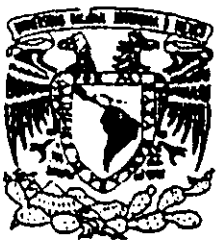


UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CONTROL VIRTUAL DE UN POSICIONADOR BIDIMENSIONAL

TESIS

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA
AREA: ELECTRICA – ELECTRONICA

PRESENTA:

WALTER FELIPE KEMPER CASTRO

DIRECTOR DE TESIS: M.I. NICOLAS KEMPER VALVERDE

MEXICO, D.F. 1.999

A mis padres Trine y Grima.
A mis hermanos Henry, Norma y Gladis. Porque son la verdadera inspiración en mi vida.

TEMARIO

Introducción

En el año de 1986 National Instruments introdujo al mercado LabVIEW y con el una nueva filosofia de medición basado en PC que integra la adquisición, análisis y visualización de señales en una sola unidad denominado instrumento virtual. Actualmente la capacidad de la instrumentación se ha incrementado debido al hardware y software especifico diseñado para prueba, medición y control de procesos industriales especificos.

La programación en Lenguaje G ha ganado gran aceptación porque permite desarrollar aplicaciones de una manera fácil, rápida e intuitiva. Como elementos demostrativos sobre el control de movimiento en National Instruments, se han construido instrumentos virtuales (VIs) para el manejo de motores de paso. Estos VIs están desarrollados en LabVIEW y son implementados sobre plataformas PXI. Por lo cual la adquisición de estas herramientas es costosa y muchas veces la resolución que maneja sobrepasa nuestras necesidades como la de construir un posicionador bidimensional para un taladro electrónico de circuitos digitales. También existen módulos de software y hardware integrados que controlan el desplazamiento lineal en posicionadores de una, dos y tres dimensiones y su costo es también bastante elevado.

Ante este panorama se propuso desarrollar una herramienta llamada VI MOTOR para el control de un posicionador bidimensional manejado con motores de paso. El VI MOTOR fue desarrollado en LabWindows/CVI que es un paquete de instrumentación virtual cuya programación es en ANSI C. El instrumento virtual integra una computadora personal como elemento de control, una Tarjeta I/O de propósito general programable como elemento de interface, un Driver Unipolar como elemento de potencia y un Posicionador Bidimensional para el desplazamiento exacto y preciso de un punto sobre el plano cartesiano.

Los objetivos que se persiguieron han sido lograr precisión y exactitud en el movimiento de un punto sobre un plano cartesiano y es por eso que se usaron motores de paso. Las ventajas más importantes de estos dispositivos electromecánicos es su alta precisión, control con señales digitales TTL y su capacidad para ser trabajados en lazo abierto. La eliminación de una retroalimentación y la capacidad de LabWindows/CVI de permitirnos construir por medio de software funciones de un elemento de hardware, nos redujo circuiteria y además nos brinda la posibilidad de usar una Tarjeta I/O digital de bajo costo implementado con elementos encontrados muy fácilmente en el mercado mexicano. La capacidad del VI MOTOR es totalmente flexible y su funcionamiento es definido programaticamente por el diseñador del instrumento virtual.

La interface hombre-máquina del VI MOTOR es muy amigable por su ambiente Windows. Cuenta con módulos para programar la Tarjeta I/O, Control Manual y Automático del posicionador bidimensional sobre la pantalla de una computadora personal. Los costos son económicos ya que utiliza circuitos electrónicos TTL y sobre todo es una tecnología que accesible para desarrollar aplicaciones mayores en el campo de la medición y control de procesos industriales.

INSTRUMENTACION VIRTUAL

1.1 Evolución de la Instrumentación

La primera generación de instrumentos fueron dispositivos análogos auto contenidos, es decir que con una señal de entrada / salida y una interface de ajuste constituido por interruptores, perillas y otros dispositivos se podía controlar el circuito especializado, el cual se encontraba en el interior de la caja. Este circuito podía incluir convertidores A/D, acondicionadores de señal, microprocesadores, memoria y un bus interno que convirtiera las señales reales. Este dispositivo leía la señal y le daba una presentación numérica o gráfica al usuario. Por lo cual se puede decir que el fabricante definía todas las funciones del instrumento, clasificándolo como un instrumento de arquitectura cerrada.

A partir de la década de los 80 varias empresas productoras de software idearon dispositivos que fortifican el desarrollo de sistemas de ingeniería. Con la invención del GPIB e instrumentos digitales, los usuarios pudieron controlar los sistemas ya sea de una manera manual y por programación. Cada instrumento GPIB fue diseñado para una medición específica, los usuarios tuvieron que juntar y amontonar muchos instrumentos para crear un sistema de medición completo. El usuario logra realizar mediciones controladas por la computadora por medio de secuencias de comandos.

Actualmente los instrumentos tienen una arquitectura abierta para ser conectadas a una computadora de propósito general. Estos instrumentos modernos pueden alcanzar altos niveles de funcionamiento y flexibilidad si se controla con software dentro de una PC. Con las nuevas técnicas para el control de un instrumento tales como programación basada en registros y memoria compartida, se puede aprovechar al máximo la potencia de la PC para adquirir, analizar y presentar las mediciones realizadas.

Por lo tanto el nuevo término que describe esta nueva generación de instrumentos basados en la PC se le conoce como Instrumentación Virtual. Con los Instrumentos Virtuales, se puede dar una amplia variedad de dotación física al instrumento en un sistema y modificar totalmente las funciones originales del instrumento por medio de una interfaz gráfica de software orientado a Instrumentación Virtual

1.2 Instrumento Electrónico Tradicional

Es una caja independiente compuesta internamente de tres elementos: Adquisición, Análisis y Presentación de Datos (Ver Figura 1.1). El límite de funcionamiento de estos instrumentos está determinado por el fabricante.

