionales debido a las características de los dispositivos activos (interruptores, transistores, etc.). El tiempo de establecimiento del *DAC800* al 0.01% del rango de la escala completa para un voltaje de 20V es de 5μs y para 10V es de 4μs.

Sensibilidad a la temperatura. Para una entrada digital fija cualquiera, la salida analógica

variará con la temperatura. Esta sensibilidad a la temperatura es ±30ppm/°C para el DAC800..

Configuración del convertidor.

El *DAC800* es un convertidor de 12 bits de resolución por lo cual se implementó una etapa de acoplamiento al bus de 8 bits de la IBM PC/XT/AT. El acoplamiento se lleva a cabo por medio de 3 circuitos *74LS245* (registro octal *-flip flops* tipo *D-*) asociados a dos puertos de salida, un puerto controla el acceso de los 4 bits más significativos a uno de los circuitos *74LS245* y el otro activa el *74LS245* asociado a los 8 bits menos significativos en conjunto con los cuatro bits presentes en el otro circuito *74LS245*, proporcionando simultáneamente los 12 bits requeridos por el *DAC800*.

Al igual que el convertidor analógico digital el nivel de voltaje lógico se toma de la referencia de 5V (*REF-02*).

Elaboración del circuito impreso para la tarjeta con resolución de 12 Bits.

Una vez que se probaron todos las etapas en forma individual y en conjunto en tarjetas de experimentación, se elaboró el circuito impreso de dos tarjetas de prueba. El proceso es similar al que se siguió en la elaboración de la tarjeta de extensión del *slot* de la PC. Una de las tarjetas contiene los circuitos de interfase con los buses de la PC, el decodificador de la tarjeta, el bloque de conversión A/D y D/A, y los circuitos que proporcionan las señales para el control del proceso y el monitoreo de los optoacopladores; la segunda tarjeta cuenta con relevadores, 8 optoacopladores y circuitos para el manejo de los transductores empleados en el control del proceso industrial, así como los circuitos de potencia necesarios para activar el equipo encargado del control del proceso.

El diagrama electrónico de la tarjeta se muestran en las figuras 3-34 y 3-35. Las figuras 3-36, 3-37 y 3-38 muestran las tarjetas elaboradas en *SMART WORK* para la fabricación de la tarjeta.

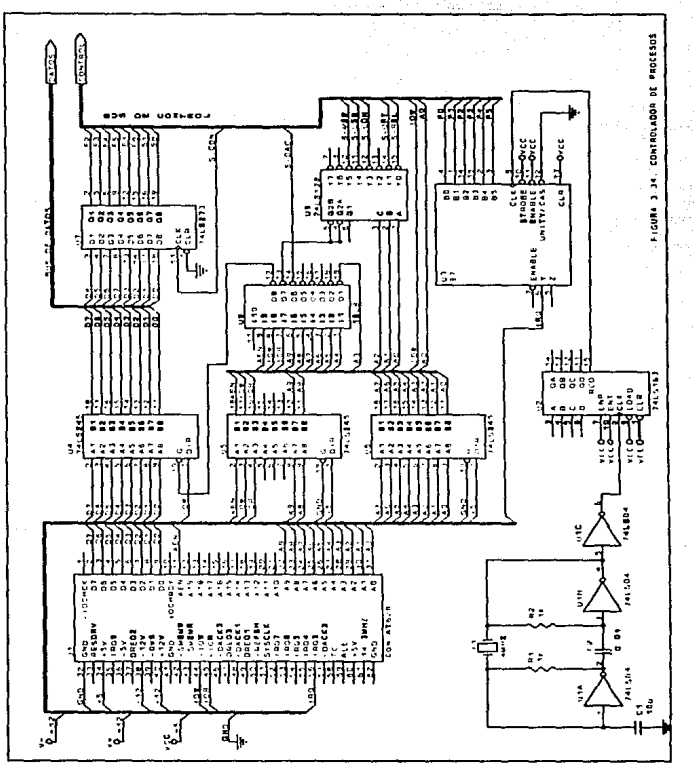


FIGURA 3.34. CONTROLADOR DE PROCESOS

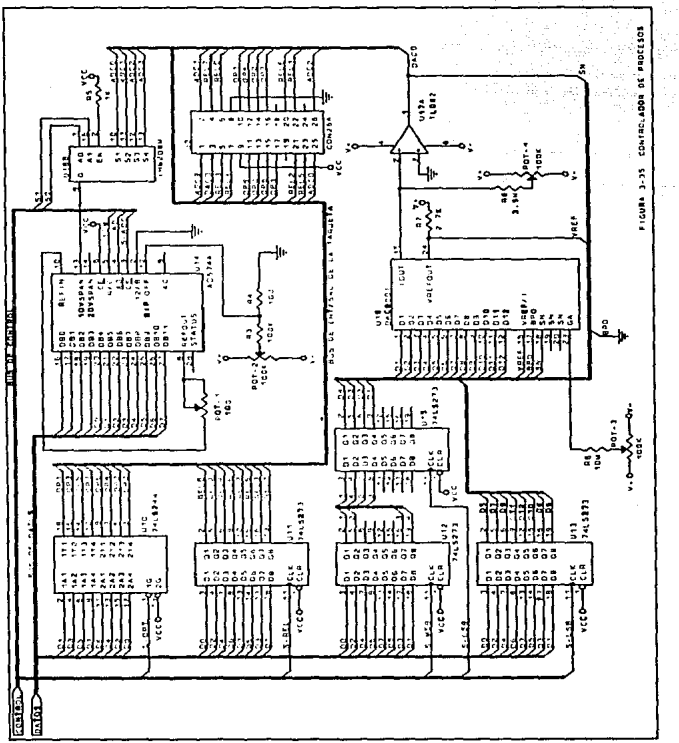


FIGURA 3-35 CONTROLADOR DE PROCESOS

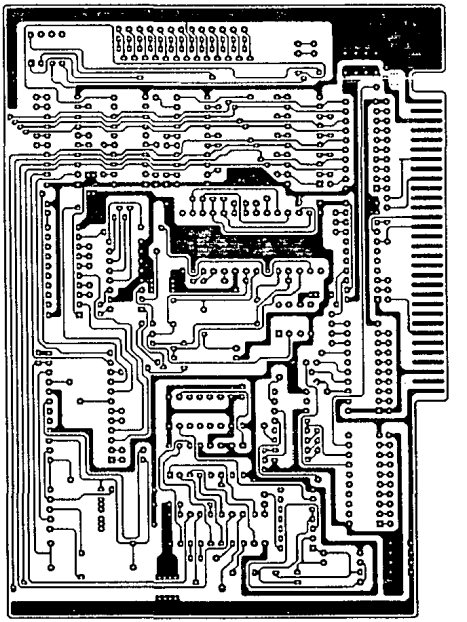


Figura 3-36. Circuito impreso de la tarjeta con resolución de 12 de bits (lado de soldadura).

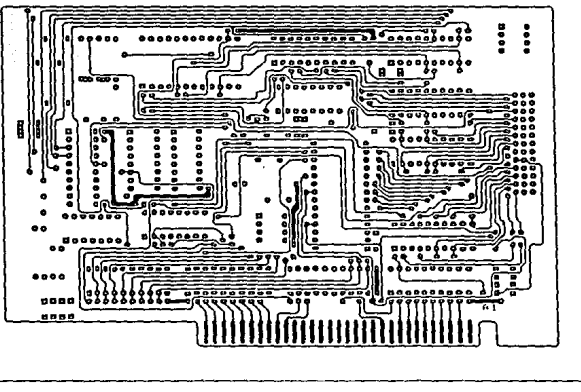


Figura 3-37. Circuito impreso de la tarjeta con resolución de 12 bits (lado de componentes).

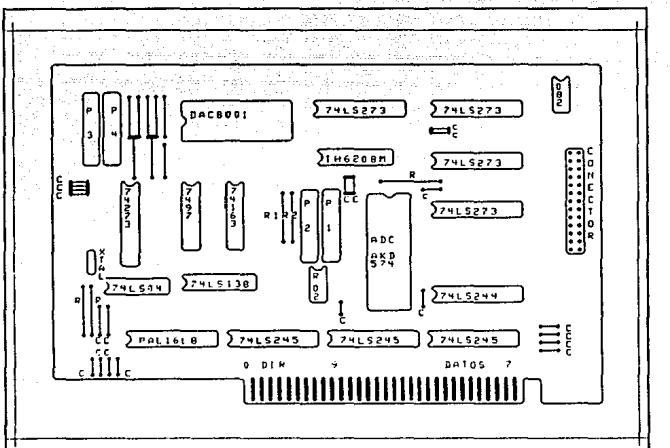


Figura 3-38. Circuito impreso de la tarjeta con resolución de 12 bits (distribución de componentes).

En el capitulo 5 se presenta una solución práctica, en la adquisición de imágenes digitalizadas de señales ultrasónicas, en la cual se comprueba el correcto funcionamiento de la tarjeta desarrollada.

IV. DESCRIPCIÓN GENERAL DE FUNCIONAMIENTO ("SOFTWARE").

4.1 Programación orientada a objetos.

El programa de aplicación se realizó en el lenguaje de alto nivel Turbo Pascal 6, empleando programación orientada a objetos (*POO*) y la librería de *TURBO VISIÓN*.

La Programación Orientada a Objetos es el método más reciente y actual de desarrollo de *software* que existe. En la versión 5.5 Borland introdujo la *POO* en el mundo de Pascal. En la versión 6.0, el soporte para objetos es aún mejor.

La *POO* se basa fuertemente en el concepto de objeto. En nuestras vidas diarias estamos familiarizados con cualquier objeto: televisiones, lámparas, cuademos, etc. Pero cuando encendemos la televisión, no distinguimos entre sus elementos físicos (selector de canal, tubo de imagen, antena) y su comportamiento (proporcionar imagen y sonido). Simplemente la encendemos y seleccionamos un canal.

Al igual que la televisión, los objetos hacen que los programas sean un reflejo más fiel de la forma en que tratamos el mundo real.

En la programación de Pascal se definen estructuras de datos que contienen información y procedimientos y funciones para manejar información. En *POO* los datos y los procedimientos se combinan en objetos.

Un objeto contiene las características de una entidad (sus datos) y su comportamiento (sus procedimientos). Combinando estas características y comportamientos, un objeto conoce cualquier cosa que necesite para hacer su trabajo.

Código + Datos = Objetos

Turbo Visión.

Turbo Visión es la interfase de los mecanismos subyacentes que ha utilizado Borland para construir el Entorno Integrado de Desarrollo (*EID*) de Turbo Pascal. Al dar acceso a estos mecanismos, Borland ha permitido que se pueda desarrollar *Software* con el mismo aspecto y filosofía que el *EID*[11].

La mayor utilidad de Turbo Visión es que consiste en una enorme colección de *Software* realizable. Por ejemplo, para crear una barra de menú, simplemente habrá que definir una variable objeto de tipo *TMenúBox* e incluir los elementos del menú en el objeto. Si *Tmenú* no maneja lo bastante una determinada capacidad, se puede crear un nuevo tipo objeto que herede toda la funcionalidad de *Tmenúbox* pero que añada la nueva capacidad; se procede del mismo modo para crear todos los elementos que integran el *EID*.

4.2 Programa de aplicación.

El programa de aplicación utilizado proporciona comandos para probar los diferentes

bloques de la tarjeta y comandos para que el sistema funcione en forma automática, una vez programadas las secuencias a ejecutar.

El programa de monitoreo (adquisición de datos) se implementó empleando un esquema de interrupciones por *Hardware*, esto es, la base de tiempo de la tarjeta interrumpe a la computadora en forma periódica a una frecuencia de muestreo previamente seleccionada, para leer el valor de las variables analógicas (presentes en el *ADC*) y digitales (del puerto asociado a los opto-acopladores), guardándolas en memoria o disco para su procesamiento posterior. El diagrama de flujo del programa de monitoreo se muestra en la figura 4-1, en él se pueden apreciar las rutinas que se realizan de acuerdo al estado de las variables del sistema.

El proceso a controlar consiste en realizar un muestreo en un plano bidimensional haciendo un barrido del material bajo prueba con la ayuda de los motores de pasos, controlados directamente desde la PC mediante la tarjeta controladora de procesos industriales, así se ubica al sensor en un punto fijo (X,Y), se excita al transmisor usando un pulso, y mediante un amplificador de instrumentación y un filtro analógico se procede a realizar la recepción de los ecos provocados por el transmisor, digitalizándose los resultados obtenidos con la ayuda del convertidor de 12 bits de resolución, los cuales se almacenan en memoria o disco. De esta manera se procede hasta terminar con el plano prefijado; finalmente se realiza el procesamiento de los datos adquiridos en todo el barrido, detectando los distintos rebotes obtenidos y se realiza una representación esquemática del resultado del procesamiento en un monitor de alta resolución (*VGA*).

El diagrama de flujo para el control del proceso se muestra en la figura 4-2, donde se definen los pasos de posicionamiento, activar Transmisor, procesamiento y entrega de resultados.

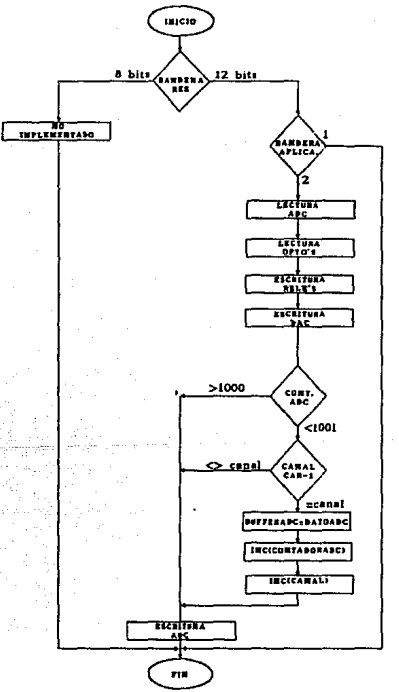


Figura 4-1. Diagrama de flujo del programa de monitoreo.

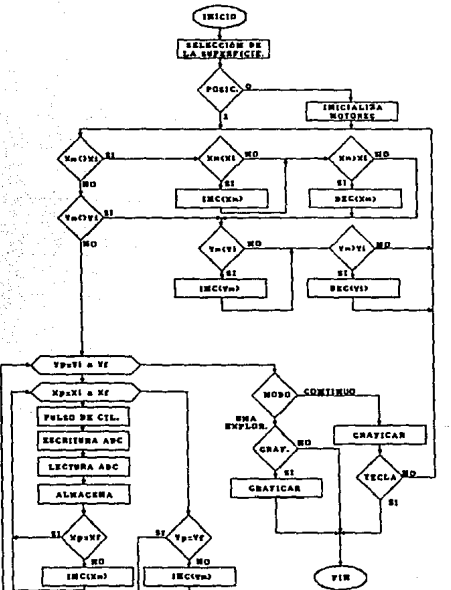


Figura 4-2.diagrama de flujo del programa de control del proceso industrial.

Programa principal.

Un sitio para cada cosa y cada cosa en si sitio. Este hecho describe exactamente el programa de aplicación que consiste en secciones bien definidas, cada una de las cuales sirve para un propósito específico. El entorno se integra con una serie de menús (que muestran todos los procedimientos disponibles), múltiples ventanas solapadas de tal forma que puede observarse el estado de distintos módulos del proceso industrial a controlar, además el empleo del ratón hace que el uso del programa sea sumamente agradable; se puede saltar de ventana en ventana, seleccionar opciones de menú, mover y variar el tamaño de las ventanas y más cosas con la sencillez de "apuntar y pulsar". Aunque el ratón no es estrictamente necesario podemos observar (al manejar el programa) que el entorno de trabajo es mucho más fácil de usar con el ratón.

En el programa se utiliza el botón izquierdo del ratón para seleccionar una ventana o activar una opción de menú. El otro botón se encuentra libre para realizar diferentes tipos de funciones.

El menú principal.

El programa utiliza un sistema de menús que soportan un acceso por abreviaturas de teclado a las operaciones utilizadas frecuentemente. Las opciones del menú principal son: Sistema, Prueba, Motores de pasos y Aplicación, tal como se muestra en la figura 4-3.

Existen dos formas de seleccionar una opción del menú principal: utilizando el teclado o utilizando el ratón. La tecla F10 activa el menú principal resaltando la opción actualmente seleccionada. Se puede seleccionar otra opción moviendo el área resaltada utilizando las teclas FECHA DERECHA y FLECHA IZQUIERDA del teclado numérico de la computadora y presionando la tecla INTRO para aceptar la opción. Un método más directo es presionar la tecla F10 y la letra resaltada de la opción deseada. Con el ratón sólo se requiere ubicarlo sobre la opción deseada y presionar el botón de la izquierda.

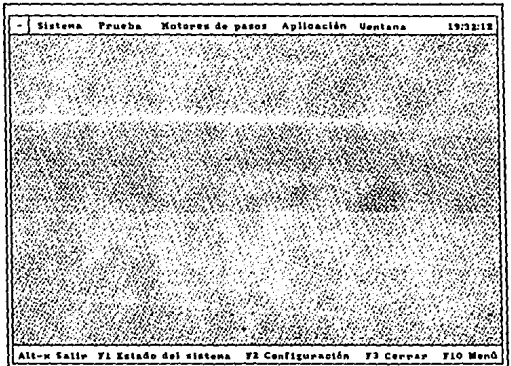


Figura 4-3. Menú principal.

La línea inferior de la pantalla muestra una lista de las abreviaturas de teclado usadas con más frecuencia. En cualquier parte donde se encuentre, se puede seleccionar cualquiera de las abreviaturas de teclado indicadas pulsando la tecla correspondiente a la opción deseada (por ejemplo F1 para observar el estado de las variables del sistema) o pulsando el ratón sobre la opción (en la pantalla) para obtener el mismo resultado.

El menú -

Tal como lo muestra la figura 4-4 este menú cuenta con una sola opción que nos despliega información asociada al sistema desarrollado. Para seleccionar la opción se presiona la tecla INTRO o se presiona el botón izquierdo con el ratón sobre la opción.

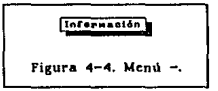


Figura 4-4. Menú -.

La ventana que se activa al elegir la opción Información se muestra en la figura 4-5.

El menú Sistema.

Este menú contiene funciones para mostrar el estado general de los diferentes módulos de la tarjeta y la configuración de la misma. También cuenta con el comando que permite terminar la ejecución del programa.