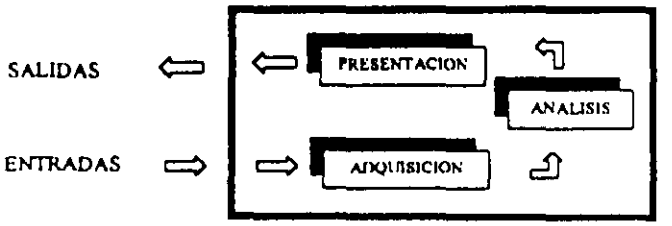


Figura 1.1 Instrumento electrónico tradicional

1.3 Instrumento Virtual

Un Instrumento Virtual es definido como una capa de software y hardware que se le agrega a una PC de tal forma que permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida" (Ver Figura 1.2).

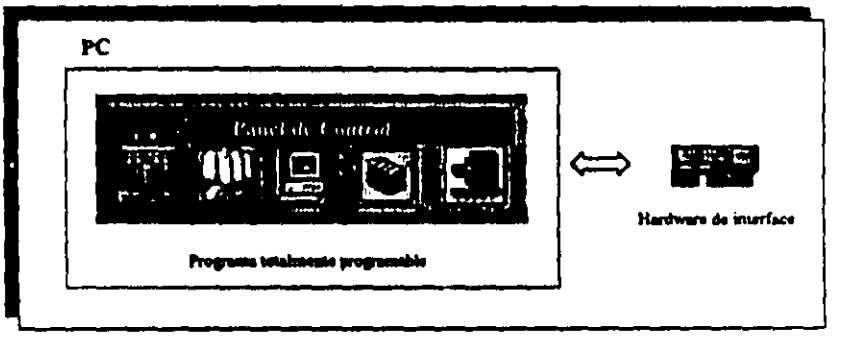


Figura 1.2 Instrumento Virtual

1.3.1 Elementos de un Instrumento Virtual

Al igual que un instrumento tradicional un instrumento virtual consta de tres partes: Adquisición, Análisis y Presentación de Datos. Todos estos componentes son muy flexibles, configurables por el usuario y no necesariamente residen en una unidad. En la Figura 1.3 se muestra los elementos de un instrumento virtual basado en PC y algunos componentes que lo forman.

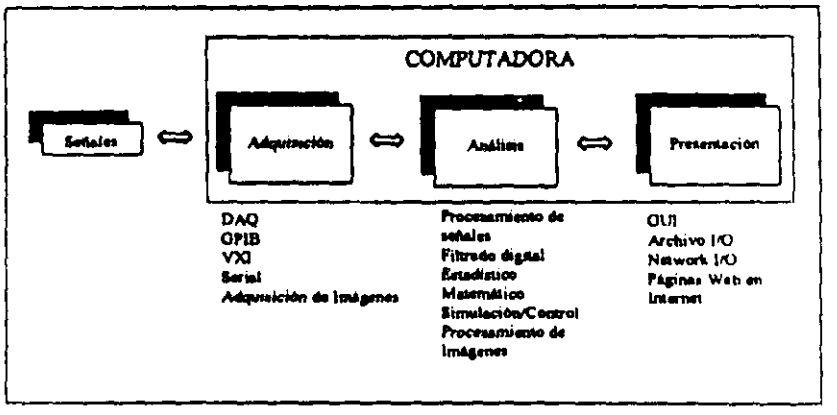


Figura 1.3 Elementos de un Instrumento Virtual

1.3.2 Formas de desarrollar un Instrumento Virtual

En la Figura 1.4 se presenta las diversas formas en que se puede desarrollar un instrumento virtual:

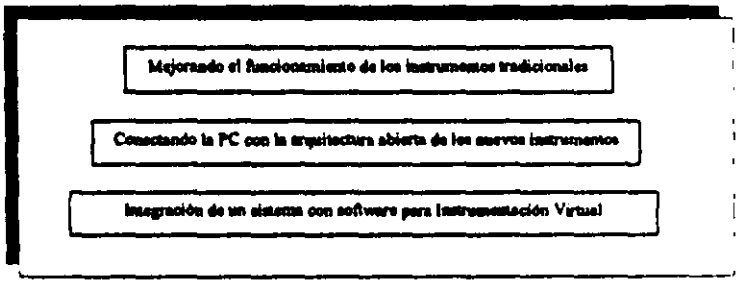


Figura 1.4 Formas de desarrollar un Instrumento Virtual

1.3.2.1 Mejorando el funcionamiento de los instrumentos tradicionales

Esto es posible con la estandarización del GPIB (Bus IEEE 488), RS232 y las herramientas de software que incluyen librerías driver del instrumento, librerías de *análisis* y librerías gráficas que permiten diseñar de una manera sencilla interfaces de usuario sobre la pantalla de una PC.

Las tendencias que ocasionó fueron:

- La poca demanda de instrumentos análogos tradicionales por carecer de un bus de comunicación estandarizado.
- El aumento en la demanda de digitalizadores GPIB, ya que junto con una computadora y herramientas de software se pueden crear sus propios instrumentos de medición.

1.3.2.2 Conectando la PC con la arquitectura abierta de los nuevos instrumentos

Como resultado de analizar los componentes usados en la fabricación de los *instrumentos* tradicionales y comparar estos componentes funcionando con la arquitectura de una PC. Por lo tanto los instrumentos modernos tienen la arquitectura de una PC, el fabricante estandariza sus productos con su propio procesador y diseño de memoria, así como librerías de software para la adquisición, análisis y presentación de los datos. Pero lo más importante es que tienen una sistema abierto con salidas para comunicación con la arquitectura de una PC y permitir crear nuestros propios instrumentos virtuales.

Las tendencias que ocasionó fueron:

- La demanda por las tarjetas PC DAQ y el VXIbus que sirven como interfaces programables entre la PC y el sistema de medición.