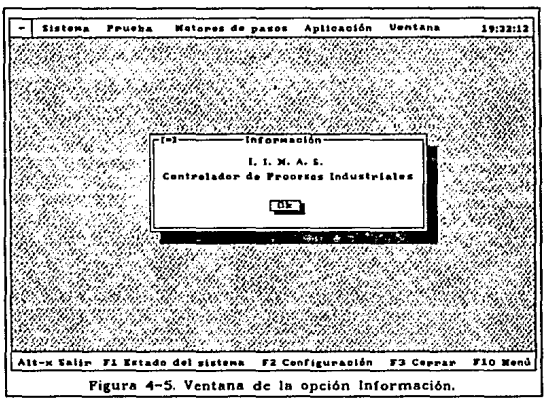


Figura 4-5. Ventana de la opción Información.

El menú Sistema muestra las opciones listadas verticalmente, como se observa en la figura 4-6.

Figura 4-6. Menú Sistema.

Para seleccionar una opción se utilizan las teclas FLECHA ARRIBA y FLECHA ABAJO para resaltar la opción y luego se presiona la tecla INTRO. Algunas opciones tienen abreviaturas de teclado equivalentes (por ejemplo, F1 para Estado general) que pueden acelerar un poco las cosas. Como siempre con el ratón es más fácil: sólo se requiere apuntar la opción y presionar el botón izquierdo del ratón.

Estado general F1.

La opción Estado general permite observar el estado de las variables asociadas a los diferentes módulos del sistema, esto es, muestra el valor de las señales que se encuentran conectadas al convertidor A/D, el estado lógico de las entradas opto-acopladas y el de las salidas de control (relevadores o etapas de potencia), así como el valor de la señal generada por el convertidor D/A. Cuando se selecciona Estado general el programa muestra la ventana de la figura 4-7.

Opción Configuración.

Esta opción permite observar las direcciones asignadas a todos los elemento que forman los diferentes módulos de la tarjeta (Control, ADC, DAC, relevadores y optoacopladores), y el estado de las variables que determinan la operación del sistema. Al elegir la opción Configuración se activa la ventana mostrada en la figura 4-8.

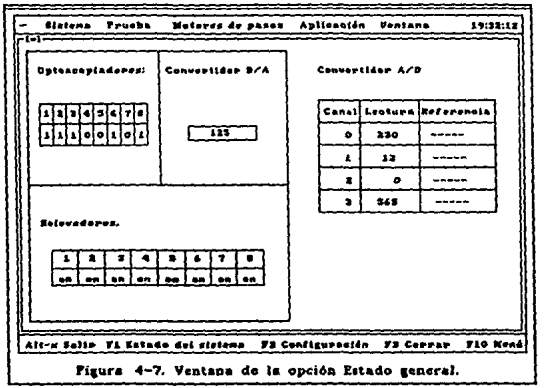


Figura 4-7. Ventana de la opción Estado general.

Opción Salir (Alt-x).

Al seleccionar esta opción se termina la ejecución del programa, regresando el control al sistema operativo. Esta opción no genera cuadros de dialogo ni ventanas.

Menú Prueba.

Por medio de este menú el programa proporciona las rutinas necesarias para verificar el buen funcionamiento de los elementos que integran la tarjeta. Al igual que el menú anterior, las opciones se listan verticalmente (figura 4-9) y para seleccionar alguna de ellas se utilizan las teclas FECHA ARRIBA y FLECHA ABAJO para mover el área resaltada hasta la opción elegida y se presiona la tecla INTRO, otra forma es presionar la tecla resaltada de la opción o bien (si se cuenta con el ratón) pulsar el ratón sobre dicha opción.

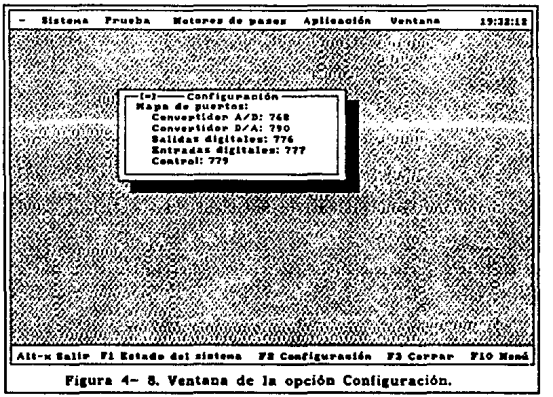


Figura 4- 8. Ventana de la opción Configuración.

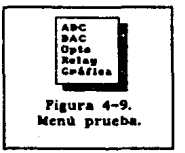


Figura 4-9.
Menú prueba.

Opción ADC.

Con esta opción se despliega una ventana (figura 4-10) que muestra el estado de las señales conectadas a los cuatro canales de entradas analógicas con que cuenta la tarjeta.

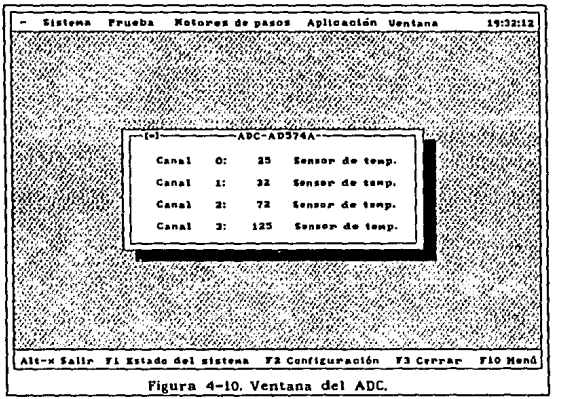


Figura 4-10. Ventana del ADC.

Opción DAC.

Al elegir esta opción el programa muestra el cuadro de dialogo de la figura 4-11; como se observar, con esta opción podemos enviar los datos que entrega el ADC, en cualquiera de sus cuatro canales, al convertidor D/A; para hacerlo basta presionar el número del canal que queremos conectar a entrada del DAC o bien pulsar el ratón sobre el botón asociado a dicho canal.

Opción Opto.

La ventana asociada al comando opto (mostrada en la figura 4-12) nos muestra el estado lógico de los 8 optoacopladores de la tarjeta. El estado de los optoacopladores nos indican las condiciones en que se encuentran las variables del proceso a controlar.

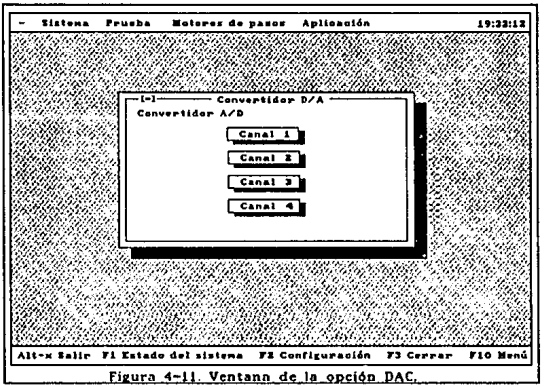


Figura 4-11. Ventana de la opción DAC.

Opción Relevadores.

Esta opción permite cambiar el estado de las señales de control asociadas a los relevadores o etapas de potencia. El cuadro de dialogo que se activa al seleccionar esta opción se muestra en la figura 4-13 y nos indica el estado de los ocho relevadores de la tarjeta y presenta un botón para cambiar el estado de cada relevador; así cuando se requiera activar o desactivar un relevador bastará con presionar el número asociado al relevador deseado, o bien pulsar el ratón (si se cuenta con él) en el botón correspondiente.

Opción Gráfica.

Con esta opción el programa despliega el oscilograma de una de las señales que se encuentran conectadas en los cuatro canales del convertidor analógico digital. La gráfica

mostrada no es en tiempo real debido a que la velocidad de muestreo es superior al despliegue de datos hecho por la computadora, por lo cual es necesario almacenar un conjunto de datos en memoria para desplegarlos posteriormente. La figura 4-14 muestra el ambiente asociado a esta opción teniendo dos modos de operación.

Figura 4-12. Ventana opción Optos.

Modo Continuo. En este modo de operación se realiza un muestreo de los datos necesarios para cubrir un barrido horizontal de la gráfica (200 muestras), después se suspende el muestreo para imprimir los datos en la pantalla mostrada en el monitor y una vez terminado se reanuda nuevamente el muestreo para desplegar otra serie de datos.

Modo de Grabación. En este modo se toma un número predeterminado de muestras,

guardándolas en memoria y suspendiendo el muestreo para desplegar los datos de memoria en el monitor de la computadora, permitiendo realizar un análisis de la señal (modificando las escalas horizontal y vertical).

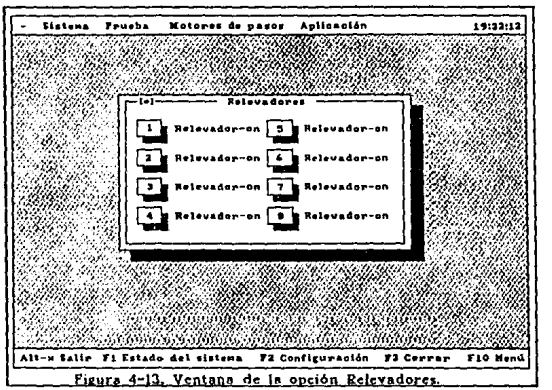


Figura 4-13. Ventana de la opción Relevadores.

En el modo de grabación se puede congelar la imagen del monitor, modificar la escala horizontal y vertical, para observar las señales adecuadamente. La teclas para controlar el modo de operación se muestran en pantalla.

Menú Motores de paso.

La operación adecuada del modulo de control de los motores de paso (cuando se encuentra conectado a la tarjeta) se puede verificar empleando los comandos de este menú. Los comandos

del menú Motores de pasos se muestran en la figura 4-15 y para seleccionar alguno de ellos se sigue el mismo procedimiento descrito para los menús previos.

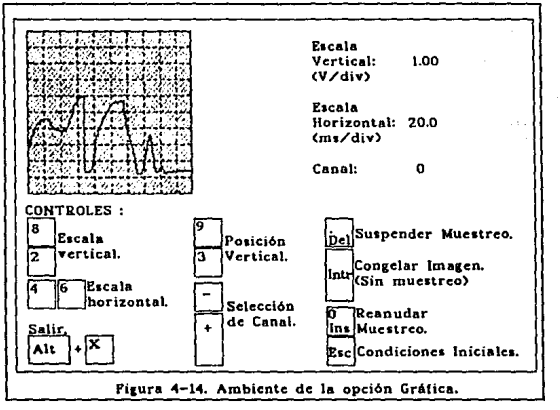


Figura 4-14. Ambiente de la opción Gráfica.

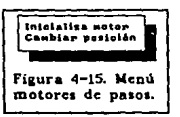


Figura 4-15. Menú motores de pasos.

Opción Inicializa motor.

Esta opción no activa ninguna ventana, ni cuadro de dialogo. Su función es colocar los

motores de pasos en la posición de referencia (coordenadas (0,0) del plano bidimensional que se puede explorar).

Opción Cambiar posición.

La opción Cambiar posición permite colocar los motores de pasos en una posición específica a una determinada velocidad. Para seleccionar la posición y la velocidad de cada motor se selecciona el motor deseado con las teclas 1 o 2 , ajustando la nueva posición con las teclas + y - (o pulsando el ratón sobre el botón correspondiente). Para colocar el motor en la nueva posición basta presionar la tecla resaltada del botón aceptar. Cuando se elige la opción Cambiar posición la computadora muestra en la pantalla la ventana de la figura 4-16, donde indica la posición actual de los motores, así como los botones necesarios para realizar los cambios de posición.

Menú Aplicación.

En este menú se implementan los comandos básicos correspondientes al proceso de aplicación descrito en el siguiente capítulo.

Como lo muestra la figura 4-17 los comandos que integran este menú son: Modo de operación, Superficie de exploración, Exploración y Despliegue de resultados. Los comandos se listan verticalmente y para seleccionarlos se utilizan las teclas FLECHA ARRIBA y FLECHA ABAJO, la tecla resaltada de cada opción o el ratón, siguiendo el procedimiento empleado en los menús anteriores.

Opción Modo de operación.

Al seleccionar esta opción se despliega el menú de la figura 4-18, en el cual muestra dos modos de exploración:

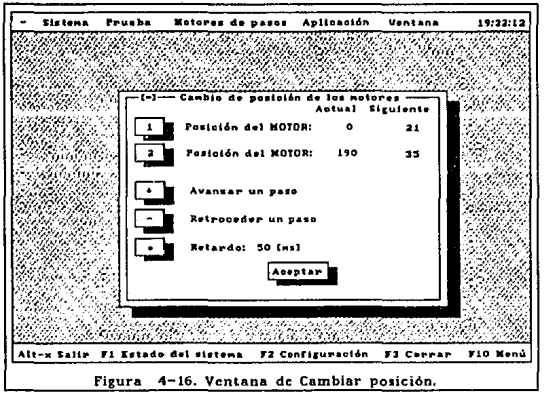


Figura 4-16. Ventana de Cambiar posición.

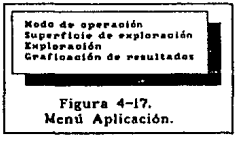


Figura 4-17.
Menú Aplicación.

-Continuo. Este modo de operación permite explorar una superficie predeterminada mostrando los resultados en el monitor directamente, una vez terminada una exploración se procede a realizar el muestreo la superficie nuevamente.

-Una sola exploración. En este modo se explora la superficie una vez, dejando los resultados

en un archivo, permitiendo analizar o desplegar los resultados en forma posterior.

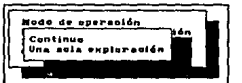


Figura 4-18.
Menú Modo de operación.

Opción Superficie de exploración.

Al seleccionar Superficie de exploración la computadora muestra una ventana (figura 4-19) que presenta los botones necesarios para ajustar dos esquinas opuestas de la superficie (rectangular) a explorar; también se puede ajustar la velocidad de exploración.

Opción Explorar.

El cuadro de dialogo mostrado correspondiente a esta opción muestra las coordenadas de las esquinas opuestas que definen la superficie de exploración (ajustadas previamente en la opción anterior) y las coordenadas correspondientes a la posición actual del motor. Antes de realizar la exploración se realiza un llamado a la rutina Inicializa motor colocándolo en la posición (0,0), después se coloca el motor el la esquina inicial de la superficie (por medio de la rutina Cambiar posición) y posteriormente se procede a realizar un barrido de la superficie almacenando los resultados obtenidos (de todos los puntos) en los archivos BINARIOS.CPI (palabras correspondientes al código -12 bits- entregado por ADC) y COLORES.CPI (código -0 a 16- correspondiente a un color determinado). La ventana que se activa al selecciona Explorar se muestra en la figura 4-20.

Despliegue de resultados.

Al seleccionar esta opción se activa el ambiente (modo gráfico) mostrado en la figura 4-21.

Este comando se encarga de desplegar en el monitor una figura que represente la superficie explorada, tomando los datos contenidos en el archivo COLORES.CPI almacenados en la rutina EXPLORACIÓN. Cada color desplegado corresponde a un valor predeterminado de la distancia comprendida entre el sensor y el objeto que causó el rebote.

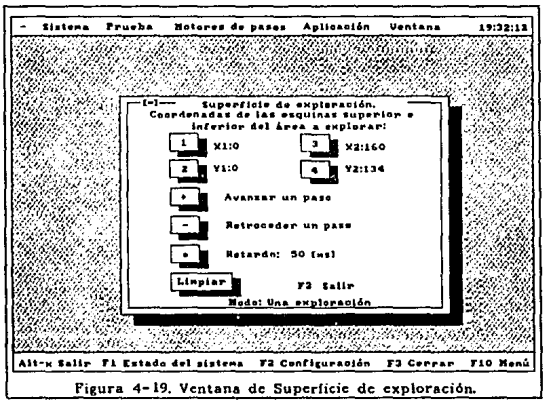


Figura 4-19. Ventana de Superficie de exploración.

Menú Ventana.

En este menú se encuentran los comandos necesarios para cerrar ventanas y para cambiarse de una a otra. Los comandos se listan verticalmente (al igual que los menús anteriores) como lo muestra la figura 4-22.

Opción Cerrar (F3).

Esta opción permite cerrar la ventana activa pasando el control a la ventana abierta previamente (si es que la hay).

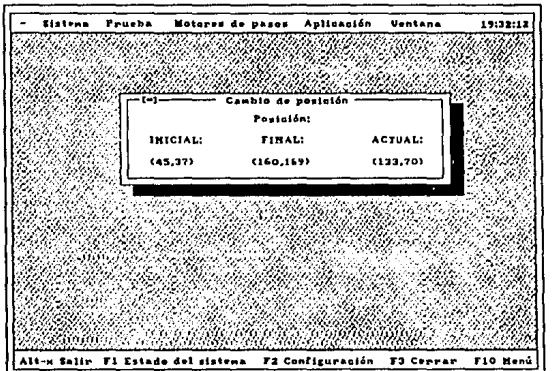


Figura 4-20. Ventana de la opción Exploración.

Opción Siguiente (F6).

Este comando permite observar el contenido de la ventana abierta después de la ventana activa.

Opción Anterior (F7).

Este comando permite pasar el control a la ventana previamente seleccionada.

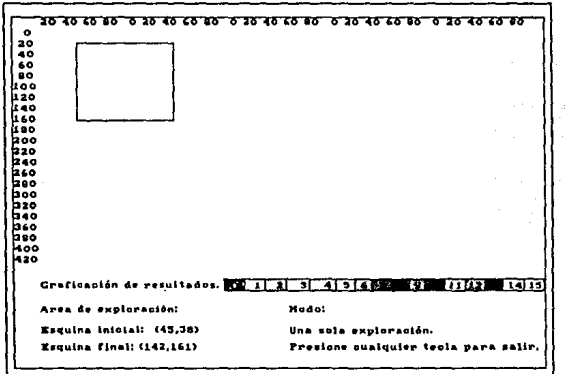


Figura 4-21. Ambiente de la opción Graficar resultados.

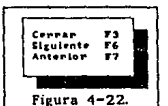


Figura 4-22.
Menú Ventana.

Para ejecutar estos comandos se pueden seleccionar con las teclas FLECHA ARRIBA Y FLECHA ABAJO (presionando la tecla Intro para aceptar la orden), con la tecla resaltada del comando, con la tecla F3, F6 y F7 (para cerrar, siguiente y anterior, respectivamente) o colocando el ratón sobre la opción y presionando el botón izquierdo.

El listado del programa (incluyendo las rutinas externas requeridas para su operación) se muestra en el apéndice A.

V. APLICACIÓN.

5.1 Diseño y construcción de una interfase para la adquisición de imágenes digitalizadas de señales ultrasónicas utilizando la tarjeta de control de procesos industriales.

El despliegue de imágenes se utiliza como una herramienta muy poderosa en el campo médico, así como también en pruebas no destructivas (*PND*). La presentación de imágenes médicas adquiridas con ayuda de ultrasonido presenta muchas ventajas potenciales sobre otras formas de despliegue en el campo médico debido a las propiedades no irradiantes del ultrasonido. Por otro lado, el uso de imágenes ultrasónicas en pruebas no destructivas permite detectar y evaluar fallas en diversos materiales en puntos más accesibles a mayores distancias de la pieza o estructura bajo prueba. Aunque en la mayoría de las pruebas no destructivas se manejan frecuencias en el rango de 1 a 15 MHz, existen varias personas involucradas en este campo (*PND*) interesadas en manejar frecuencias hasta de 50 MHz[12]. En los dos casos anteriores la generación y recepción del ultrasonido se basa en forma general en los mismos principios. La aplicación del ultrasonido en medicina tiende a presentar el mismo rango de frecuencias, de tal manera que los principios utilizados en el campo médico pueden beneficiar

al campo de las PND y viceversa.

La tarjeta diseñada y construida en esta tesis encuentra un campo de aplicación en proyectos relacionados con ultrasonido actualmente desarrollados en el Departamento de Electrónica y Automatización (*DEA*) del *IIMAS-UNAM*.

EL sistema está formado de un transductor (Transmisor/Receptor) y por tres módulos principales (entre los que se encuentra la tarjeta), tanto el transductor como los tres módulos se describen a continuación.

Transductor (transmisor/Receptor).

El tipo de transductor utilizado se muestra en la figura 5-1. El transductor cuenta con elementos piezoeléctricos que actúan como transmisor y receptor. También cuenta con un acoplamiento mecánico, aislamiento acústico y cubierta metálica. Su conexión con el módulo de excitación y recepción es por medio de un cable coaxial.

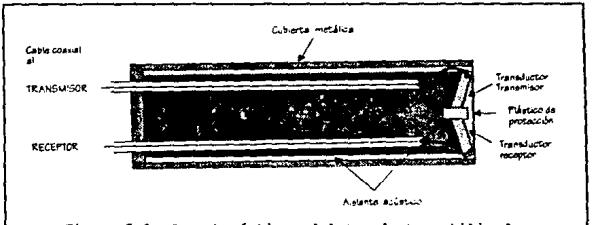


Figura 5-1. Características del tranductor utilizado.

Módulo de Excitación y Recepción.

El módulo de excitación y recepción que se muestra en la figura 5-2, tiene como finalidad acoplar el transductor con las otras etapas del sistema. En esta etapa se excita al transductor para que genere una señal, la cual rebota al presentarse un cambio de impedancia en el material bajo prueba, la señal recibida se amplifica, se demodula y se filtra de manera tal que quede condicionada para poder ser leída a través de un sistema de adquisición de datos.

Este módulo se divide en cuatro secciones:

Sección de relojes.

Esta sección tiene como función generar todas las señales de reloj utilizadas partiendo de un reloj maestro de 20Mhz y una serie de contadores para las diferentes frecuencias a utilizar.

Excitación.

Esta sección genera los pulsos de excitación al elemento piezoeléctrico de transmisión, toma como base una señal de la etapa de relojes y se controla el ancho de pulso con un monoestable, controlando de esta manera el ciclo de trabajo. El circuito cuenta con un elevador de voltaje de 5V dc a 120V dc, con el cual se excita al transductor, dicho pulso de 120V dc se corta impulsivamente con un SCR, con el fin de excitar al elemento piezoeléctrico y absorver energía para disminuir los ciclos de oscilación y de esta manera poder delimitar de forma sencilla el ciclo del pulso a medir.

Recepción.

Esta sección realiza cuatro funciones principales.

a) La primera es una etapa que habilita y deshabilita la entrada del receptor por medio de un interruptor analógico de alta velocidad, con este interruptor se deshabilita en la entrada la señal en el momento de la excitación al transmisor. Esta sección se realiza debido a que de esta manera es posible utilizar el circuito usando un solo transductor como transmisor y receptor.

b) La segunda parte es un circuito amplificador de RF de alta ganancia para recuperar la señal de eco del elemento receptor y obtener una señal con niveles fáciles de operar.

c) Una vez amplificada la señal se demodula y se filtra para obtener una señal de eco recibida.

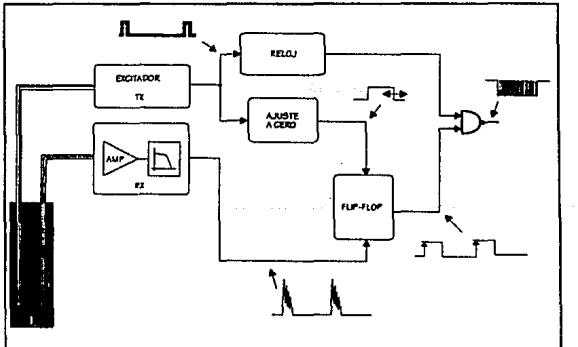


Figura 5-2. Diagrama de bloque del módulo de excitación y recepción.

d) La señal de A.F. se amplifica para tener una señal dentro de los rangos de 0-5V. Esta señal es capturada utilizando un sistema de adquisición de datos, para poder ser procesada posteriormente de diferentes formas.

Despliegue digital.

Esta sección tiene como finalidad generar una cuenta digital que proporcionó una lectura de la profundidad o distancia en que se generó el eco. Toma la señal que entrega el amplificador de A.F. y la señal del pulso de excitación, obteniendo un pulso cuyo ancho es proporcional a la distancia, este pulso generado se usa como ventana para realizar un número de cuentas con el reloj maestro, obteniéndose una cuenta digital que es medida por un dispositivo contador/manejador de *display* y se despliega en una pantalla de cristal líquido.

Como se mencionó anteriormente, la tarjeta de propósito general diseñada y construida en esta tesis ayuda en el soporte de proyectos de este tipo.

La idea principal del procesamiento es la de capturar e interpretar los datos digitales obtenidos por la tarjeta, provenientes del módulo de excitación y recepción, mediante un almacenamiento en memoria y una representación gráfica, en este caso, en un monitor de alta resolución (VGA 640X480 pixeles), haciendo un mapeo entre la posición específica del transductor, en el momento de la adquisición y la posición del pixel en el monitor VGA.

De esta manera la técnica utilizada en el procesamiento, y como primer intento, se puede resumir de la siguiente manera:

Del módulo de excitación y recepción se obtiene un pulso provocado por el primer rebote recibido, éste se cuantiza desde la excitación hasta la recepción y según el tiempo de duración

(con la ayuda de una tabla de conversión) se le asigna a un pixel específico en las coordenadas (X,Y) un color que es reflejo de la profundidad del punto explorado. Una representación de este proceso se muestra en la figura 5-3.

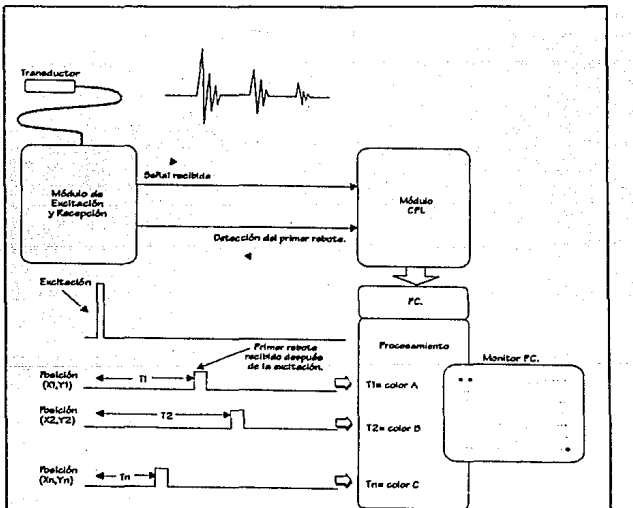


Figura 5-3. Algoritmo de procesamiento.

El diagrama de la figura 5-4 muestra en forma general el sistema, integrado por una tarjeta conectada internamente en una de las ranuras de expansión de la PC (diseñada y construida en este trabajo de tesis), una tarjeta para el módulo de excitación y recepción, el transductor ultrasónico utilizado, un acoplamiento acústico y una etapa de potencia para el manejo de los motores de pasos (construida en esta tesis), y así desde la PC contar con el control total de la posición del sensor en el plano bidimensional con una precisión menor a un milímetro para el desplazamiento del transductor.

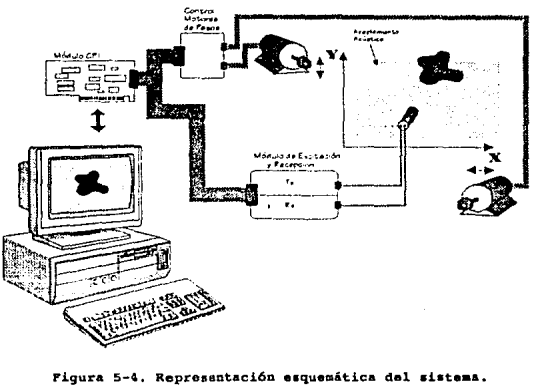


Figura 5-4. Representación esquemática del sistema.

Así se procede a realizar el muestreo de todo el objeto bajo prueba. Al mismo tiempo se va digitalizando la señal recibida por el módulo de excitación y recepción almacenándose todos los resultados en memoria o en disco, para utilizar posteriormente un procesamiento más

complejo.

5.2 Tarjeta de interfase para el desarrollo de la aplicación.

Para la implementación de la solución práctica presentada en este capítulo se diseñó y fabricó una tarjeta con los circuitos electrónicos requeridos por la aplicación.

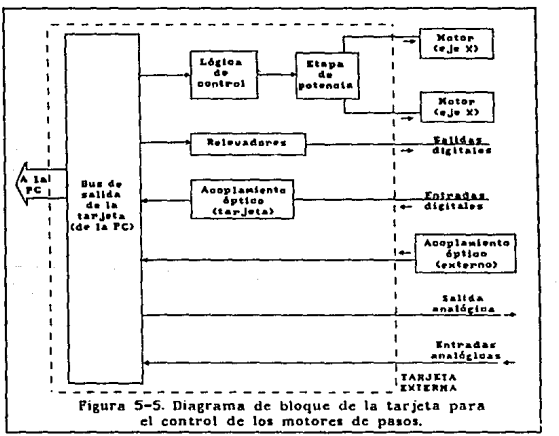


Figura 5-5. Diagrama de bloque de la tarjeta para el control de los motores de pasos.

Esta tarjeta se integra con los circuitos para el control de los motores de pasos, circuitos

para el monitoreo aislado del estado de las variables del proceso, salidas de potencia (o relevadores) con capacidad para manejar corrientes del orden de 1A, conectores para las entradas analógicas y un conector para la salida analógica.

La tarjeta proporciona las señales necesarias para colocar los motores en una posición particular en un plano determinado (movimiento en dos dimensiones) proporcionando las etepas de acoplamiento necesarias (lógica de control y etapa de potencia).

El diagrama de bloques de la tarjeta se puede observar en la figura 5-5.

5.3 Motores de paso a paso.

Un motor paso a paso es un elemento de acción incremental electromagnético que convierte entradas de impulsos digitales en movimientos de salida de un eje analógico. En un motor paso a paso en movimiento, el eje de salida del motor gira incrementos iguales en respuesta a un tren de impulsos de entradas. Cuando está controlado adecuadamente, los pasos de salida del motor son iguales (en número) al número de los impulsos de entrada. Debido a que los sistemas de control modernos presentan a menudo movimientos de tipo incremental, los motores paso a paso se han convertido en elementos de acción importantes en los últimos años. Por ejemplo, podemos encontrar control de movimiento incremental en todos los tipos de equipo periférico de ordenadores, tales como impresoras, armarios de cintas, conductores de cabezales, mecanismos de acceso de memoria, asi como en una gran variedad de máquinas, herramientas y sistemas de control de procesos.

La figura 5-6 muestra la aplicación de un motor paso a paso en el mecanismo de conducción de papel de una impresora. En este caso el motor está acoplado directamente a un rodillo portapapel, de manera que el papel es conducido un cierto incremento cada vez. La resolución

típica de los motores paso a paso disponibles comercialmente abarca gamas desde pocos pasos por revolución hasta 800 pasos por revolución, e incluso superiores.

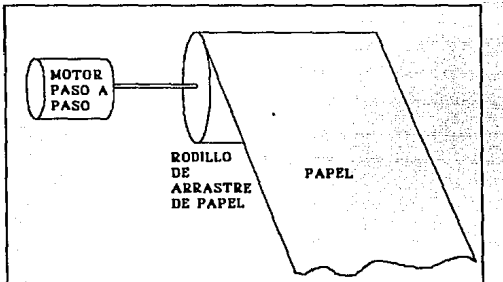


Figura 5-6. Mecanismo de accionamiento del papel usando un motor paso a paso.

Existen motores paso a paso de una gran variedad de tipos, según su principio de funcionamiento. Los dos tipos más comunes de motor paso a paso son el tipo de reluctancia variable y el tipo de imán permanente. El analisis matemático de estos motores es bastante complejo, ya que las características del motor son fuertemente no lineales. A diferencia de los motores de c.d. y los motores de inducción, los modelos linealizados de un motor paso a paso son normalmente poco realistas.

El motor paso a paso de reluctancia variable tiene un estator devanado y un rotor no exitado. El motor puede ser del tipo de una sola unidad o de unidades múltiples. En la versión de unidades múltiples, el estator y el rotor constan de tres o más conjuntos separados de dientes.

Los conjuntos separados de dientes sobre el rotor, normalmente laminados, están montados sobre el mismo eje. Los dientes en todas partes del rotor están perfectamente alineados. La figura 5-7 muestra un modelo típico rotor estator de un motor que tiene tres partes separadas sobre el rotor, o sea un motor trifásico.

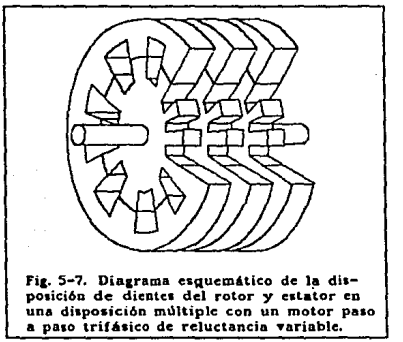


Fig. 5-7. Diagrama esquemático de la disposición de dientes del rotor y estator en una disposición múltiple con un motor paso a paso trifásico de reluctancia variable.

El motor paso a paso de reluctancia variable debe tener como mínimo tres fases para que tenga control direccional. Los tres juegos de dientes de rotor son magnéticamente independientes y están montados en un eje que está soportado por cojinetes. Dispuesto alrededor de cada sección del rotor existe un núcleo del estator con devanados. Los devanados no se muestran en la figura 5-7. La figura 5-8 es un diagrama esquemático de los devanados del estator. En la Figura 5-9 puede verse la vista frontal del estator de una fase y el rotor en un motor real. En este caso el motor se ve en una posición tal que sus dientes están alineados con los de la respectiva fase del estator.

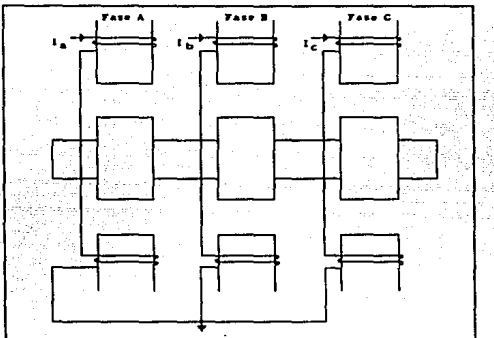


Figura 5-8. Diagrama esquemático de un motor paso a paso de reluctancia variable trifásico de unidades múltiples.

El rotor y el estator tienen el mismo número de dientes, lo que significa que el paso de dientes sobre el estator y el rotor es el mismo. Para hacer que el motor gire, las secciones de estator del motor trifásico están indexadas una tercera parte de un paso de diente, en el mismo sentido. La figura 5-10 muestra esta dirección para un rotor de 10 dientes. Por consiguiente, el diente de una fase del está desplazado 12 grados respecto a la fase del estator. Aquí los dientes de la fase C del estator puede verse alineados con los correspondientes dientes del rotor. Los dientes de la fase A del estator estan desplazados 12 grados en el sentido del reloj respecto a los de la fase C. Los dientes de la fase B del estator están desplazados 12 grados en el sentido del reloj respecto a los de la fase A o 12 grados en sentido contrario al reloj respecto a los de la fase C. Es facil ver que es necesario un mínimo de tres fases para dar control de sentido. En general, también son normales los motores de 4 y 5 fases, y se comercializan motores de hasta

8 fases. Para un motor de n fases, los dientes del estator deben estar desfasados 1/n de paso de diente de una sección a otra.

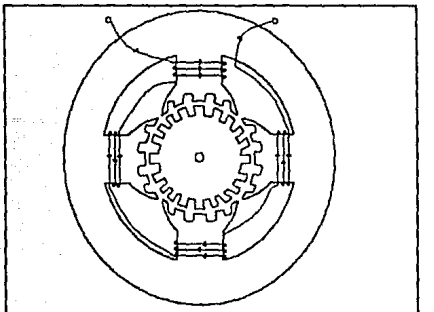


Fig 5-9. Vista frontal del estator de una fase de un motor de paso a paso de reluctancia variable y unidad múltiple.

El principio de funcionamiento del motor paso a paso de reluctancia variable es directo. Supongamos que a una fase cualquiera del devanado se le aplique una señal de c.d. La fuerza magnetomotriz ajustará la posicion del rotor de manera que los dientes de la sección del rotor bajo la fase excitada está alineado en oposición con los dientes de la fase excitada del estator. Esta es la posición de mínima reluctancia y el motor está en equilibrio estable.

Si se da tensión a la fase C en la figura 5-10, el rotor se posicionaría (en régimen permanente) como se muestra: También puede verse en la misma figura que si la señal de c.d. se aplica a la fase A, el rotor girará 12 grados en el sentido del reloj, y los dientes del rotor se

alinearán en oposición con los dientes de la fase A del estator. Continuando de la misma manera, la secuencia de entrada CABCAB hará girar el motor en el sentido del reloj en pasos de 12 grados.

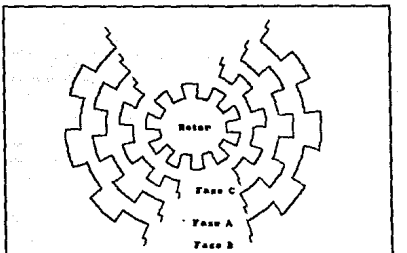


Figura 5-10. Disposiciones de dientes de rotor y estator de un motor de paso a paso trifásico de reluctancia variable de unidades múltiples.