1.3.2.3 Integración de un sistema con software para Instrumentación Virtual

Este tipo de software vino a impulsar la instrumentación virtual. Este software combina drivers en bajo nivel que sirve de interface con los instrumentos modernos y permite crear paneles de control en lenguaje de alto nivel.

Las tendencias que ocasionó fueron:

- La aparición de software para instrumentación virtual sobre ambiente windows.
- Programación en lenguaje G y además la programación orientada a eventos usando C.

1.4 Comparación entre un Instrumento Tradicional Vs un Instrumento Virtual

Algunas de las principales diferencias entre el instrumento tradicional y el instrumento virtual se resumen en la Tabla 1.1:

Tabla 1.1

Instrumento Tradicional	Instrumento Virtual
Definido por el fabricante	Definido por el usuario
Funcionalidad específica y conectividad limitada	Funcionalidad ilimitada y conectividad amplia
Hardware es la clave	Software es la clave
Alto costo / función	Bajo costo / función, variedad de funciones, reusable
Arquitectura cerrada	Arquitectura abierta
Lenta incorporación de nuevas tecnologías	Rápida incorporación de nuevas tecnologías, gracias a la plataforma PC
Bajas economías de escala. Alto costo de mantenimiento	Altas economías de escala, bajos costos de mantenimiento

Los instrumentos virtuales también se pueden implementar en equipos móviles (laptops), equipos distribuidos en campo (RS-485), equipos a distancia (conectados vía radio, internet) o equipos industriales. Existe una tarjeta de adquisición de datos para casi cualquier bus o canal de comunicación en PCs (ISA,PCI,USB, serial RS-232/485, paralelo EPP, PCMCIA, CompactPCI). Mención a esto se presenta en la Tabla 1.2 un cuadro comparativo entre un multímetro digital (DMM Digital Multi-Meter) y un multímetro virtual (VMM Virtual Multi-Meter):

Tabla 1.2

	DMM	VMM con tarjeta especializada	VMM con tarjeta de propósito general
Hardware utilizado	HP 34401 A DMM	DAQ Card 4050	PCI-MIO-16XE-10
Numero de canales	1	1	16
Conversión AC	RMS	RMS	RMS (por software)
Resolución (convertidor de 16 bits)	6 ½ - 4 ½ dígitos	5 ½ dígitos	4 ½ dígitos
Rango de entrada (AC voltaje)	100 mV - 750 V	20 mV - 250 V	100 mV - 250 V (con acondicionamiento SCXI)
Sensibilidad (AC voltaje)	0.1 uV	0.1 uV	1.5 uV
Rango de entrada (DC voltaje)	100 mV - 1000V	20 mV - 250 V	100 mV - 250 V
Sensibilidad (DC voltaje)	0.1 uV	0.1 uV	1.5 uV
NMRR	60 dB	80 dB	Variable (80-120 dB)
CMRR	70 dB (AC), 140 dB (DC)	90 dB (AC), 30 dB (DC)	Variable (80-120 dB)
Velocidad de medición (lecturas/seg)	5 - 1 K lecturas/seg	10, 50, 60 K lecturas/seg	100 K lecturas/seg

1.5 Instrumentación Virtual

El concepto de instrumentación virtual nace a partir del uso de la computadora personal como "Instrumento" de medición de señales tales como:

- Temperatura
- Presión
- Caudal

La PC comienza a ser utilizado para realizar mediciones de fenómenos físicos representados en señales de corriente (4-20 mA) y/o voltaje (0-5 V).

Sin embargo, el concepto de "Instrumentación Virtual" va más allá de la simple medición de corriente o voltaje, si no que *también* involucra el procesamiento, análisis, almacenamiento, distribución, y despliegue de los datos e información relacionado con la medición de una o varias señales específicas. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz hombre-máquina, la función de análisis y procesamiento de la señal, las rutinas de almacenamiento de datos, generación de señales de control y la comunicación con otros equipos.

Por ejemplo un osciloscópio tradicional tiene una funcionalidad pre-definida desde la fábrica donde lo diseñan, producen y ensamblan. La funcionalidad de este tipo de *instrumento* no es definida por el usuario mismo. El término "Virtual" nace precisamente a partir del hecho de que cuando se utiliza la PC como "Instrumento" es el usuario mismo quién, a través de software, define su funcionalidad y "Apariencia" y por ello decimos que "Virtualizamos" el instrumento, ya que su funcionalidad puede ser modificada una y otra vez por el usuario y no por el fabricante.

La Instrumentación Virtual es una nueva filosofía de control donde el software es el instrumento. La Instrumentación Virtual integra la Adquisición, Análisis y Despliegue de los instrumentos reales en una sola unidad denominada Instrumento Virtual.

1.5.1 Estructura de un Sistema de Control Virtual

Con este nuevo paradigma y como lo *muestra* la Figura 1.5 la configuración de un sistema de control tiene tres partes bien definidas:

- Software para Instrumentación Virtual
- Computadora
- Interfaces

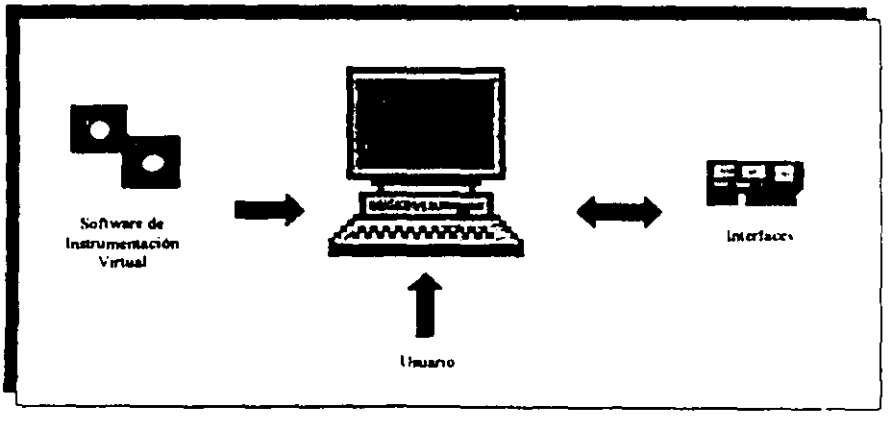


Figura 1.5 Configuracion de un Sistema de Control Virtual

1.5.1.1 Software para Instrumentación Virtual

Es el elemento de programación visual orientado a eventos. Los lenguajes de programación más comunes son:

- Programación en lenguaje G
- Programación en lenguaje C

También existen en el mercado aplicaciones que permiten utilizar Visual Basic, Visual C++, Borland Delphi o Excel en instrumentación virtual. El software que usa programación en lenguaje G es el más aceptado debido a su fácil programación que consiste básicamente en el alambrado de elementos, cuyas funciones ya vienen definidas y sólo es necesario ingresar un identificador y algunos parámetros.

1.5.1.2 Computadora

Es el elemento donde se diseña y reside el software del Instrumento Virtual. La PC es una herramienta relativamente poderosa que puede ser aplicada en ciertos procesos de control en tiempo real. Esto ha sido posible por el desarrollo de:

- Buses PCI para una mayor velocidad en la transmisión de datos
- Software de Instrumentación Virtual

1.5.1.3 Interfaces

Son los elementos que permiten adaptar las señales del mundo real en señales digitales y viceversa.

- Módulos Acondicionamiento de Señal
- Tarjetas DAQ
- GPIB
- Serial
- PLC
- Fieldbus
- Visión

1.6 Herramientas de Software

1.6.1 LabVIEW

Sistema de programación gráfico para la adquisición, análisis y presentación de señales en procesos de medición y automatización. LabVIEW tiene una nueva metodología de programación llamado Lenguaje G. Esta nueva metodología de programación se basa en el alambrado de controles, indicadores y otros componentes que modelan un sistema de medición y automatización. Los programas son llamados instrumentos virtuales (VIs) e interactuan con el hardware integrado diseñado especificamente para trabajar la instrumentación virtual.

Entre las principales caracteristicas se nombra:

Controles e indicadores:

- Botones / Switches / LEDs
- Slides / Diplay digitales
- Gauges / Dials / Knobs
- Tanks / Termómetros
- Graphs / Charts
- Tablas / Arreglos
- Menús / Listas / Rings
- Cajas de Texto
- Decoraciones
- Controles personalizados

Hardware integrado y librerías para:

- Instrumentos basados en computadora GPIB/VXI/PXI
- Protocolo RS-232/485
- Adquisición de datos Plug-in
- Acondicionamiento de señal
- Adquisición de datos distribuido
- Adquisición de imágenes y Visión
- Control de movimiento
- PLCs

Librerías de análisis:

- Matemática
- Estadística
- Regresión
- Algebra lineal
- Algoritmos para la generación de señales
- Algoritmos para análisis en el dominio del tiempo y la frecuencia
- Filtros digitales
- Métodos de Optimización

Salida de Archivos:

- Spreadsheet
- Binario / ASCII

Conectividad Abierta:

- Internet
- SQL
- DataSocket
- TCP/IP
- ActiveX / DDEs
- DLLs

Estructuras de programación:

- Lazos While / For
- Estructuras Case
- Estructuras Secuenciales

Fundamentos de programación:

Computación numérica
Lógica Boolena
Manipulación de arreglos y cadenas
Funciones de Time y Date
Estructuras de datos múltiples
Subrutinas personalizadas

Visualización:

Plots 3D
Superficie mesh de contorno
Gráficos personalizados y Animación

Herramientas para Lógica Difusa:

Funciones de membresía:

Triangular
Trapezoidal
Forma de Z
Forma de S
Singleton

Métodos de defuzificación:

Centro de Area
Centro de Máximo
Media de Máximo

LabVIEW es muy utilizado en aplicaciones:

- GPIB
- Instrumentación
- Automatización
- Simulación
- Científicas
- Control de Movimiento

1.6.2 BridgeVIEW

Ambiente de programación gráfico que ofrece flexibilidad en aplicaciones de medición y automatización industrial. BridgeVIEW utiliza como elemento de programación a LabVIEW, pero además cuenta con el manejo de eventos, alarmas, Vis, DDEs, Base de datos en tiempo real y una total conectividad OPC, para diseñar paneles de visualización y control de un proceso industrial completo.

Entre las principales características se nombra:

Alarmas y Eventos:

BridgeVIEW automáticamente calcula y registra la información de alarmas dentro del sistema. Las condiciones de las alarmas se pueden configurar y usar programación gráfica para desarrollar esquemas más sofisticados. Los niveles de alarma dependen de su configuración y de los VIs usados para cambiar programaticamente estos niveles. Los operadores son avisados por medio de despliegues gráficos o de audio.

Registro histórico de datos y tendencias:

BridgeVIEW permite registrar los datos, tendencias y capacidades de visualización gráfica en tiempo real. Toda la información histórica reside en la base de datos Citadel ODBC, así como también se puede usar otras herramientas de base de datos.

BridgeVIEW es muy utilizado en aplicaciones:

- SCADA
- GPIB
- Instrumentación
- Control de Procesos
- Automatización
- Simulación
- Científicas

1.6.3 Lookout

Software de automatización basado en objetos para el Control de Procesos SCADA. Entre sus principales características se mencionan:

Manejo de eventos:

La ejecución de un objeto dentro de Lookout, no usa el reloj del CPU hasta que ha ocurrido un evento. Un evento puede ser disparado por el cambio de un dato, de un switch o el encendido de un tiempo. Entonces cuando ha ocurrido el evento, la señal se propaga a través de la red de conexión de los objetos y se crea una cadena de reacción con los objetos relevantes en el programa de control.