Invirtiendo la secuencia de entrada se invertirá el sentido de rotación. O sea, la secuencia de entrada CBACB hará girar el motor en sentido contrario del reloj en pasos de 12 grados.

La curva en régimen permanente de cada fase es aproximadamente la que muestra la figura 5-11. La línea 0 grados representa el eje de cualquier diente de la fase del estator a la que se ha aplicado tensión. El eje del diente del rotor más cercano estará situado dentro de 18 grados a cualquier lado de esta línea. El par de arranque correspondiente ejercido cuando se aplica también a esta fase puede verse en la figura 5-11. Las flechas marcan el sentido de movimiento del rotor.

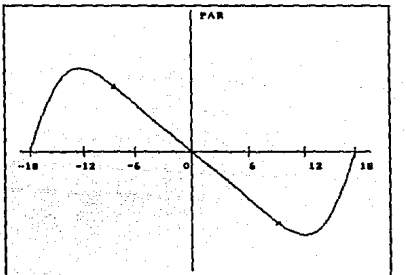


Figura 5-11. Curva de par para una fase de un motor paso a paso.

Supongamos que los desplazamientos angulares positivos representan movimientos en el sentido del reloj. Supongamos también que se haya aplicado tensión a la fase C durante un cierto tiempo. Esto significa que la condición inicial del rotor será la que muestra la figura 5-10. Si ahora se aplica tensión a la fase A y la figura 5-11 representa la variación de par de la fase A, la posicion inicial de los dientes del rotor estará en -12 grados. Asi, al dar tension a la fase A, el rotor se ajustará finalmente después de algunas oscilaciones, suponiendo que la inercia y la fricción sean tales que no exista rebase más allá del punto de 18 grados.

Puede observarse que la posición ± 18 grados también representa un punto de equilibrio. Esto es debido a que en esta posición el par de flexión es cero. Sin embargo es una posición de equilibrio inestable ya que la más ligera variación del eje respecto a la posición hará que el motor pase a 0 grados.

Si al aplicar tensión a una fase ocurre que el rotor está situado exactamente en el punto ± 18 grados permanecerá teóricamente en él. Sin embargo, en la práctica siempre existirán algunas imperfecciones mecánicas de construcción, y la asimetría resultantemente evitará cualquier bloqueo en el punto inestable.

5.4 Módulo de control de los Motores paso a paso.

El módulo para el manejo de los motores paso a paso se divide en los siguientes bloques (figura 5-12):

1. Lógica de control.
2. Etapa de potencia.

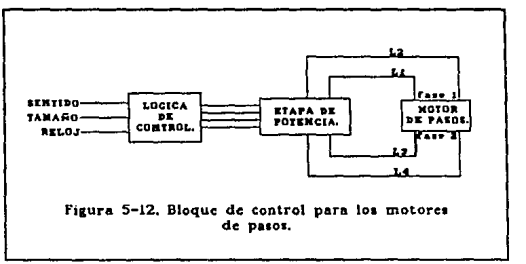


Figura 5-12. Bloque de control para los motores de pasos.

Lógica de control.

Este bloque se implementa con el circuito integrado MC3479 que recibe 3 líneas de control (proporcionadas por la tarjeta) que indican el modo de operación de los motores (paso completo ó medio paso), la dirección de giro y la señal para avanzar un paso (reloj).

Este bloque proporciona al motor la secuencia adecuada para producir un movimiento angular discreto (paso) en dos modos de operación: paso completo y medio paso. Las tablas de la figura 5-13 muestran las secuencias proporcionadas al motor en los dos modos de operación[13].

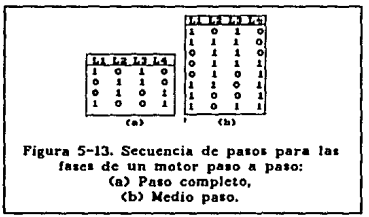


Figura 5-13. Secuencia de pasos para las fases de un motor paso a paso: (a) Paso completo, (b) Medio paso.

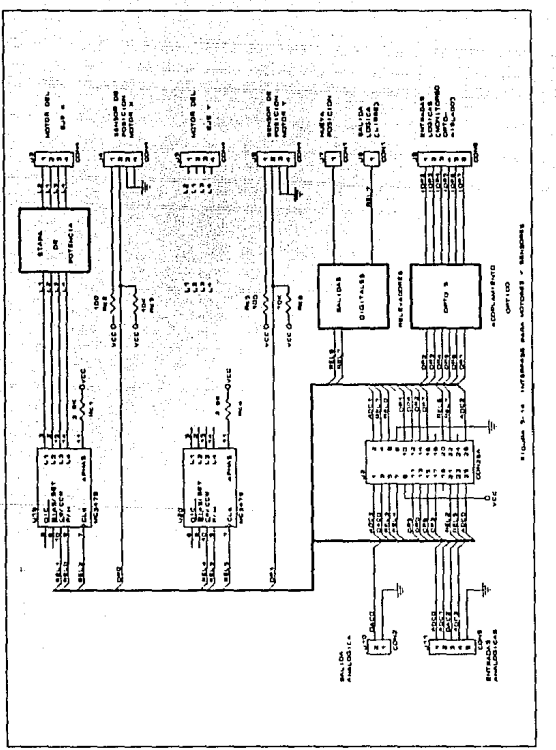
Etapa de potencia.

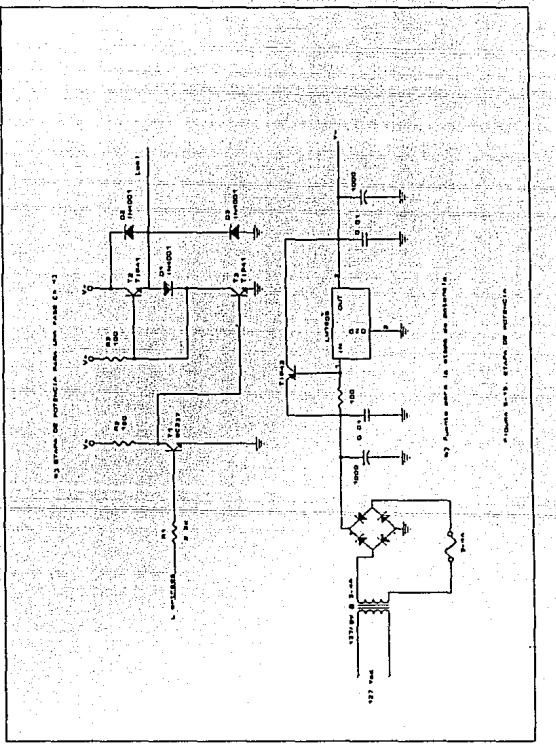
La computadora, empleando la tarjeta diseñada en este trabajo de tesis, proporciona las señales de control necesarias para colocar el transductor en una posición específica; las señales proporcionadas por el circuito MC3479 no son capaces de proporcionar la corriente que manejan los motores utilizados por el sistema implementado, por lo cual fué necesario implementar una etapa de acoplamiento que maneje la corriente requerida por los motores. La etapa de

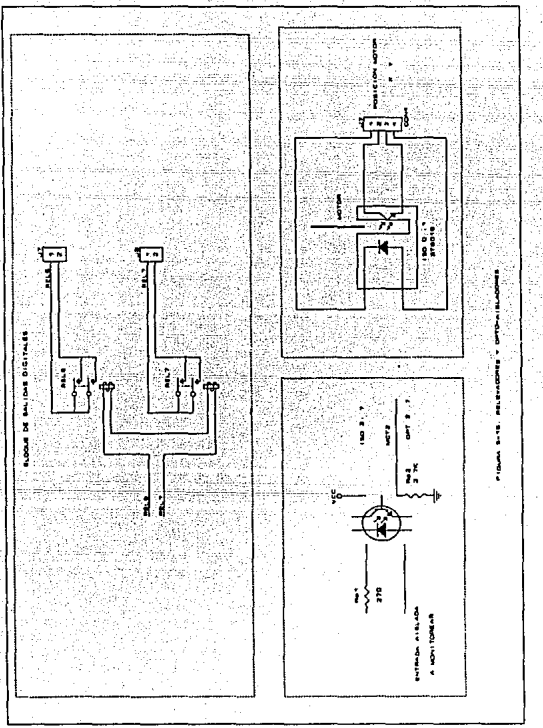
acoplamiento consta de un arreglo de transistores en configuración *Push-Pull* (un transistor de señal pequeña y dos de potencia) con capacidad para proporcionar una corriente superior a 1A.

El diagrama electrónico de la tarjeta de interfase para implementar la aplicación se muestra en las figuras 5-14, 5-15 y 5-16.

La tarjeta de interfase se diseño en el paquete de computadora SMART WORK siguiendo un procedimiento similar al empleado en la diseño de las tarjetas anteriores. Los diagramas de la tarjeta (lado de soldadura, componentes y distribución de los mismos) se muestra en las figuras 5-17, 5-18 y 5-19.






BLOQUE DE SALIDAS DIGITALES
RELE 6
RELE 7
RELE 8
RELE 9
POSICION MOTOR
MOTOR
FIGURA 5-15 RELEVADORES Y OPTO-AISLADORES

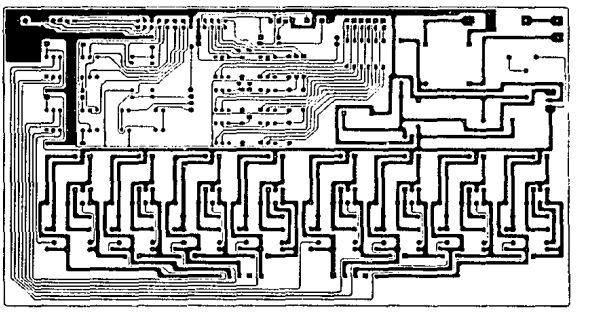


Figura 5-17. Circuito impreso (lado de soldadura).

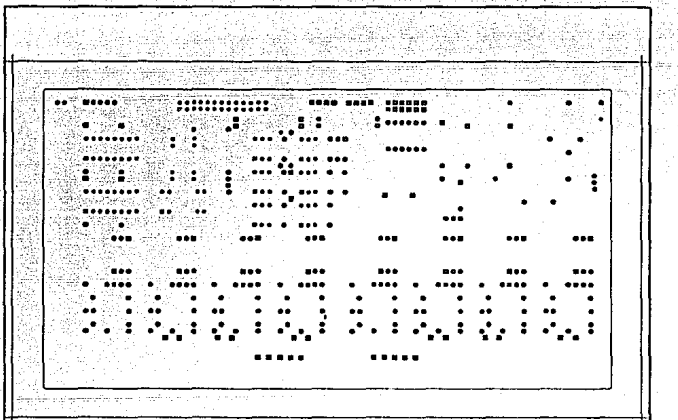


Figura 5-18. Circuito impreso (lado de componentes).

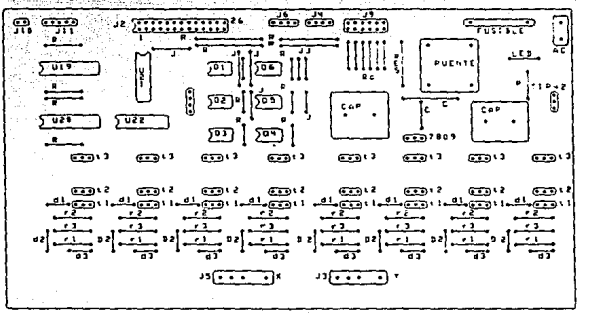


Figura 5-19. Circuito impreso (distribución de componentes).

VI. CONCLUSIONES.

El objetivo de esta tesis consistió en el diseño y construcción de una tarjeta con capacidad para monitorear y controlar un proceso industrial o de laboratorio, con la ayuda de una computadora personal del tipo IBM PC/XT/AT.

Al terminar este trabajo podemos resumir y concluir lo siguiente:

- La tarjeta se elaboró para un proceso industrial en general utilizando a la computadora como herramienta, para no construir un sistema completo.

- Al final se contemplo una aplicación en la adquisición de imágenes digitalizadas de señales ultrasónicas.

- Además de la solución práctica presentada en este trabajo se puede dar solución a problemas en el campo de la instrumentación electrónica, como son la adquisición, procesamiento y graficación de datos de variables físicas (temperatura, presión, velocidad de fluidos, humedad).

- Los circuitos de las tarjetas fabricadas se diseñaron en el paquete de computadora SMART WORK.

- Los circuitos integrados y dispositivos electrónicos utilizados se encuentran disponibles en el

mercado nacional.

- El programa esta hecho en el lenguajes de alto nivel TURBO PASCAL versión 6.0, empleando la libreria de procedimientos proporcionados por TURBO VISION.

- El programa de aplicación presenta un entorno integrado por una serie de menús que hacen su manejo sencillo, soportando el uso del ratón y múltiples ventanas solapadas que permiten observar el estado de distintos módulos del proceso industrial.

- La información que se obtiene del convertidor analógico digital se almacena en disco para su procesamiento posterior (código ASCII).

- Posibilidad de cambio en la estrategia de control con la simple modificación del programa de aplicación.

- La frecuencia de muestreo se establece considerando la velocidad de conversión del ADC (25μs) y la computadora empleada. La tabla siguiente indica la frecuencia de muestreo empleada para diferentes computadoras; para seleccionar la frecuencia de muestreo se considera el tiempo de muestreo y el tiempo que dura la rutina de atención de interrupción en la computadora empleada (100μs en una computadora PC/XT con velocidad de 8MHz, 50μs utilizando una computadora 80286 con velocidad de 16 MHz, etc).

Computadora	Velocidad [Mhz]	Frecuencia de muestreo [KHz]
IBM PC/XT/AT	8	10
IBM 80286	16	20
IBM 80386	33	~40

- La computadora genera mensajes y señales de control con objeto de que el funcionamiento del proceso sea el adecuado.

- Se cuenta con etapas de potencia para el manejo de actuadores que requieran corrientes de 1 A (en las salidas digitales).

- Graficación de las señales analógicas conectadas al ADC.

- Graficación con asignación de colores (cada color representa un intervalo de voltajes) para las señales analógicas conectadas al ADC.

- En aplicaciones en las cuales se tienen señales que cambian lentamente se tiene posibilidad de procesamiento en tiempo real.

- La frecuencia de las señales que se pueden generar con el convertidor digital analógico esta limitada por el tiempo de conversión del DAC ($1\mu s$), la velocidad de procesamiento y el número de datos (por periódo) utilizados para generar la señal. Por ejemplo si se tiene una señal generada con 200 puntos su frecuencia máxima sería de 5 KHz ($f=1/T$, donde "T" se calcula multiplicando el número de datos -utilizados para generar la señal- por el tiempo de conversión) despreciando el tiempo de procesamiento de la computadora.

- La resolución de las gráficas mostradas por el programa de aplicación esta limitada por el tipo de monitor empleado (VGA, CGA, etc), sin embargo los datos se almacenan en disco conservando la resolución del convertidor analógico digital (12 bits), lo cual permite transferir ese archivo a otro dispositivo de despliegue de información de mayor resolución (osciloscopio digital, graficador, etc.).

Finalmente, una vez concluido este trabajo, podemos decir que se tiene una tarjeta de propósito general que opera satisfactoriamente (como se comprobó en la aplicación implementada) sirviendo para el desarrollo de proyectos dentro del cualquier área donde se requiera el monitoreo de señales analógicas (que representen alguna variable física) y la vigilancia del correcto funcionamiento de un proceso, aplicando las señales de control requeridas, cuando esto sea necesario.

APÉNDICE A.

PROGRAMA DE APLICACIÓN.

En este apéndice se presente el listado del programa principal incluyendo las unidades elaboradas para su funcionamiento.

Para la elaboración de este programa se modificaron estructuras implementadas en la librería de TURBO VISION (para manejo de ventanas, ratón, cuadros de dialogo, etc.) y se elaboraron las unidades para comprobar el funcionamiento de la tarjeta diseñada y construida en este trabajo de tesis.

Programa Principal.

```
{*************************************************************
*                    Programa principal                     *
*                 Control de procesos industriales          *
*                        I. I. M. A. S.                     *
*                          1992-1993                        *
*************************************************************

Se utilizan las siguientes unidades:
-Objects
-App
```

-Drivers
-Dialogs.
-Views
-Menus
-Gadgets
-Converci: En esta unidad se implementan las rutinas necesarias para mostrar en el monitor la ventana que despliega los datos del convertidor analógico digital.
-interrup: Esta unidad contiene las rutinas necesarias para el manejo de interrupciones; la rutina para modificar la tabla de vectores de interrupciones de la computadora, la rutina para reinstalar la tabla de vectores original y el manejador de interrupciones.
-Aplicaci: Las rutinas necesarias para implementar la aplicación, descrita en el capitulo V, se encuentran en esta unidad.
-Grafica: En esta unidad se presentan las rutinas necesarias para desplegar información en modo gráfico (osciloscopio y resultados obtenidos al explorar una superficie en la aplicación).
**********}

Program tesis;
{**********Directivas de compilación**********}
{$X+,S-}{$m 16384,8192,655360}
{**********Decalaración de objetos utilizados**********}
Uses
Objects,Drivers,dialogs,Views,Menus,App,converci,gadgets,intp,graphap,crt,graph,msgbox,stddlg,dos,dacfunc,datos,grafica,aplicaci;
{**********Declaración de estructuras empleadas**********}
type

```
poptoview = ^toptoview;
toptoview = object(tview)
  col:word;
  constructor init(bounds:trect);
  procedure draw;virtual; procedure update;
end;
poptowindow = ^toptowindow;
toptowindow = object (twindow)
  popt:poptoview;
  constructor init(popto:poptoview;x1,y1,x2,y2:byte);
end;

pcontrolview=^tcontrolview;
tcontrolview=object(tview)
  constructor init(bounds:trect);
  procedure draw;virtual;procedure update;
end;

pcambview = ^tcambview;
tcambview = object(tview)
  col:word;
  constructor init(bounds:trect);
```

```
  procedure draw;virtual;procedure update;
end;
pcambwindow = ^tcambwindow;
tcambwindow = object (twindow)
  pcam:pcambview;
  constructor init(pcamb:pcambview;x1,y1,x2,y2:byte);
end;

Pcontrol = ^Tcontrol;
Tcontrol = object(twindow)
  control:pcontrolview;
  constructor init(controlvi:pcontrolview);
end;

{*****************Objeto Principal TSISTEMA*********************}
tsistema = object(Tapplication)
  canal:word;
  t:padcview;
  opcion,banderaade:byte;
  clock:pclockview;opto:poptoview;
  controlview:pcontrolview;
  AppDriver: Integer;AppMode: Integer;
  BGIPath: PString;
  constructor init;
  procedure Initmenubar;virtual;
  procedure Initstatusline;virtual;
  procedure handleevent(var event:tevent);virtual;
  procedure newdialog;
  procedure informacion;procedure idle;virtual;
end;

pdemodialog = ^TDemoDialog;
TDemoDialog = object(Tdialog)
end;
{*************************************************************}
{***************Objetos secundarios**************************}
preleview = ^treleview;
treleview = object(tview)
  estado:string[3];
  relenum:byte;
  constructor init(bounds:trect;s:string;i:byte);
  procedure draw;virtual;
  function getestado:string;procedure setestado(s:string);
  function getrelenum:byte;procedure setrelenum(i:byte);
end;
prelevadores = ^trelevadores;
trelevadores = object(tdialog)
```

```
  rele:array[1..8] of preleview;
  constructor init;
  procedure handleevent(var event:tevent);virtual;
end;

pmotorview=^tmotorview;
tmotorview=object(tview)
  mensaje1:string;
  constructor init(bounds:trect;s:string);
  procedure draw;virtual;procedure setmensaje1(i:integer);
end;
pmotores=^tmotores;
tmotores=object(tdialog)
  motor:array[1..4] of pmotorview;
  cambio:pcambview;motoractivo:byte;
  constructor init;
  procedure handleevent(var event:tevent);virtual;
end;

psuperview=^tsuperview;
tsuperview=object(tview)
  mensaje1:string;
  constructor init(bounds:trect;s:string);
  procedure draw;virtual;
  procedure setmensaje1(i:integer);
end;
psuperfi=^tsuperfi;
tsuperfi=object(tdialog)
  super:array[1..7] of psuperview;
  superactivo:byte;
  constructor init;
  procedure handleevent(var event:tevent);virtual;
end;
pdacview=^tdacview;
tdacview=object(tview)
  mensaje:string;
  constructor init(bounds:trect;s:string);
  procedure draw;virtual;
end;
pdacwindow=^tdacwindow;
tdacwindow=object(tdialog)
  dac:pdacview;
  constructor init;
  procedure handleevent(var event:tevent);virtual;
end;
```

```
{*****Implementacción de las rutinas de los objetos empleados**}
{***********************Menú sistema***************************}
{*****************opción Estado general**********************}
constructor tcontrol.init(controlvi:pcontrolview);
var
  r,bound:trect;
begin
  R.assign(0,0,80,23);
  twindow.init(R, 'Estado General del sistema', 0 );
  growmode :=0;palette :=0;
  control:=controlvi;insert(control);
end;
Constructor tcontrolview.init(bounds:trect);
begin
  Tview.init(bounds);options:=options or ofselectable;
  eventmask:= eventmask or evmouseauto;drawview;
end;
procedure tcontrolview.draw;
const
  width=77;
var
  s:array[0..21] of string;
  b:array[1..width] of word;
  color,i,j,d, boldcolor, specialcolor:byte;
  dacs:string[5];
  adcs:array[0..7] of string[5];
begin
  color:= getcolor(6);boldcolor:= getcolor(7);
  str(datodac:5,dacs);
  for j:=0 to 3 do str(datoadcw[j]:5,adcs[j]);
  {******123456789012345678901234567890123456789012345678901234567890123456789012345678
  *******        10        20        30        40        50        60        70**}
  s[0]:=' ┌──────────────────────────────────────────┬──────────────────────────────┐
             ';
  s[1]:=' │              │              │     Convertidor A/D.               ';
   s[2]:= '   │     Optoacopladores:       │  Convertidor  D/A   │
┌──────────┬──────────────────┬──────────────────┐
                                                                  ';
  s[3]:=' │              │              │     │ Canal │ Lectura   │ Referencia │      ';
  s[4]:=' │    ┌──┬──┬──┬──┬──┬──┬──┬──┐    │                                 │
├──────────┼──────────────────┼──────────────────┤';
  s[5]:=' │  │1│2│3│4│5│6│7│8│   │          │  │ 0 │    '+adcs[0]+'  │  200   │';
  s[6]:=' │    ├──┼──┼──┼──┼──┼──┼──┼──┤    │  ┌────────────────────┐   │
├──────────┼──────────────────┼──────────────────┤';
  s[7]:=' │  │'; d:=datooptos;
  for i:=1 to 8 do
  begin
  j:= d mod 2;
```

```
if j=1 then s[7]:=s[7]+'1 | ' else s[7]:=s[7]+'0 | ';d:= d div 2;
end;
s[7]:=s[7]+'  |  |  '+dacs+'  |  |  |  1  |  '+adcs[1]+'  |  ——  |';
s[8]:='  |  └──┴──┴──┴──┴──┴──┴──┘  |  └─────────────┘  |
├─────┼──────────┼──────────┤';
s[9]:='  |        |        |  |  2  |  '+adcs[2]+'  |  ——  |';
s[10]:='  ├──────────────────────┼──────────────────────┤
├─────┼──────────┼──────────┤';
s[11]:='  |        |  |  3  |  '+adcs[3]+'  |  ——  |';
s[12]:='  |   Relevadores.                                |
└─────┴──────────┴──────────┘';  s[13]:='  |
  |                    ';
s[14]:='  |  ┌──┬──┬──┬──┬──┬──┬──┬──┐  |
 ';
s[15]:='  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |       ';  s[16]:='  |
├────┼────┼────┼────┼────┼────┼────┼────┤  |      ';
s[17]:='  |  | ';d:=datorelay;
for i:=1 to 8 do
begin
  j:= d mod 2;
  if j=1 then s[17]:=s[17]+'on  | ' else s[17]:=s[17]+'off | ';d:= d div 2;
end;
s[17]:=s[17]+'        |        ';
s[18]:='  |  └──┴──┴──┴──┴──┴──┴──┴──┘  |
 ';
s[19]:='  |                |                ';
s[20]:='  └──────────────────────────────────────┘
            ';
for i:=0 to 21 do
begin
  movechar(b,' ',color,width);
  movestr(b,s[i],color);writeline(0,i,width,1,b);
end;
end;
procedure tcontrolview.update;
begin
  drawview;
end;
{*************************Menú Prueba***************************}
{*************************opción DAC***************************}
constructor tdacwindow.init;
var
  r:trect;ee:pview;s:string;
procedure escribe(a,b,c,d,i:byte;s:string);
var
  r:trect;ee:pview;
begin
```

```
  r.assign(a,b,c,d);
  ee:=new(pbutton,init(r,s,cmfuncion1-1+i,bfnormal));
  insert(ee);
end;
begin
  r.assign(18,4,62,18);Tdialog.init(r,'Convertidor D/A');
  r.assign(2,1,27,2);dac:=new(pdacview,init(r,'Conectar Convertidor A/D:'));
  insert(dac);
  escribe(17,2,28,4,1,'Canal ~1~');escribe(17,5,28,7,2,'Canal ~2~');
  escribe(17,8,28,10,3,'Canal ~3~');escribe(17,11,28,13,4,'Canal ~4~');
end;
procedure tdacwindow.handleevent(var event:tevent);
begin
   tdialog.handleevent(event);
   if Event.what=evCommand then
   begin
      CASE event.command of
        cmfuncion1:fundac(0);
        cmfuncion2:fundac(1);
        cmfuncion3:fundac(2);
        cmfuncion4:fundac(7);
        else
          exit;
      end;
      clearevent(event);
   end;
end;
constructor tdacview.init(bounds:trect;s:string);
begin
   Tview.init(bounds);mensaje:=s;
end;
procedure tdacview.draw;
var
   color:byte;i:integer;b:tdrawbuffer;
begin
   color:=getcolor(1);movechar(B,' ',color,size.x);
   movestr(b,mensaje,color);writebuf(0,0,size.x,1,b);
end;
{**********************************************************************
Objeto: Ventana del optoacoplador
**********************************************************************}
constructor toptowindow.init(popto:poptoview;x1,y1,x2,y2:byte);
var
   R:trect;
begin
   R.assign(x1,y1,x2,y2);twindow.init(R, 'Optoacopladores', 0 );
   growmode :=0;palette :=2;popt:=popto;insert(popt);
```

```
end;
{**********************************************************************
 Objeto: Area de datos del optoacoplador
**********************************************************************}
constructor toptoview.init(bounds:trect);
var
  H:word;
begin
  Tview.init(bounds);
  options:=options or ofselectable;
  eventmask:= eventmask or evmouseauto;
  col:=6;drawview;
end;
procedure Toptoview.draw;
const
  width=40;
var
  s:string;b:array[1..width] of word;
  color,i,j,d, boldcolor, specialcolor:byte;
begin
  color:= getcolor(col);boldcolor:= getcolor(7);
  s:='Optoacoplador:  1  2  3  4  5  6  7  8';
  movechar(b,' ',color,width);movestr(b,s,color);
  writeline(0,0,width,1,b); movechar(b,' ',color,width);
  s:='                                    ';
  movestr(b,s,color);writeline(0,1,width,1,b);
  movechar(b,' ',color,width);s:='Estado lógico:';
  d:=datooptos;
  for i:=0 to 8 do
  begin
    j:= d mod 2;if j=1 then s:=s+'  1' else s:=s+'  0';
    d:= d div 2;
  end;
  movestr(b,s,color);writeline(0,2,width,1,b);
end;
procedure toptoview.update;
begin
  drawview;
end;
{**********************************************************************
Objeto: Ventana del cambio de posición
**********************************************************************}
constructor tcambwindow.init(pcamb:pcambview;x1,y1,x2,y2:byte);
var
  R:trect;
begin
  R.assign(x1,y1,x2,y2);twindow.init(R, 'Cambio de posición', 0 );
```

```
  growmode := 0;palette := 2;pcam: = pcamb;insert(pcam);
end;
{**********************************************************************
Objeto: Area de datos del cambio de posición
**********************************************************************}
constructor tcambview.init(bounds:trect);
var
  H:word;
begin
  Tview.init(bounds);options: = options or ofselectable;
  eventmask: = eventmask or evmouseauto;col: = 6;drawview;
end;
procedure Tcambview.draw;
const
  width = 40;
var
  S,S1,S2,S3,S4,S5,S6:string;b:array[1..width] of word;
  color,i,j,d, boldcolor, specialcolor:byte;
begin
  color: = getcolor(col);boldcolor: = getcolor(7);
  s: = '              Posición:';movechar(b,' ',color,width);
  movestr(b,s,color);writeline(0,0,width,1,b);
  movechar(b,' ',color,width);s: = ' INICIAL:      FINAL:      ACTUAL:';
  movestr(b,s,color);writeline(0,2,width,1,b);movechar(b,' ',color,width);
  s: = '                                    '; movestr(b,s,color);
  writeline(0,1,width,1,b);movechar(b,' ',color,width);
  str(x1:3,s1);str(y1:3,s2);str(x2:3,s3);str(y2:3,s4);
  str(xm:3,s5);str(ym:3,s6);
  s: = '('+s1+','+s2+')    ('+s3+','+s4+')    ('+s5+','+s6+')';
  movestr(b,s,color);writeline(0,3,width,1,b);
end;
procedure tcambview.update;
begin
  drawview;
end;
{**********************************************************************
 Objeto: Ventana de los relevadores
**********************************************************************}
constructor trelevadores.init;
var r:trect;dd:integer;
procedure escribe(a,b,c,d,e,f,g,h,i,j,l:byte);
var
  r:trect;ee:pview;s:string;jj:integer;
begin
  r.assign(a,b,c,d);str(i:1,s);s: = '-'+s+'-';
  ee: = new(pbutton.init(r,s,cmrele1-1+i,bfnormal));
  insert(ee);r.assign(e,f,g,h);jj: = 1 mod 2;
```

```
  s:='off';
  if jj=1 then
  begin
    s:=+'on ';j:=not(j)
  end;
  rele[i]:=new(preleview,init(r,s,j));insert(rele[i]);
end;
begin
  r.assign(18,4,62,18);Tdialog.init(r,'Relevadores');
  dd:=datorelay;escribe(1,2,6,4,7,2,20,3,1,$fe,dd);
  dd:= dd div 2;escribe(1,5,6,7,7,5,20,6,2,$fd,dd);
  dd:= dd div 2;escribe(1,8,6,10,7,8,20,9,3,$fb,dd);
  dd:= dd div 2;escribe(1,11,6,13,7,11,20,12,4,$f7,dd);
  dd:= dd div 2;escribe(23,2,28,4,29,2,42,3,5,$ef,dd);
  dd:= dd div 2;escribe(23,5,28,7,29,5,42,6,6,$df,dd);
  dd:= dd div 2;escribe(23,8,28,10,29,8,42,9,7,$bf,dd);
  dd:= dd div 2;escribe(23,11,28,13,29,11,42,12,8,$7f,dd);
end;
procedure trelevadores.handleevent(var event:tevent);
procedure rel(i:byte);
procedure releon;
var
  salida:byte;
begin
  with rele[i]^ do
  begin
    setestado('on');drawview;
    setrelenum(not(getrelenum));
    datorelay:=datorelay or getrelenum;
  end;
end;
procedure releoff;
var
  salida:byte;
begin
  with rele[i]^ do
  begin
    setestado('off');drawview;
    setrelenum(not(getrelenum));
    datorelay:=datorelay and getrelenum;
  end;
end;
begin
  if rele[i]^.getestado='off' then releon  else releoff;
end;
begin
  tdialog.handleevent(event);
```

```
  if Event.what = evCommand then
  begin
    CASE event.command of
      cmrele1:rel(1);
      cmrele2:rel(2);
      cmrele3:rel(3);
      cmrele4:rel(4);
      cmrele5:rel(5);
      cmrele6:rel(6);
      cmrele7:rel(7);
      cmrele8:rel(8);
      else
        exit;
    end;
    clearevent(event);
  end;
end;
{**********************************************************************
 Objeto: Área de datos del relevador
**********************************************************************}
constructor treleview.init(bounds:trect;s:string;i:byte);
begin
  Tview.init(bounds);estado:=s;relenum:=i;
end;
procedure treleview.draw;
var
  color:byte;i:integer;b:tdrawbuffer;
begin
  color:=getcolor(1);movechar(B,' ',color,size.x);
  movestr(b,'Relevador-'+estado,color);
  writebuf(0,0,size.x,1,b);
end;
function treleview.getestado:string;
begin
  getestado:=estado;
end;
function treleview.getrelenum:byte;
begin
  getrelenum:=relenum;
end;
procedure treleview.setestado(s:string);
begin
  estado:=s;
end;
procedure treleview.setrelenum(i:byte);
begin
  relenum:=i
```

```
end;
{**********************************************************************
 Objeto: Ventana de los motores
**********************************************************************}
constructor tmotores.init;
var r:trect;dd:integer;pos1,pos2,ret:string;
procedure escribe(a,b,c,d,e,f,g,h,i,j:byte;s1,s2:string);
var
  r:trect;ee:pview;jj:integer;
begin
  r.assign(a,b,c,d);ee:=new(pbutton.init(r,s1,i,bfnormal));
  insert(ee);r.assign(e,f,g,h);motor[j]:=new(pmotorview.init(r,s2));
  insert(motor[j]);
end;
begin
  x2:=xm;y2:=ym;motoractivo:=1;r.assign(18,4,62,21);
  Tdialog.init(r,'Cambio de posición de los motores');
  escribe(25,1,25,2,25,1,42,2,cmcancel,4,'',' Actual
  Siguiente');str(xm:3,pos1);str(x2:3,pos2);
  escribe(1,3,6,5,7,3,42,4,cmmotor1,1,'~1~','Posición del
  MOTOR:'+pos1+'      '+pos2);str(ym:3,pos1);str(y2:3,pos2);
  escribe(1,5,6,7,7,5,42,6,cmmotor2,2,'~2~','Posición del
  MOTOR:'+pos1+'      '+pos2);str(velocidad:3,ret);
  escribe(1,7,6,9,7,7,42,8,cmretardo,3,'~*~','Retardo:'+ret+'[ms]');
  escribe(1,12,6,14,7,12,42,13,cmavanza,4,'~+~','Avanzar un paso');
  escribe(1,10,6,12,7,10,42,11,cmretrocede,4,'~-~','Retroceder un paso');
  escribe(18,14,29,16,31,14,31,15,cmmover,4,'~A~ceptar','');
end;
procedure tmotores.handleevent(var event:tevent);
var
  r:trect;
procedure pasomotor(i:byte);
begin
  motoractivo:=i;
end;
procedure avanzar;
begin
  if motoractivo=1 then if x2<maxmotor1 then inc(x2);
  if motoractivo=2 then if y2<maxmotor2 then inc(y2);
  if motoractivo=3 then if velocidad<1000 then inc(velocidad);
  with motor[motoractivo]^ do
  begin
    setmensaje1(motoractivo);drawview;
  end;
end;
procedure retroceder;
begin
```

```
  if motoractivo=1 then if x2>0 then dec(x2);
  if motoractivo=2 then if y2>0 then dec(y2);
  if motoractivo=3 then if velocidad>0 then dec(velocidad);
  with motor[motoractivo]^ do
  begin
    setmensaje1(motoractivo);drawview;
  end;
end;
procedure mover;
var
  p:pcambwindow;i:byte;
begin
  x1:=xm;y1:=ym;getextent(r);
  r.assign(3,1,41,5);cambio:=new(pcambview,init(r));
  p:=new(pcambwindow,init(cambio,18,7,62,14));
  desktop^.insert((p));
  while (xm<>x2) or (ym<>Y2) do
  begin
    if xm<x2 then incrementax;if ym<Y2 then incrementay;
    if xm>x2 then decrementax;if ym>y2 then decrementay;
    with cambio^ do update;delay(velocidad);
  end;
  for i:=1 to 2 do
  with motor[i]^ do
  begin
    setmensaje1(i);drawview;
  end;
end;
begin
  tdialog.handleevent(event);
  if Event.what=evCommand then
  begin
    CASE event.command of
      cmmotor1:pasomotor(1);
      cmmotor2:pasomotor(2);
      cmretardo:pasomotor(3);
      cmavanza:avanzar;
      cmretrocede:retroceder;
      cmmover:mover;
      else
        exit;
    end;
    clearevent(event);
  end;
end;
{*********************************************************************
 Objeto: Área de datos de los motores
```

```
**********************************************************************)
constructor tmotorview.init(bounds:trect;s:string);
begin
  Tview.init(bounds);mensaje1:=s;
end;
procedure tmotorview.draw;
var
  color:byte;b:tdrawbuffer;
begin
  color:=getcolor(1);movechar(B,' ',color,size.x);
  movestr(b,mensaje1,color);writebuf(0,0,size.x,1,b);
end;
procedure tmotorview.setmensaje1(i:integer);
var
  s1,s2:string;
begin
  if i=1 then
  begin
    str(xm:3,s1);str(x2:3,s2);
  end;
  if i=2 then
  begin
    str(ym:3,s1);str(y2:3,s2);
  end;
  mensaje1:='Posición del MOTOR:'+s1+'    '+s2;
  if i=3 then
  begin
    str(velocidad:3,s1);mensaje1:='Retardo:'+s1+' [ms]'
  end;
end;
{**********************************************************************
 Objeto: Ventana de la superficie a explorar
**********************************************************************}
constructor tsuperfi.init;
var
  r:trect;dd:integer;s1,s2,s3,s4:string;
procedure escribe(a,b,c,d,e,f,g,h,i,j:byte;s1,s2:string);
var
  r:trect;ee:pview;jj:integer;
begin
  r.assign(a,b,c,d);ee:=new(pbutton,init(r,s1,i,bfnormal));
  insert(ee);r.assign(e,f,g,h);super[j]:=new(psuperview,init(r,s2));
  insert(super[j]);
end;
Begin
  r.assign(18,4,62,22);
  Tdialog.init(r,'Superficie de exploración.');
```

```
  superactivo:=1;str(x1:3,s1);str(y1:3,s2);str(x2:3,s3);str(y2:3,s4);
  escribe(1,1,1,1,3,1,42,2,cmclose,6,'','Coordenadas de las esquinas superior e');
  escribe(1,1,1,1,6,2,42,3,cmclose,6,'','inferior del área a explorar:');
  escribe(6,4,11,6,12,4,20,5,cmsuper1,1,'~1~','X1:'+s1);
  escribe(6,6,11,8,12,6,20,7,cmsuper2,2,'~2~','Y1:'+s2);
  escribe(22,4,28,6,29,4,42,5,cmsuper3,3,'~3~','X2:'+s3);
  escribe(22,6,28,8,29,6,42,7,cmsuper4,4,'~4~','Y2:'+s4);
  str(velocidad:3,s1);
  escribe(6,8,11,10,12,8,42,9,cmretardo,5,'~*~','Retardo:'+s1+' [ms]');
  escribe(6,12,11,14,12,12,42,13,cmavanza,6,'~+~','Avanzar un paso');
  escribe(6,10,11,12,12,10,42,11,cmretrocede,6,'~-~','Retroceder un paso');
  escribe(6,14,18,16,28,14,42,15,cmmover,6,'~L~impiar','~F3~ Salir ');
  if banderamodo=0 then s1:='UNA exploración' else s1:='CONTINUA';
  escribe(17,17,17,17,16,16,31,17,cmclose,6,'','Modo:'+s1);
end;
procedure tsuperfi.handleevent(var event:tevent);
procedure pasosuper(i:byte);
begin
  superactivo:=i;
end;
procedure avanzar;
begin
  if superactivo=1 then if x1<maxmotor1 then inc(x1);
  if superactivo=2 then if y1<maxmotor2 then inc(y1);
  if superactivo=3 then if x2<maxmotor1 then inc(x2);
  if superactivo=4 then if y2<maxmotor2 then inc(y2);
  if superactivo=5 then if velocidad<1000 then inc(velocidad);
  with super[superactivo]^ do
  begin
    setmensaje1(superactivo);drawview;
  end;
end;
procedure retroceder;
begin
  if superactivo=1 then if x1>0 then dec(x1);
  if superactivo=2 then if y1>0 then dec(y1);
  if superactivo=3 then if x2>0 then dec(x2);
  if superactivo=4 then if y2>0 then dec(y2);
  if superactivo=5 then if velocidad>0 then dec(velocidad);
  with super[superactivo]^ do
  begin
    setmensaje1(superactivo);drawview;
  end;
end;
procedure limpiar;
var
  i:byte;
```

```
begin
  x1:=0;x2:=0;y1:=0;y2:=0;
  for i:=1 to 4 do
    with super[i]^ do
    begin
      setmensaje1(i);drawview;
    end;
end;
begin
  tdialog.handleevent(event);
  if Event.what=evCommand then
  begin
    CASE event.command of
      cmsuper1:pasosuper(1);
      cmsuper2:pasosuper(2);
      cmsuper3:pasosuper(3);
      cmsuper4:pasosuper(4);
      cmmover:limpiar;
      cmretardo:pasosuper(5);
      cmavanza:avanzar;
      cmretrocede:retroceder;
      else
        exit;
    end;
    clearevent(event);
  end;
end;
{*********************************************************************
 Objeto: Área de datos de la superficie a explorar
*********************************************************************}
constructor tsuperview.init(bounds:trect;s:string);
Begin
  Tview.init(bounds);mensaje1:=s;
end;
procedure tsuperview.draw;
var
  color:byte;b:tdrawbuffer;
begin
  color:=getcolor(1);movechar(B,' ',color,size.x);
  movestr(b,mensaje1,color);writebuf(0,0,size.x,1,b);
end;

procedure tsuperview.setmensaje1(i:integer);
var
  s1:string;
begin
  if i=1 then
```

```
  begin
    str(x1:3,s1);s1:='X1:'+s1;
  end;
  if i=2 then
  begin
    str(y1:3,s1);s1:='Y1:'+S1;
  end;
  if i=3 then
  begin
    str(x2:3,s1);s1:='X2:'+s1;
  end;
  if i=4 then
  begin
    str(y2:3,s1);s1:='Y2:'+S1;
  end;
  mensaje1:=s1;
  if i=5 then
  begin
    str(velocidad:3,s1);mensaje1:='Retardo:'+s1+' [ms]'
  end;
end;
{*****************************************************************
 Objeto: Sistema principal
*****************************************************************}
constructor tsistema.init;
var
  R:TRect;event:tevent;
begin
  tapplication.init;registerobjects;
  registerviews;registerdialogs;
  registerapp;registeradc;
  getextent(R);r.a.x:=r.b.x-9;r.b.y:=r.a.y+1;
  clock:=new(PClockview,init(R));insert(CLOCK);
  datosinic;BGIPath := NewStr(FExpand(PathToDrivers));
  AppDriver := Detect;AppMode := 0;
  if not GraphAppInit(AppDriver, AppMode, BGIPath, True) then MessageBox('Cannot load graphics driver.',
    nil, mfError or mfOkButton);
  event.what:=evcommand;event.command:=cminfo;
  putevent(event);
end;
procedure tsistema.informacion;
var
  R:trect;D:pdialog;C:word;
begin
  R.assign(20,7,61,15);D:=new(PDialog, init(R,'Información'));
  R.assign(14,2,31,3);D^.insert(New(pstatictext, init(R,'I. I. M. A. S.')));
  R.assign(2,3,39,4);D^.insert(New(pstatictext, init(R,'Controlador de Procesos
```

```
  Industriales.')));R.assign(18,5,24,7);
  D^.insert(new(Pbutton, init(R, '~O~k',cmcancel,bfnormal)));
  C:= Desktop^.execview(D);
end;
procedure tsistema.idle;
begin
  Tapplication.idle;clock^.update;t^.update;opto^.update;controlview^.update;
end;
procedure Tsistema.Handleevent(var Event:TEvent);
var
  P:tview;r:Trect;
procedure adcopen;
var
  contador,x,y:integer;
  x1,x2,y1,y2:byte;
  p:padcwindow;
begin
  contador:=0;x1:=0+10;y1:=0+3;x2:=50+10;y2:=14;
  getextent(r);r.assign(5,2,48,9);
  t:=new(padcview,init(r));p:=new(padcwindow,init(t,x1,y1,x2,y2));
  desktop^.insert(validview(p));
end;
procedure rele;
var
  a:prelevadores;c:word;
begin
  a:=new(prelevadores,init);desktop^.insert(validview(a));
end;
procedure pasos;
var
  a:pmotores;c:word;
begin
  a:=new(pmotores,init);desktop^.insert(validview(a));
end;
procedure setsup;
var
  a:psuperfi;c:word;
begin
  a:=new(psuperfi,init);desktop^.insert(validview(a));
end;
procedure dac;
var
  a:pdacwindow;c:word;
begin
  a:=new(pdacwindow,init);desktop^.insert(validview(a));
end;
procedure opt;
```

```
var
  p:poptowindow;
begin
  getextent(r);r.assign(3,2,41,5);
  opto:=new(poptoview,init(r));p:=new(poptowindow,init(opto,18,7,62,14));
  desktop^.insert(validview(p));
end;
procedure grafic;
begin
  if not GraphicsStart then
    MessageBox(GraphErrorMsg(GraphResult) + '.',
    nil, mfError or mfOkButton)
  else
  begin
    GRAFICS;
    if banderagrafica=0 then
    begin
      freemem(p1,dimension);freemem(p2,dimension);
    end;
    GraphicsStop;
  end;
end;
procedure activacontrol;
var
  r:trect;proc:pcontrol;
begin
  getextent(r);r.assign(1,1,77,22);
  controlview:=new(pcontrolview,init(r));proc:=new(pcontrol,init(controlview));
  desktop^.insert(validview(proc));
end;
Procedure exploracion;
procedure digitaliza;
var
  transferencia:word;
begin
  transferencia:=datoadcw[canal];writeln(datosbinarios,transferencia);
  writeln(datoscolor,(transferencia div 256))
end;
procedure unaexpl;
var
  p:pcambwindow;xp,yp:integer;cambio:Pcambview;
begin
  rewrite(datosbinarios);rewrite(datoscolor);
  banderaarchivo:=1;
  if banderamodo=0 then
  begin
    getextent(r);r.assign(3,1,41,5);
```

```
    cambio:=new(pcambview,init(r));p:=new(pcambwindow,init(cambio,18,7,62,14));
    desktop^.insert(validview(p));
  end;
  xi:=x1;yi:=y1;xf:=x2;yf:=y2;
  if x1>x2 then
  begin
    xi:=x2;xf:=x1;
  end;
  if y1>y2 then
  begin
    yi:=y2;yf:=y1;
  end;
  if banderaposicion=0 then resetmotor;
  while (xm<>xi) or (ym<>yi) do
  begin
    if xm<xi then incrementax;if ym<Yi then incrementay;
    if xm>xi then decrementax;if ym>yi then decrementay;
    if banderamodo=0 then with cambio^ do update;
    delay(velocidad);
  end;
  for yp:=yi to yf do
  begin
    if (yp<>yi) then incrementay;
    for xp:=xi to xf do
    begin
      if (((yi-yp) mod 2) = 0) and (xp<>xi) then incrementax;
      if (((yi-yp) mod 2) <> 0) and (xp<>xi) then decrementax;
      if banderamodo=0 then cambio^.update;
      canal:=00;generapulsocontrol;escrituraadc;delay(velocidad);
      lecturaadc;digitaliza;
    end;
  end;
end;
procedure expcont;
begin
  if not GraphicsStart then
    MessageBox(GraphErrorMsg(GraphResult) + '.',nil, mfError or mfOkButton)
  else
  begin
    Banderamodo:=1;unaexpl;banderagrafica:=2;GRAFICS;banderagrafica:=3;
    repeat
      unaexpl;grafics;
    until keypressed;
  end;
  GraphicsStop;
end;
begin
```

```
  if banderamodo=0 then unaexpl;if banderamodo=1 then expcont;
end;
procedure grafic1;
begin
  banderagrafica:=0;grafic
end;
procedure grafic2;
begin
  banderagrafica:=1;grafic
end;
procedure modo1;
begin
  banderamodo:=0;
end;
procedure modoC;
begin
  banderamodo:=1;
end;
begin
  Tapplication.Handleevent(Event);
  if Event.what=evCommand then
  begin
    CASE event.command of
      cmnone:newdialog;
      cmadcopen:adcopen;
      cminfo:informacion;
      cmrele:rele;
      cmopto:opt;
      cmgrafica1:grafic1;
      cmgrafica2:grafic2;
      cmdac:dac;
      cmcontrol:activacontrol;
      cmresmot:resetmotor;
      cmpasos:pasos;
      cmsetsup:setsup;
      cmexploracion:exploracion;
      cmmodo1:modo1;
      cmmodoc:modoC;
      else
        exit;
    end;
    ClearEvent(Event)
  end;
end;
{Tsistema}
procedure tsistema.Initmenuhar;
var
```

```
  R:Trect;
begin
  GetExtent(R);R.B.Y:=R.A.Y+1;
  Menubar:=New(Pmenubar,Init(R,Newmenu(
    Newsubmenu('~-~-~',hcnocontext,newmenu(
      newitem('~I~nformacion','',kbnokey,cminfo,hcnocontext,
      nil)),
    Newsubmenu('~S~istema',hcnocontext,NewMenu(
      Newitem('~E~stado
        genelar','F1',kbF1,cmcontrol,hcnocontext,
      Newitem('~C~onfiguración','F2',kbF2,cmnone,hcnocontext,
      Newline(
      Newitem('S~a~lir','Alt-x',kbaltx,cmquit,hcnocontext,
      nil))))),
    Newsubmenu('~P~rueba',hcnocontext,newmenu(
      Newitem('~A~DC','',kbnokey,cmadcopen,hcnocontext,
      Newitem('~D~AC','',kbnokey,cmdac,hcnocontext,
      Newitem('~O~pto','',kbnokey,cmopto,hcnocontext,
      Newitem('~R~elay','',kbnokey,cmrele,hcnocontext,
      newitem('~G~ráfica','',kbnokey,cmgrafica1,hcnocontext,
      Nil))))),
    Newsubmenu('~M~otores de pasos',hcnocontext,newmenu(
      Newitem('~I~nicializa
        motor','',kbnokey,cmresmot,hcnocontext,
      Newitem('~C~ambiar
        posición','',kbnokey,cmpasos,hcnocontext,
      Nil))),
    Newsubmenu('~A~plicación',hcnocontext,newmenu(
      Newsubmenu('~M~odo de operación',hcnocontext,newmenu(
        Newitem('~C~ontinuo','',kbnokey,cmmodoC,hcnocontext,
        Newitem('~U~na sola
          exploración','',kbnokey,cmmodo1,hcnocontext,
        nil))),
      Newitem('~S~uperficie de
        exploración','',kbnokey,cmsetsup,hcnocontext,
      Newitem('~E~xploración','',kbnokey,cmexploracion,
        hcnocontext,
      Newitem('~G~raficación de
        resultados','',kbnokey,cmgrafica2,hcnocontext,
      Nil))))),
    Newsubmenu('~V~entana',hcnocontext,newmenu(
      Newitem('~C~errar','F3',kbF3,cmclose,hcnocontext,
      Newitem('~S~iguiente','F6',kbF6,cmnext,hcnocontext,
      Newitem('~A~nterior','F7',kbF7,cmprev,hcnocontext,
      Nil))))),
    Nil))))))
  )));
```

```
end;
procedure Tsistema.Initstatusline;
var
  R:Trect;
begin
  Getextent(R);
  R.A.Y:=R.B.Y-1;
  Statusline:=New(Pstatusline,Init(R,
    Newstatusdef(0,$ffff,
      Newstatuskey('~Alt-x~ Salir',KbAltx,cmquit,
      Newstatuskey('~F1~ Estado del sistema',kbF1,cmcontrol,
      Newstatuskey('~F2~ Configuración',kbF2,cmnone,
      Newstatuskey('~F3~ Cerrar',kbF3,cmclose,
      Newstatuskey('~F10~ Menu',KbF10,cmMenu,
      nil)))),
    nil)
  ));
end;
procedure Tsistema.newdialog;
var
  R:trect;D:pdialog;C:word;s:string;
begin
  R.assign(1,2,38,14);D:=new(PDialog, init(R,'Configuración'));
  R.assign(4,2,36,3);D^.insert(New(pstatictext, init(R,'Mapa de puertos.')));
  R.assign(4,3,36,4);str(portadc:4,s);D^.insert(New(pstatictext, init(R,'  Convertidor A/D:'+s)));
  R.assign(4,4,36,5);str(portdac:4,s);D^.insert(New(pstatictext, init(R,'  Convertidor D/A'+s)));
  R.assign(4,5,36,6);str(portrelay:4,s);D^.insert(New(pstatictext, init(R,'  Salidas
  digitales:'+s)));R.assign(4,6,36,7);str(portoptos:4,s);
  D^.insert(New(pstatictext, init(R,'  Estradas Digitales:'+s)));
  R.assign(4,7,36,8);str(portcontrol:4,s);
  D^.insert(New(pstatictext, init(R,'  Control:'+s)));C:= Desktop^.execview(D);
end;
{********Termina bloque de declaración de objetos**************}
{******************Inicia programa principal***************}
var
  sistema:tsistema;
begin
  datosinic;
  port[portcontrol]:=(freemuestreo and $03)*2*2 or (canal and $03);
  instala;sistema.init;sistema.run;sistema.done;reinstala;
  if banderaarchivo=1 then
  begin
    close(datosbinarios);close(datoscolor);
  end;
end.
{**********************Termina programa principal***************}
```