Configuración en línea:

Lookout permite hacer cualquier tipo de cambio dentro de una aplicación sin tener que parar el proceso. Se puede agregar, borrar o modificar dispositivos, puntos de I/O, registros de base de datos, graficos y control lógico mientras se está monitoreando y controlando el proceso.

Registro de datos:

Lookout tiene dos formas para registrar los datos. Una es por medio de Citadel, un registro basado en eventos que comprime datos históricos en archivos binarios. La otra forma es por medio de objetos de la clase Spreadsheet, los cuales pueden ser abiertos por ejemplo en Microsoft Excel.

Registros y Charts:

Lookout provee una diversidad de herramientas gráficas que incluyen objetos de las clases HyperTrend y X-Y. Los objetos de la clase HyperTrend visualizan datos en tiempo real, combinando además con datos históricos en un solo gráfico.

SPC:

Es una herramienta que se puede configurar para crear un control de proceso estadístico. Los SPC combinan datos modificados en una base de datos en tiempo real con herramientas de análisis gráfico para generar alarmas que avisen al operario y tome una acción correctiva.

1.6.4 ComponentWorks

Colección de controles contenedores ActiveX a 32 bits para que ingenieros y científicos puedan construir sistemas de instrumentación virtual.

Entre sus principales características mencionaremos:

Herramientas de medición y automatización para lenguajes de programación visual:

ComponentWorks combina la flexibilidad y potencia de herramientas estándar tales como Microsoft Visual Basic, Visual C++, Borland Delphi e Internet Explorer para convertirlos en herramientas de instrumentación virtual. ComponentWorks incluye una librería de controles ActiveX para construir interfaces gráficas de usuario (GUI). La interface de usuario incluye controles, indicadores, termometros, displays y otras herramientas que hace posible usar la programación visual en instrumentación virtual.

1.6.5 Measure

Herramienta que permite usar Microsoft Excel en la adquisición de datos desde dispositivos DAQ, GPIB y serial RS-232. Entre sus principales características mencionaremos:

Productividad para usuarios de Hojas de Cálculo:

La hoja de cálculo es una de las herramientas más usadas y por medio de Measure los ingenieros incrementan su productividad integrando una colección de datos directamente dentro de Microsoft Excel. Measure hace que los datos adquiridos desde los dispositivos externos sean colocados dentro de las celdas especificadas por el usuario y realizar su análisis deseado.

No requiere programación:

Measure nos proporciona herramientas para configurar nuestros dispositivos de adquisicion de datos. Toma las lecturas y lo almacena dentro de una hoja de cálculo en Excel.

1.6.6 HiQ

Ambiente interactivo donde se analiza, visualiza y documenta problemas matemáticos, científicos y de ingeniería en tiempo real. Entre sus principales características mencionaremos:

Metodología Notebook:

HiQ soluciona problemas del mundo real usando una metodología notebook que combina el análisis interactivo, visualización de datos y la generación de reportes en un solo ambiente.

ActiveMath para Microsoft Office:

Al crear los reportes técnicos en HiQ facilita el uso de la tecnología ActiveX (OLE) disponible bajo WindowsNT/95. Estos reportes pueden ser incrustados en Microsoft Word, Excel, PowerPoint y Paintbrush para hacer que los cálculos técnicos, presentaciones y reportes sean más informativos.

Visualización de datos en 2D y 3D:

HiQ posee herramientas de visualización 2D y 3D. Estos gráficos manejan coordenadas cartesianas, polares y cilíndricas, escala lineal y logarítmica, animación, zooming y sobre todo el manejo de funciones matemáticas de ecuaciones diferenciales y su visualización para simulación.

1.6.7 VirtualBench

Conjunto de Instrumentos Virtuales desarrollados en LabView para usar con un DAQ. Incluye instrumentos virtuales tales como:

Osciloscopio:

El VirtualBench Scope ofrece muestreos en tiempo real de 20 MHz, ancho de banda de 15 MHz, resolución vertical de 16 bits (dependiendo del tipo de DAQ usado). Tiene 16 canales y un rango de sensitividad de 1 mV/div hasta 10 mV/div, un tiempo base del rango de 10 ns/div hasta 100 ms/div. La capacidad de almacenamiento es de 500 a 660,000 puntos.

Analizador Dinámico de Señales:

El VirtualBench DSA mide la distorsión armónica total, respuesta en frecuencia, respuesta impulso, espectro de potencia, espectro de la amplitud y el espectro de potencia transversal. Los despliegues se hacen en unidades de Vpk, Vpk2, Vrms, Vrms2, dB, dBm, dBVrms, etc.

Generador de Señales:

El VirtualBench Generador de señales nos proporciona salidas de ondas cuadradas, senoidales, triangulares de diferentes frecuencias y de 2,048 puntos.

Multímetro Digital:

VirtualBench DMM mide voltajes en DC y AC, corrientes en DC y AC, resistencia y temperatura. Sus rangos de entrada son seleccionados de 1 mV hasta 10 V. La temperatura es medida de termocuplas, termistores o sensores en circuito integrado.

Registrador de Datos:

VirtualBench Logger adquiere datos desde 384 canales de datos y despliega hasta 16 canales en tiempo real. Con los menús se puede configurar los parámetros de cada uno de los canales como nombre, factor de conversión, rango de despliegue, escala de despliegue.

1.6.8 LabWindows/CVI

Ambiente de desarrollo de instrumentación virtual para programadores en ANSI C. Con las librerías I/O, rutinas de análisis y las herramientas de interface de usuario es posible crear sistemas de medición y de control muy sofisticados. Entre sus principales características mencionaremos:

Compatibilidad con C/C++:

Las librerías LabWindows/CVI están disponibles como DLLs a 32 bits y son compatibles con compiladores de C/C++ bajo Windows, Unix, incluyendo Microsoft Visual C++, Borland C+, Watcom C++ y Symantec C++. Por lo tanto es opcional el ambiente de desarrollo en C para construir los instrumentos virtuales.

Librerías de instrumentación poderosas:

Cuenta con librerías de Interface de Usuario, Análisis Estadístico, Procesamiento de Señales, Control de Instrumentos VXI,GPIB,RS-232, Adquisición de datos, ActiveX, ANSI C.

Programación con Paneles de Función:

Con el CodeBuilder generamos el cascarón de nuestro código automáticamente. Con las paneles de función se agrega líneas de código al programa, definiendo los parámetros que aparecen en su interface de usuario.

LabWindows/CVI es muy utilizado en aplicaciones:

- GPIB
- Instrumentación
- Control de Procesos
- Automatización
- Simulación
- Científicas

1.7 LabWindows/CVI

Herramienta de desarrollo de instrumentación virtual para programadores en C.

1.7.1 Estructura de un VI

Todo programa de un VI desarrollado en LabWindows/CVI incluye, si no todos, los elementos que se muestran en la Figura 1.6:

- Interfaces gráficas de usuario (GUI)
- Programa de control
- Adquisición de datos
- Análisis de datos

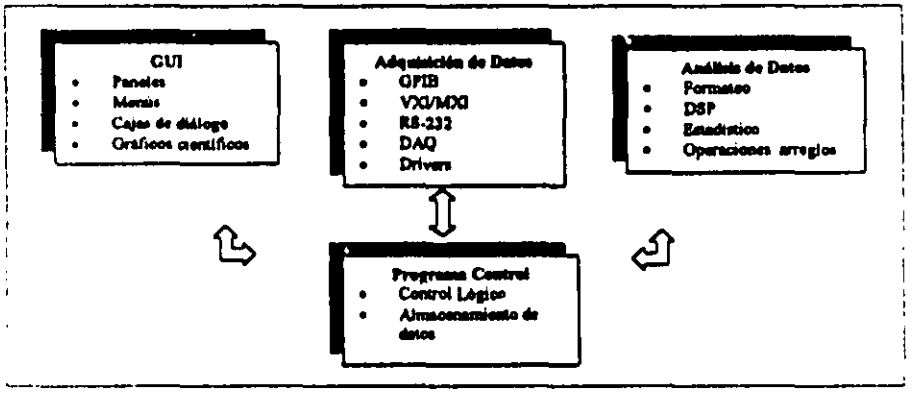


Figura 1.6 Estructura de un Instrumento Virtual con LabWindows

1.7.1.1 GUI

Son las interfaces gráficas de usuario que sirven para diseñar el panel de control de nuestro instrumento virtual

1.7.1.1.1 Elementos

Una GUI puede tener los siguientes elementos:

- Panel
- Barra de menú
- Numérico
- Cadena
- Etiqueta
- Caja de Texto
- Botón Comando
- Botón Toggle
- Led

- Swith
- Ring
- Caja de Lista
- Decoración
- Gráfico
- Imagen
- Timer

1.7.1.1.2 Eventos

Los elementos de una GUI son áreas en la pantalla que pueden activarse por eventos. Cada elemento tiene cierto número de eventos y se le puede asociar una función definida por el diseñador. Al activarse cualquier evento, se transmite información al programa de control y se ejecuta la función correspondiente.

Los eventos de los elementos sobre una GUI son:

- COMMIT
- GOT_FOCUS
- LOST_FOCUS
- LEFT_CLICK
- LEFT_DOUBLE_CLIK
- RIGHT_CLIK
- RIGHT_DOUBLE_CLIK
- VAL_CHANGED
- KEYPRESS
- DISCARD
- TIMER_TICK
- CLOSE
- PANEL_SIZE
- PANEL_MOVE
- EVENT_END_TASK

1.7.1.1.3 Librería GUI

Son funciones que sirven para controlar una interface de usuario. Los elementos que componen las GUI se pueden editar por medio de estas funciones en tiempo de diseño o de ejecución. Estas funciones son escritas en el programa de control y sirven para obtener información o simplemente modificar los atributos del elemento.

La librería GUI se puede usar en conjunción con el editor GUI dentro del ambiente del paquete. En el editor GUI se puede crear todos los elementos y salvar estos objetos en un archivo de extensión (*.uir) Las funciones de la librería GUI permiten cargar este archivo con todos sus elementos para la entrada y despliegue de información. La librería GUI también nos permite crear programaticamente una interface de usuario. Se organizan en los siguientes módulos:

- Paneles. Es una clase de funciones que crea, modifica y descarga paneles definido por el usuario.
- Estructuras de Menú. Es una clase de funciones que crea, modifica y descarga estructuras de menú definidas por el usuario.

- Controles Graphs y Strip Charts. Es una clase de funciones que crea, controla, modifica y descarga Graphs y Strip Charts.
- Paneles Popup. Es una clase de funciones que instala e interactua con las cajas de diálogo predefinidas y definadas por el usuario.
- Funciones Callback. Es una clase de funciones que instala funciones callback definidas por el usuario que responde a los eventos.
- Manejo de la Interface de Usuario. Es una clase de funciones que controla la entrada del usuario y las operaciones de visualización en pantalla.
- Impresión. Es una clase de funciones que configura y genera salidas hacia la impresora.
- Miscelaneas. Es una clase de funciones que no se clasifican dentro de las otras clases.
- Funciones de Compatibilidad LW DOS. Es una clase de funciones que nos proporciona hacer compatibles un VI con Labwindows para DOS.