Declaración de unidades.

```
{************************************************************          Unidad datos
                 *
*          Control de procesos industriales        *
*                   I. I. M. A. S.            *
*                   1992-1993                 *
************************************************************
{ En esta rutina se inicializan todas las variables empleadas por el sistema}
************************************************************
Unit datos;
interface
  uses graph;
type
  tipo=array[1..4] of pointtype;
var
  portadc, portdac, portrelay, portoptos, portcontrol:word;
  datodac:word; dacLSB,dacMSB:byte; datoadc:array[0..7] of byte;
  datoadcw:array[0..4] of word; bufferadc:array[0..1000] of word;
  datooptos, datorelay, contadoradc:word; resolucion:byte;
  cuatro:tipo; incremento, decremento, altura, banderamodo, banderagrafica, banderagrafica1,
  banderagrafica2, banderaaplicacion:integer; factorY:real; canalvar, canal:byte;
  conEscX, conEscY:integer; canalactivo,EscY:array[1..5] of string; escX:array[1..15] of string;
  velocidad, frecmuestreo, maxmotor1, maxmotor2, control:integer; direccion1,direccion2:char;
  paso1,paso2:byte; PathToDrivers:String; xi,yi,xf,yf:integer; xm,ym,x1,x2,y1,y2:integer; controlmotorx,
  controlmotory,tamx,tamy,sentidox,sentidoy:byte;datosbinarios,datoscolor:text;
  banderaarchivo:byte;banderaposicion:byte;
const
  cmnone=100; cmadcopen=102; cmretardo=103; cminfo=104; cmadc=105; cmrele=101; cmrele1=106;
  cmrele2=107; cmrele3=108; cmrele4=109; cmrele5=110; cmrele6=111; cmrele7=112; cmrele8=113;
  cmopto=114; cmfuncion1=116; cmfuncion2=117; cmfuncion3=118; cmfuncion4=119; cmsuper1=120;
  cmsuper2=121;cmsuper3=122; cmsuper4=123; cmmodoc=124; cmmodo1=125; cmgrafica1=126;
     cmgrafica2=127; cmdac=128; cmcontrol=129; cmresmot=130; cmpasos=131; cmmotor1=132;
     cmmotor2=133; cmavanza=134; cmretrocede=135; cmmover=136; cmexploracion=139; cmsetsup=140;
  cmarea=138; cmgrafica=115; cmmodo=137; Procedure datosinic;Procedure datosgrafinic;
implementation
procedure datosgrafinic;
begin
  randomize;contadoradc:=0;cuatro[1].x:=40;cuatro[1].y:=40;cuatro[2].x:=40+200+3;
  cuatro[2].y:=40;cuatro[4].x:=40;cuatro[4].y:=40+200+3;cuatro[3].x:=40+200+3;
  cuatro[3].y:=40+200+3;incremento:=1;decremento:=1;altura:=199;factorY:=1;
  banderagrafica1:=0;banderagrafica2:=0;canal:=0;canalvar:=1;conEscY:=1;conEscX:=10;
  EscX[1]:='2.0';EscX[2]:='2.2';EscX[3]:='2.5';EscX[4]:='2.8';EscX[5]:='3.3';EscX[6]:='4.0';
  EscX[7]:='5.0';EscX[8]:='6.6';EscX[9]:='10.0';EscX[10]:='20.0';EscX[11]:='40.0';EscX[12]:='60.0';
  EscX[13]:='80.0';EscX[14]:='100';Escy[1]:='1.00';Escy[2]:='0.50';Escy[3]:='0.33';
  Escy[4]:='0.25';Escy[5]:='0.20';canalactivo[1]:='0';canalactivo[2]:='1';canalactivo[3]:='2';
```

```
  canalactivo[4]:='3';canalactivo[5]:='4';
end;
procedure datosinic;
begin
  datosgrafinic;
  PathToDrivers:='a:';  { Default location of *.BGI files }
  portadc:=$300;portdac:=$30c;portrelay:=$308;portoptos:=$309;portcontrol:=$30b;
  datorelay:=$ff;resolucion:=12;frecmuestreo:=1;control:=frecmuestreo*2*2+(canal and $03);
  velocidad:=50;maxmotor1:=2000;maxmotor2:=2000;direccion1:='i';direccion2:='i';
  paso2:=0;paso1:=0;port[portrelay]:=datorelay;port[portadc]:=00;x1:=0;xm:=0;
  x2:=0;ym:=0;y1:=0;y2:=0;banderamodo:=0;banderagrafica:=0;assign(datosbinarios,'c:\binario.cpi');
  assign(datoscolor,'c:\colores.cpi');banderaarchivo:=0;
  banderaaplicacion:=00;banderaposicion:=0;
end;
end.
{*****************************************************************                    Unidad Interrup
                    *
*        Rutinas para el soporte de interrupciones          *
*****************************************************************}
unit intp;
interface
uses datos;
procedure interrupcion;
procedure instala;
procedure  reinstala;
procedure lecturaadc;
procedure lecturaoptoacopladores;
procedure escriturarelevadores;
procedure escrituraadc;
procedure escrituradac;
implementation
var
  oldmask,segmento,offset:integer;
const
  stat8259=$20;mask8259=$21;eoirq3=$63;enable=$f7;
  vector=$002c;
procedure lecturaadc;
begin
  datoadc[canal*2+0]:=port[portadc];datoadc[canal*2+1]:=port[portadc+1];
  datoadcw[canal]:=datoadc[canal*2+0]*16+(datoadc[canal*2+1] div     16);
end;
procedure lecturaoptoacopladores;
begin
  datooptos:=port[portoptos];
end;
procedure escriturarelevadores;
begin
```

```
  port[portrelay]:=datorelay;
end;
procedure escrituraadc;
begin
  control:=frecmuestreo*2^2+canal;port[portcontrol]:=control;
  port[portadc]:=00;
end;
procedure escrituradac;
begin
  port[portdac]:=dacLSB;port[portdac+1]:=dacMSB;
end;
procedure interrupcion;
procedure proceso;
begin
end;
procedure bits8;
begin
end;
procedure bits12;
begin
  if banderaaplicacion=0 then
  begin
    lecturaadc;lecturaoptoacopladores;
    escrituradac;escriturarelevadores;
    if (contadoradc<1001) and ((canalvar-1)=canal) then
    begin
      bufferadc[contadoradc]:=datoadcw[canal];
      contadoradc:=contadoradc+1;
    end;
    canal:=(canal + 1) mod 4;
  end;
end;
begin
  inline($50/$53/$51/$52/$57/$56/$06/$1e/
       $b8/$00/$00/
       $50/$1f/$fb);
  case resolucion of
    8:bits8;
    else
      bits12
  end;
  if banderaaplicacion=0 then escrituraadc;
  port[stat8259]:= eoirq3;
  inline($1f/$07/$5e/$5f/$5a/$59/$5b/$58/$cf);
end;
procedure instala;
begin
```

```
  inline($fa);
  offset:=memw[$0000:vector];segmento:=memw[$0000:vector+2];
  memw[$0000:vector]:=ofs(interrupcion)+$a;memw[$0000:vector+2]:=cseg;
  memw[cseg:ofs(interrupcion)+$13]:=dseg;oldmask:=port[mask8259];
  port[mask8259]:= port[mask8259] and enable;inline($fb);
end;
procedure reinstala;
begin
  inline($fa);
  memw[$0000:vector]:=offset;memw[$000:vector+2]:=segmento;
  port[mask8259]:=oldmask;
  inline($fb);
end;
end.
{*****************************************************************
 *                    tesis                                      *
 *                    convertidor                                *
 *                    1992-1993                                  *
*****************************************************************
```