En el Apéndice A se da una lista detalla de las funciones de librería GUI.

1.7.1.2 Programa de Control

El programa de control es el encargado de coordinar la adquisición de datos, análisis de datos y las GUI El código fuente esta en lenguaje ANSI C.

El flujo de un programa puede ser diseñado de dos maneras: Usando funciones callback y event-loops.

La función callback (Ver Figura 1.7) permite escribir funciones individuales que son llamadas directamente por los controles de la GUI.

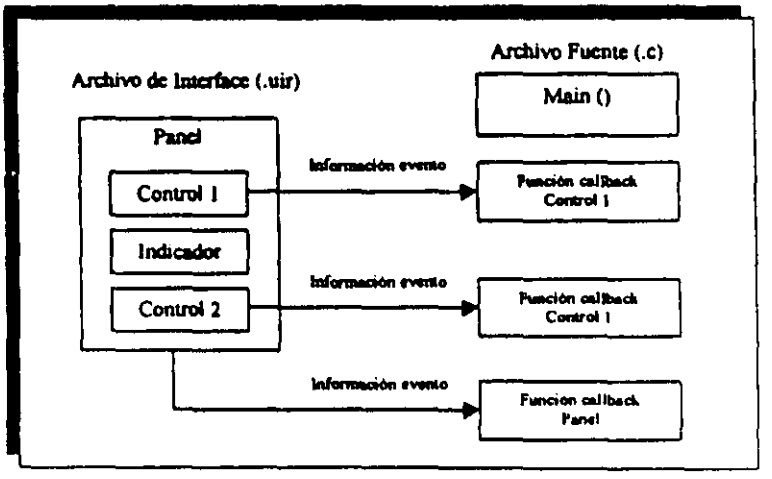


Figura 1.7 Concepto de Función Callback

Los event-loops (Ver Figura 1.8) llama a todos los eventos Commit por medio de la función GetUserEvent y además identifica el control que generó el evento.

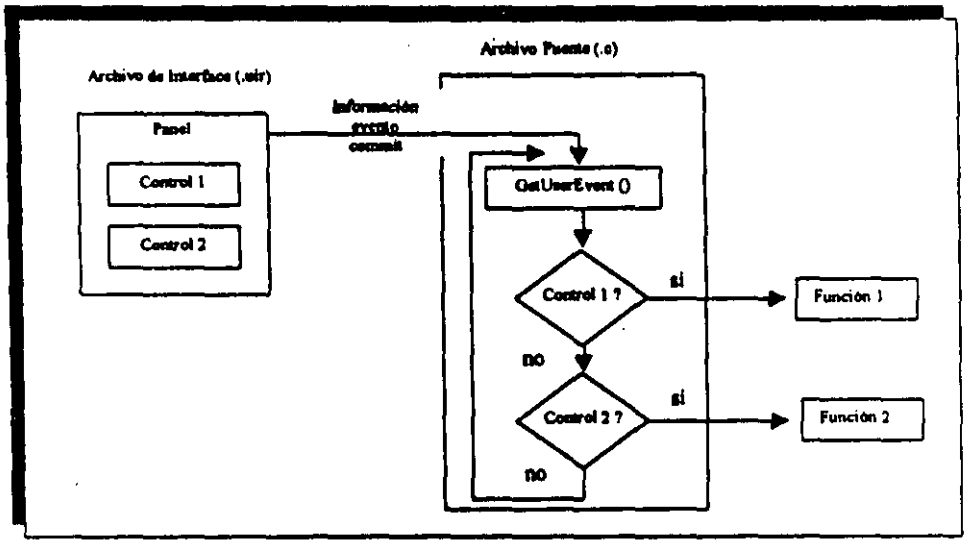


Figura 1.8 Concepto de Event_Loop

1.7.1.3 Adquisición de Datos

Esta compuesta de funciones en alto nivel para controlar las tarjetas insertables de adquisición de datos Junto a las tarjetas de adquisición traen consigo un software driver para programar directamente los registros y además controlar su operación e integración con los recursos de una PC como interrupciones, DMA y memoria.

La selección de una tarjeta DAQ es un punto crítico para la integración de un sistema de adquisición de datos, por lo tanto se presenta los pasos a seguir para una adecuada selección:

i) Identificar el tipo de señal I/O. Al seleccionar un dispositivo DAQ, se debe identificar el tipo de sensor y los tipos de señales I/O que están usándose. Los sistemas DAQ's típicos incluyen.

Entrada análoga
- Temperatura
- Presión
- Tensión
- Voltaje
- Corriente
- Acústica y vibración de señales

Salida análoga
- Voltaje
- Corriente
- Generación de onda

Entrada
- Entrada / Salida compatible con TTL
- Entrada / Salida de voltaje alto AC / DC
- Comunicación Serial / Paralela
- Control de transmisión

Entrada / Salida de tiempo
- Entrada / Salida frecuencia
- Eventos de conteo y cronómetro
- Medición de ancho de pulso
- Generación de tren de pulsos

ii) Seleccionar un método de acondicionamiento de señal. Muchas veces las señales de los sensores deben de acondicionarse antes de conectar a un DAQ. El acondicionamiento de señal usa hardware para amplificación, aislamiento, filtrado, excitación y un alto multiplexaje de canales.

Los productos de acondicionamiento de señal se agrupan en tres clases:

SCXI Acondicionamiento de señal de alto funcionamiento
- Bajo ruido
- Flexible
- Modular
- Amplio rango de I/O
- Alta velocidad de multiplexaje
- Hasta 3,072 canales
- Conexión a una sola tarjeta DAQ.

Series SCC, 5B SC de bajo costo
- Hasta 16 canales
- Modularidad por canal

Tarjetas de propósito general sin acondicionamiento de señal
- Terminales rosca
- Adaptadores BNC

iii) Seleccionar el dispositivo DAQ. Los criterios son tales como exactitud, razón de muestreo, número de canales, flexibilidad, fiabilidad, expandibilidad y comunicación para una PC.