Rutina para implementar la estructura utilizada para desplegar el estado de las variables conectadas al convertidor analógico/digital.

Unidades utilizadas:
-Drivers
-Objects
-App
-Views
-Dialogs
-Interrupt

```
 *****************************************************************}

unit converci;
{$f+,o+,x+,s-,d-}
interface
  uses drivers,objects, app, views,dialogs,intp,datos;
var
  dato,selec:byte;
type
padcview = ^tadcview;
tadcview = object(tview)
  col:word;
  salida:byte;
  constructor init(bounds:trect);
  procedure draw;virtual;
  procedure update;
end;
padcwindow = ^tadcwindow;
```

```
tadcwindow = object (twindow)
  padc:padcview;
  constructor init(pad:padcview;x1,y1,x2,y2:byte);
end;
const
Radcview : tstreamrec = (
  objtype: 10020;
  vmtlink: ofs(typeof(tadcview)^));
Radcwindow : tstreamrec =(
  objtype : 10021;
  vmtlink: ofs(typeof(tadcwindow)^));
procedure registeradc;
implementation
constructor tadcwindow.init(pad:padcview;x1,y1,x2,y2:byte);
var
  R:trect;
begin
  R.assign(X1,y1,x2,y2);twindow.init(R, 'ADC0809', 0 );
  growmode :=0;palette :=wpbluewindow;padc:=pad;
  insert(padc);
end;
constructor tadcview.init(bounds:trect);
var
  H:word;
begin
  Tview.init(bounds);options:=optionsor ofselectable;
  eventmask:= eventmask or evmouseauto;
  col:=6;drawview;
end;
procedure Tadcview.draw;
const
  width=47;
var
  i:integer;s,contador:string;b:array[1..width] of word;
  color, boldcolor, specialcolor:byte;
begin
  color:=getcolor(col);boldcolor:=getcolor(7);
  for i:=0 to 3 do
  begin
    str(datoadcw[i]:5,s);str(i:2,contador);
    movechar(b,' ',color,width);
    movestr(b,'Canal '+contador+':'+s+' Sensor de temp.',color);
    writeline(0,i*2+0,width,1,b);movechar(b,' ',color,width);
    writeline(0,i*2+1,width,1,b);
  end;
end;
procedure tadcview.update;
```

```
begin
  drawview
end;
procedure registeradc;
begin
  registertype(radcview);registertype(radcwindow);
end;
end.
{*****************************************************************  Unidad Aplicaci
                  *
*                 Control de procesos industriales          *
*                        I. I. M. A. S.                     *
*                          1992-1993                        *
*****************************************************************
{ En esta rutina se implementan las rutinas necesarias para el desarrollo de la aplicación descrita en el capítulo 5.}
*****************************************************************
Unit aplicaci;
interface
  uses crt,datos,intp;
  procedure resetmotor;
  procedure pasomotorx;
  procedure pasomotory;
  procedure incrementax;
  procedure decrementax;
  procedure incrementay;
  procedure decrementay;
  procedure generapulsocontrol;
implementation
procedure pasomotorx;
begin
  controlmotorx:=tamx+sentidox+$00;datorelay:=controlmotorx + controlmotory;
  escriturarelevadores;controlmotorx:=tamx+sentidox+$04;
  datorelay:=controlmotorx + controlmotory;escriturarelevadores;
  controlmotorx:=tamx+sentidox+$00;datorelay:=controlmotorx + controlmotory;
  escriturarelevadores;
end;
procedure pasomotory;
begin
  controlmotory:=tamy+sentidoy+$00;
  datorelay:=controlmotorx + controlmotory;escriturarelevadores;
  controlmotory:=tamy+sentidoy+$20;datorelay:=controlmotorx + controlmotory;
  escriturarelevadores;controlmotory:=tamy+sentidoy+$00;
  datorelay:=controlmotorx + controlmotory;escriturarelevadores;
end;
procedure incrementax;
begin
  inc(xm);sentidox:=0;tamx:=0;pasomotorx;
```

```
end;
procedure decrementax;
begin
  dec(xm);sentidox:=2;tamx:=0;pasomotorx;
end;
procedure incrementay;
begin
  inc(ym);sentidoy:=0;tamy:=0;pasomotory;
end;
procedure decrementay;
begin
  dec(ym);sentidoy:=$10;tamy:=0;pasomotory;
end;
procedure generapulsocontrol;
begin
  datorelay:=controlmotorx + controlmotory + $40;
  escriturarelevadores;datorelay:=controlmotorx + controlmotory + $00;
  escriturarelevadores;
end;
procedure resetmotor;
var
  d:byte;
begin
  banderaaplicacion:=$ff;d:=port[portoptos];
  while ((d and $01)=0) or ((d and $02)=0)  do
  begin
    if ((d and $01)=0) then decrementax;
    if ((d and $02)=0) then decrementay;
    delay(velocidad);d:=port[portoptos];
  end;
  xm:=0;ym:=0;banderaposicion:=1;
end;
end.
{****************************************************************                Unidad Gráfica
                    *
*                   Control de procesos industriales          *
*                          I. I. M. A. S.              *
*                           1992-1993                  *
****************************************************************
{ En esta rutina se implementan las rutinas necesarias para desplegar resultados en modo gráfico (rutina del
osciloscopio y rutina para mostrar resultados de la superficie explorada en la aplicación implementada}
****************************************************************
Unit grafica;
interface
  uses crt, graph, drivers,app,datos,intp;
var
  evento:tevent;
```

```
  s1,s2,s3:string;
  Gd, Gm,i,j : Integer;
  Color,x,y,dimension,dalow: Word;
  Palette : PaletteType;
  p1,p2:pointer;
  opcion:string;
  incx,incz,incp:integer;
  incy:real;
  cuadro:tipo;
  procedure grafics;
implementation
procedure grafics;
procedure parametros;
begin
  incx:=incremento;incp:=decremento;incy:=factory;
  incz:=altura;conescx:=10;conescy:=1;
end;
procedure cuadrado(x1,y1,x2,y2,fs,cf,cc:integer);
var
  cuadro:tipo;
begin
  setcolor(cc);setfillstyle(fs,cf);
  cuadro[1].x:=x1;cuadro[1].y:=y1;cuadro[2].x:=x2;cuadro[2].y:=y1;
  cuadro[3].x:=x2;cuadro[3].y:=y2;cuadro[4].x:=x1;cuadro[4].y:=y2;
  fillpoly(4,cuadro);
end;
procedure
  texto(xt,yt,x1,y1,x2,y2,fs,cf,cc,ct:integer;letra:string);
begin
  cuadrado(x1,y1,x2,y2,fs,cf,cc);setcolor(ct);
  outtextXY(xt,yt,letra);
end;
procedure flechaup(x1,y1:integer);
var
  x,y:integer;
begin
  for y:=0 to 1 do for x:=0 to 10 do putpixel(x1+y,y1+x,0);
  for x:=1 to 4 do
  begin
    putpixel(x1-1,y1+x,0);putpixel(x1+2,y1+x,0);
  end;
  for x:=3 to 4 do
  begin
    putpixel(x1-2,y1+x,0);putpixel(x1+3,y1+x,0);
  end;
end;
procedure flechadn(x1,y1:integer);
```

```
var
  x,y:integer;
begin
  for y:=0 to 1 do for x:=-10 to 0 do putpixel(x1+y,y1+x,0);
  for x:=-4 to -1 do
  begin
    putpixel(x1-1,y1+x,0);putpixel(x1+2,y1+x,0);
  end;
  for x:=-4 to -3 do
  begin
    putpixel(x1-2,y1+x,0);putpixel(x1+3,y1+x,0);
  end;
end;
procedure flechaizq(x1,y1:integer);
var
  x,y:integer;
begin
  for y:=0 to 1 do for x:=0 to 10 do putpixel(x1+x,y1+y,0);
  for x:=1 to 4 do
  begin
    putpixel(x1+x,y1-1,0);putpixel(x1+x,y1+2,0);
  end;
  for x:=3 to 4 do
  begin
    putpixel(x1+x,y1-2,0);putpixel(x1+x,y1+3,0);
  end;
end;
procedure flechader(x1,y1:integer);
var
  x,y:integer;
begin
  for y:=0 to 1 do
  for x:=-10 to 0 do putpixel(x1+x,y1+y,0);
  for x:=-4 to -1 do
  begin
    putpixel(x1+x,y1-1,0);putpixel(x1+x,y1+2,0);
  end;
  for x:=-4 to -3 do
  begin
    putpixel(x1+x,y1-2,0);putpixel(x1+x,y1+3,0);
  end;
end;
procedure tablero;
begin
  setcolor(0);outtextxy(40,260,'C O N T R O L E S :');
  setlinestyle(0,0,1);texto(42,280,40,275,40+35,275+35,1,15,8,0,'8');
  flechaup(55,295);outtextXY(80,305,'Escala');outtextxy(80,315,'Vertical.');
```

```
  texto(42,320,40,315,40+35,315+35,1,15,8,0,'2');flechadn(55,345);
  texto(42,360,40,355,40+35,355+35,1,15,8,0,'4');flechaizq(50,385);
  outtextxy(120,360,'Escala');outtextxy(120,370,'Horizontal.');
  texto(82,360,80,355,80+35,355+35,1,15,8,0,'6');flechader(100,385);
  texto(222,280,220,275,220+35,275+35,1,15,8,0,'9');outtextXY(222,300,'PgUp');
  outtextxy(260,305,'Posición');outtextxy(260,315,'Vertical.');
  texto(222,320,220,315,220+35,315+35,1,15,8,0,'3');outtextXY(222,340,'PgDn');
  texto(422,394,420,390,420+35,390+35,1,8,8,0,'0');outtextXY(422,414,'Ins');
  outtextXY(460,400,'Reanudar');outtextXY(460,410,'Muestreo.');
  texto(422,280,420,275,420+35,275+35,1,15,8,0,'.');outtextXY(422,302,'Del');
  outtextXY(460,290,'Suspender Muestreo.');texto(422,345,420,315,420+35,315+70,1,8,8,0,'Intr');
  outtextXY(460,340,'Congelar Imagen.');outtextXY(460,350,'(Sin muestreo)');
  texto(422,455,420,430,420+35,430+35,1,8,8,0,'Esc');outtextXY(460,440,'CondicionesIniciales.');
  texto(42,455,40,440,40+50,440+35,1,8,8,0,'Alt');texto(119,444,115,440,115+35,440+35,1,15,8,0,'X');
  outtextXY(100,455,'+');outtextXY(42,425,'Salir.');texto(235,372,220,355,220+35,355+35,1,8,8,0,'-');
  outtextxy(260,385,'Selección');outtextxy(260,395,'de Canal.');
  texto(235,412,220,395,220+35,395+70,1,8,8,0,'+');
end;
procedure escalas;
begin
  putimage(410,60,p2^,normalput);outtextxY(410,60,'      ');
  outtextxY(410,60,escy[conescy]);outtextxy(410,100,'      ');
  outtextxy(410,100,escX[conescx]);outtextxy(410,130,'      ');
  outtextxy(410,130,canalactivo[canalvar]);
end;
procedure insgraficososciloscopio;
begin
  datosgrafinic;parametros;setpalette(0,0);
  cuadrado(0,0,getmaxX,getmaxY,1,7,1);
  setcolor(4);setfillstyle(1,1);fillpoly(4,cuatro);
  setlinestyle(1,0,0);
  for i:=1 to 9 do line(42,42+20*i,241,42+20*i);
  for i:=1 to 9 do line(42+20*i,42,42+20*i,241);
  dimension:=imagesize(42,42,242,242);getmem(p1,dimension);
  getimage(42,42,242,242,p1^);dimension:=imagesize(445,42,480,120);
  getmem(p2,dimension);getimage(445,42,480,120,p2^);
  setcolor(0);randomize;tablero;setviewport(42,42,639,242,clipon);
  outtextxy(300,50,'Escala');outtextxy(300,60,'Vertical:');outtextxy(300,70,'(V/div)');
  outtextxy(300,90,'Escala');outtextxy(300,100,'Horizontal:');outtextxy(300,110,'(ms/div)');
  outtextxy(300,130,'Canal:');escalas;
end;
procedure insgraficosexploracion;
var
  sx,sy:string;con1,con2,colorpixel:integer;
begin
  datosgrafinic;setpalette(0,0);
  cuadrado(0,0,getmaxX,getmaxY,1,0,0);setcolor(15);
```

```
  setlinestyle(1,0,0);outtextxy(32,440,'Graficación de resultados.' );
  outtextxy(32,450,'Area de exploración:'  );str(xi:3,sx);str(yi:3,sy);
  outtextxy(32,460,'Esquina inicial: ('+sx+','+sy+')');str(xf:3,sx);str(yf:3,sy);
  outtextxy(32,470,'Esquina final:   ('+sx+','+sy+')');outtextxy(320,450,'Modo:');
  if banderamodo=0 then sx:='Una sola exploración.'else sx:='CONTINUO';
  outtextxy(320,460,sx);outtextXY(320,470,'Presione cualquier tecla para salir.');
  for colorpixel:=0 to 15 do for con1:=1 to 25 do
  for con2:=1 to 3 do putpixel(230+con1+colorpixel*25,440+con2,colorpixel);
  for con1:=0 to 15 do
  begin
    str(con1:2,sx);outtextxy(239+con1*25,440,sx);
  end;
  for con1:=1 to 30 do
  begin
    str(((con1*2*10) mod 100):2,sx);outtextxy(30+con1*20,0,sx);
  end;
  for con1:=1 to 21 do
  begin
    str((con1*2*10):3,sx); outtextxy(0,10+con1*20,sx);
  end; outtextxy(0,10,'  0');
end;
procedure explorar;
var
  xp,yp,ci,cf,ri,rf,x,y:integer;
  color,con1,con2,colorpixel:integer;
begin
  setviewport(0,0,639,440,clipon);
  x:=30;y:=10; color:=0; colorpixel:=0; ri:=yi;rf:=yf;ci:=yi;cf:=xf;
  for yp:=ri to rf do
  begin
    for xp:=ci to cf do
    begin
      readln(datoscolor,color); putpixel(x,y,color);
      if ((yp-ri) mod 2) = 0 then inc(x) else dec(x);
    end; inc(y);
    for con1:=4 to 34 do
      for con2:=1 to  5 do putpixel(con1,con2,(colorpixel mod 16)); inc(colorpixel);
    if ((yp-ri) mod 2) = 0 then dec(x) else inc(x);
end; end;
Procedure Actualiza;
procedure incrementaP;
begin
  incp:=incp-1; conescx:=conescx-1; escalas;
end;
procedure decrementaX;
begin
  incx:=incx-1; conescx:=conescX+1; escalas;
```

```
end;
begin
   getkeyevent(evento);
   if evento.keycode=kbaltx then opcion:='para';
   if (banderaGrafica2=0)  or (banderaGrafica1=0) then
   begin
     if (evento.keycode=kbRight) then
       if (incp>1) then incrementaP
         else
           if (incx<10)  then
           begin
             incx:=incx+1; conescx:=conescx-1; escalas;
           end;
     if (evento.keycode=kbLeft) then
       if (incx>1) then decrementaX
         else
           if (incp<5) then
           begin
             incp:=incp+1; conescX:=conescx+1; escalas;
           end;
     if (evento.keycode=kbup) and (incy<5) then
     begin
       incy:=incy+1; conescY:=conescY+1; escalas;
     end;
     if (evento.keycode=kbdown) and (incy>1) then
     begin
       incy:=incy-1; conescY:=conescY-1; escalas;
     end;
     if (evento.keycode=kbpgup) and (inca>0) then inca:=inca-15;
     if (evento.keycode=kbpgdn) then inca:=inca+15;
     if (evento.keycode=kbhome) and (inca>0) then inca:=inca-1;
     if (evento.keycode=kbend) then inca:=inca+1;
     if (evento.keycode=kbesc) then
     begin
       parametros; escalas;
     end;
     if (evento.keycode=kbgrayminus) and (canalvar>1) then
     begin
       canalvar:=canalvar-1; escalas;
     end;
     if (evento.keycode=kbgrayplus) and (canalvar<4) then
     begin
       canalvar:=canalvar+1; escalas;
   end;end;
   if (evento.keycode=kbDel) then banderaGrafica1:=1;
   if (evento.Keycode=kbIns) then banderaGrafica1:=0;
   if (evento.keycode=kbEnter) then banderaGrafica2:=not(banderaGrafica2);
```

```
end;
procedure osciloscopio;
begin
  opcion:='sigue';
  repeat
    if (banderaGrafica1=0) or (banderaGrafica2=0) then
    begin
      putimage(0,0,p1^,andput); if banderaGrafica1=0 then contadorade:=0; j:=0;
      repeat
        i:=j;
        if (i*incx+1)<199 then
        putpixel(i*incx+1,round(inca-(bufferade[j*incp])*incY*(200/4095)),15);j:=(j+1);actualiza;
      until ((i+1)*incx)>199;
    end;
    if not ((banderaGrafica1=0) or (banderaGrafica2=0)) then actualiza;
  until opcion<>'sigue';
end;
Begin
  if banderagrafica=0 then
  begin
    insgraficososciloscopio; osciloscopio;
  end;
  if (banderagrafica=1) or (banderagrafica=2) then
  begin
    reset(datoscolor); insgraficosexploracion;
    explorar; if banderagrafica=1 then opcion:=readkey;
  end;
  if banderagrafica=3 then
    begin
      reset(datoscolor); explorar;
end;
end;
end.
```

APÉNDICE B.

INTERRUPCIONES DE LA IBM PC.

Tipo de Interrupción	Tipo de Interrupción (Hex).	Nombre	
0	0	división por cero	Dirección 00000H
1	1	Un sólo paso	
2	2	NMI	
3	3	*breakpoint*	
4	4	sobreflujo	
5	5	Impresión de pantalla	
6	6	no usado	
7	7	no usado	
8 IRQ0	8	tiempo	
9 IRQ1	9	teclado	
10 IRQ2	A	no usado	
11 IRQ3	B	COM1	Líneas de interrupción 8259A
12 IRQ4	C	COM2	
13 IRQ5	D	no usado	
14 IRQ6	E	diskette	
15 IRQ7	F	LPT1	
16	10	video I/O	
17	11	equipo	
18	12	memoria	
19	13	disco I/O	
20	14	serial I/O	Puntos de Entrada BIOS
21	15	cassette	
22	16	teclado I/O	
23	17	impresora	
24	18	BASIC residente	
25	19	*bootstrap*	
26	1A	tiempo	

27	1B	interrupción de teclado	
28	1C	tiempo	
29	1D	instalación de video	Rutinas
30	1E	instalación de disco	Usuario
31	1F	video Gráfico	
32	20	programa DOS	
33	21	llamada de función DOS	
34	22	dirección DOS	
35	23	error fatal DOS	
36	24	Ctrl Brk exit DOS	Interrup-
37	25	leyendo en disco DOS	ciones
38	26	escribiendo en disco DOS	de DOS y
39-63	27-3F	reservado DOS	de BASIC
64-95	40-5F	reservado	
96-103	60-67	reservado para Usuario	
104-127	68-7F	disponible	
128-133	80-85	reservado para Basic	
134-240	86-F0	reservado para Basic	Dirección
241-255	F1-FF	no usado	003FFH

APENDICE C.

ASIGNACION DE SEÑALES DE LAS TERMINALES DE LOS CONECTORES DE LAS TARJETAS.

1. Conector de la tarjeta de extensión del SLOT.

TERMINAL	SEÑAL	TERMINAL	SEÑAL
1	GND	2	A0
3	OSC	4	A1
5	+5V	6	A2
7	+5V	8	A3
9	ALE	10	A4
11	T/C	12	A5
13	IRQ3	14	A6
15	CLOCK	16	A7
17	AEN	18	A8
19	-IOR	20	A9
21	-IOW	22	IOCHRDY
23	-MEMR	24	D0
25	-MEMW	26	D1
27	+12V	28	D2
29	-12V	30	D3
31	-5V	32	D4
33	IRQ2	34	D5
35	+5V	36	D6
37	RESETDRV	38	D7
39	GND	40	-IOCHCK

2. Conector de la tarjeta de control de procesos (8 bits).

TERMINAL	SEÑAL	TERMINAL	SEÑAL
1	GND	2	GND
3	V(-)	4	V(+)
5	REL0	6	REL7
7	REL1	8	REL6
9	REL2	10	REL5
11	REL3	12	REL4
13	OP0	14	GND0
15	OP7	16	GND7
17	OP1	18	GND1
19	OP6	20	GND6
21	OP2	22	GND2
23	OP5	24	GND5
25	OP3	26	GND3
27	OP4	28	GND4
29	ADC7	30	ADC6
31	ADC5	32	ADC4
33	ADC3	34	ADC2
35	ADC1	36	ADC0
37	VCC	38	GND
39	VCC	40	GND

V(+), V(-): Señal del convertidor D/A.
REL0-REL7: Señales de los relevadores.
OP0-OP7: Señales de optoacopladores.
ADC0-ADC7: Señales del convertidor A/D.

3.-Conector de la tarjeta de procesos J2 (12 bits).

TERMINAL	SEÑAL	TERMINAL	SEÑAL
1	ADC3	2	ADC2
3	ADC1	4	ADC4
5	DAC0	6	NC
7	REL7	8	REL5
9	REL3	10	REL1
11	REL0	12	REL2
13	REL4	14	REL6
15	GND	16	NC
17	VCC	18	NC
19	OP1	20	OP3
21	OP5	22	OP7
23	OP4	24	OP6
25	OP0	26	OP2

REL0-REL7: Señales de los relevadores.
OP0-OP7: Señales de optoacopladores.
ADC1-ADC3: Señales del convertidor A/D.
DAC0: Señal del convertidor D/A.

4. Conectores de la tarjeta de interfase de los motores de pasos.

Conector J3 (motor X).

TERMINAL	SEÑAL
1	FASE1A
2	FASE1B
3	FASE2A
4	FASE2B

Conector J4 (sensor X).

TERMINAL	SEÑAL
1	SALIDA
2	GND
3	ENTRADA
4	GND

Conector J5 (motor Y).

TERMINAL	SEÑAL
1	FASE1A
2	FASE1B
3	FASE2A
4	FASE2B

Conector J6 (sensor Y).

TERMINAL	SEÑAL
1	SALIDA
2	GND
3	ENTRADA
4	GND

Conector J7 y J8 (relevadores).

TERMINAL	SEÑAL
1	REL1
2	REL2
3	REL3
4	REL4

Conector J9 (optoacopladores).

TERMINAL	SEÑAL	TERMINAL	SEÑAL
1	GND1	2	OP1
3	GND2	4	OP2
5	GND3	6	OP3
7	GND4	8	OP4
9	GND5	10	OP5
11	GND6	12	OP6

Conector J10 (convertidor D/A).

TERMINAL	SEÑAL
1	GND
2	DAC

Conector J11 (convertidor A/D).

TERMINAL	SEÑAL
1	GND
2	ADC1
3	ADC2
4	ADC3
5	ADC4

APENDICE D.

TABLAS DE PROGRAMACIÓN DE LOS TIBPAL16L8.

1. Asignación de terminales.

PAL TARJETA DE 8 BITS				PAL TARJETA DE 12 BITS			
N°	SEÑAL	N°	SEÑAL	N°	SEÑAL	N°	SEÑAL
1	S9	20	Vcc	1	A4	20	Vcc
2	S8	19	ST	2	A5	19	NC
3	S7	18	CS	3	A6	18	A3
4	S6	17	P2	4	A7	17	F
5	S5	16	P1	5	A8	16	S4
6	AEN	15	WR	6	A9	15	S3
7	A9	14	RD	7	IR	14	AS
8	A8	13	A5	8	IW	13	PS
9	A7	12	OE	9	AEN	12	TS
10	GND	11	A6	10	GND	11	NC

2. Funciones.

a) Tarjeta con 12 bits de resolución.

F=A9'+A8'+A7+A6+A5+AEN
TS=FIW+FIR
PS=TA4A3'
AS=TS'A4A3
S3=TS'A4'A3
S4=TSA4'A3'

b) Tarjeta con 8 bits de resolución.

P1=S9'A9+S9A9'+S8'A8+S8A8'+S7'A7+S7A7'+S6'A6
P2=S6A6'+S5'A5+S5A5'+AEN+RDWR
ST=CS'WR'
OE=CS'RD'

3. Mapas de fusibles.

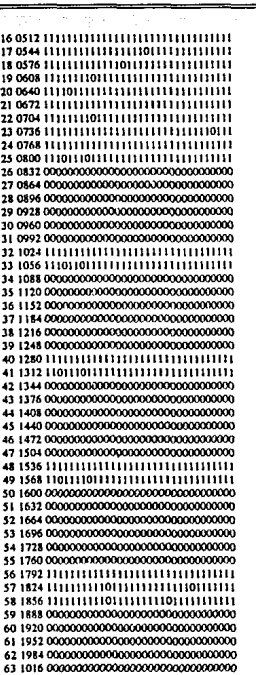
a) Tarjeta de 8 bits de resolución.

```
00 0000 00000000000000000000000000000000
01 0032 00000000000000000000000000000000
02 0064 00000000000000000000000000000000
03 0096 00000000000000000000000000000000
04 0128 00000000000000000000000000000000
05 0160 00000000000000000000000000000000
06 0192 00000000000000000000000000000000
07 0224 00000000000000000000000000000000
08 0256 00000000000000000000000000000000
09 0288 00000000000000000000000000000000
10 0320 00000000000000000000000000000000
11 0352 00000000000000000000000000000000
12 0384 00000000000000000000000000000000
13 0416 00000000000000000000000000000000
14 0448 00000000000000000000000000000000
15 0480 00000000000000000000000000000000
16 0512 11111111111111111111111111111111
17 0544 11111111111111101111111111111111
18 0576 11111111101111111111111111111101
19 0608 11111111111101111111111111101111
20 0640 11111111111111101111111111111111
21 0672 11111111111111111011101111111111
22 0704 00000000000000000000000000000000
23 0736 11111111111101111111111011111111
24 0768 11111111111111111111111111111111
25 0800 11011111111111111111011111111111
26 0832 11101111111111111111101111111111
```

27 0864 01111111111111111111111110111111
28 0896 10111111111111111111111101111111
29 0928 11110111111111111111111111111011
30 0960 11111011111111111111111111110111
31 0992 11111111101111111111111111111110
32 1024 00000000000000000000000000000000
33 1056 00000000000000000000000000000000
34 1088 00000000000000000000000000000000
35 1120 00000000000000000000000000000000
36 1152 00000000000000000000000000000000
37 1184 00000000000000000000000000000000
38 1116 00000000000000000000000000000000
39 1248 00000000000000000000000000000000
40 1280 00000000000000000000000000000000
41 1212 00000000000000000000000000000000
42 1344 00000000000000000000000000000000
43 1376 00000000000000000000000000000000
44 1408 00000000000000000000000000000000
45 1440 00000000000000000000000000000000
46 1472 00000000000000000000000000000000
47 1504 00000000000000000000000000000000
48 1534 00000000000000000000000000000000
49 1568 00000000000000000000000000000000
50 1600 00000000000000000000000000000000
51 1632 00000000000000000000000000000000
52 1664 00000000000000000000000000000000
53 1696 00000000000000000000000000000000
54 1628 00000000000000000000000000000000
55 1760 00000000000000000000000000000000
56 1792 11111111111111111111111111111111
57 1724 11111111011111111111111101111111
58 1856 00000000000000000000000000000000
59 1888 00000000000000000000000000000000
60 1820 00000000000000000000000000000000
61 1952 00000000000000000000000000000000
62 1984 00000000000000000000000000000000
63 2016 00000000000000000000000000000000

b) Tarjeta de 12 bits de resolución.

00 0000 00000000000000000000000000000000
01 0032 00000000000000000000000000000000
02 0064 00000000000000000000000000000000
03 0096 00000000000000000000000000000000
04 0128 00000000000000000000000000000000
05 0160 00000000000000000000000000000000
06 0192 00000000000000000000000000000000
07 0224 00000000000000000000000000000000
08 0256 00000000000000000000000000000000
09 0288 00000000000000000000000000000000
10 0320 00000000000000000000000000000000
11 0352 00000000000000000000000000000000
12 0384 00000000000000000000000000000000
13 0416 00000000000000000000000000000000
14 0448 00000000000000000000000000000000
15 0480 00000000000000000000000000000000


16 0512
17 0544
18 0576
19 0608
20 0640
21 0672
22 0704
23 0736
24 0768
25 0800
26 0832
27 0864
28 0896
29 0928
30 0960
31 0992
32 1024
33 1056
34 1088
35 1120
36 1152
37 1184
38 1216
39 1248
40 1280
41 1312
42 1344
43 1376
44 1408
45 1440
46 1472
47 1504
48 1536
49 1568
50 1600
51 1632
52 1664
53 1696
54 1728
55 1760
56 1792
57 1824
58 1856
59 1888
60 1920
61 1952
62 1984
63 1016

BIBLIOGRAFIA.

MOTOROLA.
Linear and Interface Integrated Circuits.
Second Printing 1988.

MOTOROLA.
Optoelectronics.
Device Data, INC 1989.

MOTOROLA.
Bipolar power.
transistor Data, 1989.

National Semiconductor.
General Purpose Linear Devices.
Data book, 1989.

National Semiconductors.
Special Purpose Linear Devices.
Data book, 1989.

National Semiconductors.
Data Acquisition Linear Devices.
Data book, 1989.

TEXAS Instruments.
TTL Logic.
Data book, 1988.

PRECISION MONOLITHICS INCORPORATED.
Full Line Catalog.
1980.

BURR-BROWN
General Catalog.
1979.

HARRIS.
CMOS Digital Data Book
1984.

S. W. AMOS.
Diccionario de Electrónica.
Edit. Paranindo, 1988.

COOPER William David.
Instrumentación Electrónica y Mediciones.
Edit. Prentice Hall, 1982.

O'BRIEND Stephen.
Turbo Pascal 6.
Manual de referencia.
Edit. McGraw-Hill, 1992.

BOYLESTAD, Nashelsky.
Electrónica Teoría de Circuitos.
Edit. Prentice Hall 1988.

TOOLEY Mike.
PC-Based instrumentation and control.
Edit. Newnes BH, 1991.

TOMPKINS Willis J., WEBSTER John G.
Interfacing sensors to the IBM PC.
Edit. Prentice Hall, 1988.

OLLERO Baturone Aníbal.
Control por computadora.
Edit. Marcombo.

KOU Benjamín C.
Sistemas Automáticos de Control.
Edit. CECSA.

REFERENCIAS.

(1) TOOLEY Mike, PC-Based instrumentation and control, pag. 259.

(2) OLLERO Baturone Anfbal, Control por computadora, pag. 2.

(3) OLLERO Baturone Anfbal, Control por computadora, pag. 5.

(4) TOOLEY Mike, PC-Based instrumentation and control, pag. 15.

(5) TOOLEY Mike, PC-Based instrumentation and control, pag. 37.

(6) TOOLEY Mike, PC-Based instrumentation and control, pag. 41.

(7) TOOLEY Mike, PC-Based instrumentation and control, pag. 47.

(8) TOMPKINS Willis, Interfacing sensors to the IBM PC, pag. 63.

(9) HARRIS, CMOS data booK, pags. 4-35 a 4-45.

(10) BURR-BROWN General Catalog, pags. 6-123 a 6-129.

(11) O´BRIEN Stephen, Turbo Pascal 6, pag. 505.