Alto funcionamiento multifunción I/O
- Mas versatilidad
- Exactitud garantizada
- Muestro rápido
- Sincronización de multi tarjetas
- Contadores y timer avanzado
- Protección I/O

Instrumentos basados en PC
- Osciloscópios digitales
- Multímetros digitales
- Generadores de ondas arbitrarias
- Analizadores dinámicos de señales
- Analizador de datos seriales
- Medidores de temperatura / voltaje

Bajo Costo
- Menos de 16 canales
- Funcionamiento básico
- Solución económica

Portable
- Tarjeta PC (PCMCIA)
- USB
- Puerto paralelo
- Multifunción I/O

iv) Seleccionar los cables apropiados. Cables para la conexión de la tarjeta DAQ con los dispositivos de acondicionamiento de señal.

Cables cilíndricos
- Mayor exactitud
- Aislamiento de ruido
- Para instrumentación

Cables planos
- Bajo costo
- Flexible

v) Seleccionar el software. Para implementar un sistema de adquisición de datos se puede realizar por medio de un driver o de un software de instrumentación.

Sofware Driver NI-DAQ
- Tarjeta con librería completa de funciones
- Multiplataforma

Lenguajes y ambientes de desarrollo
- Labview
- LabWindows/CVI
- Visual Basic
- Measure

Sistemas Operativos
- Windows NT/98/95/3.1
- Mac OS
- DOS

1.7.1.3.1 Librería GPIB

Describe las funciones NI-488 y NI-488.2, así como también las funciones de manejo del Dispositivo en LabWindows. Se organizan en los siguientes módulos:

- Abrir / Cerrar. Funciones de panel que abren y cierran los dispositivos y tarjetas GPIB
- Configuración. Funciones de panel que alteran los parámetros de configuración que fueron colocados durante la instalación del GPIB.
- I/O. Funciones de panel que leen o escriben datos sobre el GPIB.
- Control del Dispositivo. Funciones de panel que proporciona un alto nivel, normalmente usado para el control del instrumento GPIB.

- Control del Bus. Funciones de panel que proporciona un control en bajo nivel del bus GPIB.
- Tarjeta de Control. Funciones de panel que proporciona un control en bajo nivel de la tarjeta GPIB.

En el Apéndice B se da una lista detalla de las funciones de librería GPIB.

Al usar las funciones de librería GPIB se debe tener en cuenta lo siguiente:

- Antes de llevar a cabo cualquier operación se debe abrir el dispositivo. Se puede usar cualquiera de las funciones OpenDev, Ibfind o el ibdev. Cuando se abre un dispositivo, un valor entero que representa al dispositivo es retornado.
- Si se usa OpenDev, se debe usar la función CloseDev para cerrar el dispositivo y terminar la ejecución del programa.
- Cada Librería GPIB tiene tres controles etiquetados como Status, Error y Count. Estos controles muestran el valor del status del GPIB (ibsta), error (iberr) y el count (ibcnt).

  - El control Status se despliega en formato hexadecimal. La ayuda para el Status explica el significado para cada bit de la palabra status. Si el bit más significativo esta encendido, ha ocurrido un error GPIB.
  - Cuando a ocurrido un error, el control Error despliega el número del error. La ayuda para el Error describe el tipo de error para cada numero.
  - El control Count despliega el número de bytes transferidos durante la más reciente transferencia de información sobre el bus.

1.7.1.3.2 Librería RS-232

Describe las funciones para la adquisición de datos serialmente. Se organizan en los siguientes módulos:

- Abrir y Cerrar. Funciones de panel que abren, cierran y configuran un puerto com.
- Entrada / Salida. Funciones de panel que leen y escriben a un puerto com.
- Xmodem. Funciones de panel que transfieren archivos usando el protocolo Xmodem.
- Control. Funciones de panel que colocan el límite de tiempo fuera, los modos de comunicación, llenar las colas de I/O y enviar la señal de interrupción.
- Estado. Funciones de panel que retornan el estado del puerto com y la longitud de las colas.

En el Apéndice C se da una lista detalla de las funciones de librería RS-232.

Con las funciones Xmodem se puede transferir archivos usando un protocolo para la transferencia de datos. Para transferir un archivo se debe abrir un puerto com, usar la función XmodemSend y en el otro lado del sistema la función XmodemReceive y finalmente cerrar el puerto com. Las funciones Xmodem manipula todos los aspectos del protocolo de transferencia. También se pueden usar las funciones ComToFile y ComFromFile para tareas mas precisas.

En la Figura 1.9 se da la configuración del cable serial de la PC:

Pin	Significado
2	TxD Transmit Data *
3	RxD Receive Data
4	RTS Request to Send *
5	CTS Clear To Send
6	DSR Data Set Ready
20	DTR Data Terminal Ready *
7	Common

Figura 1.9 Pines de Conexión Serial del Cable PC

El simbolo * indica que son líneas manejadas por la PC y las otras son líneas del monitor de la PC.

Todos los dispositivos serial es del tipo Data Communication Equipment (DCE) o del tipo Data Transmission Equipment (DTE). La PC es del tipo DTE. La diferencia entre los dos tipos de dispositivos es el orden de asignación de sus pines.

El cable requerido por la PC (o DTE) para comunicarse con un dispositivo DCE tiene la siguiente configuración (Ver Figura 1.10):

(PC)	Conexión de pines	(Dispositivo)
TxD*	2 con 2	RxD
RxD	3 con 3	TxD*
RTS*	4 con 4	CTS
CTS	5 con 5	RTS*
DSR	6 con 6	DTR
DTR*	20 con 20	DSR*
common	7 con 7	common

Figura 1.10 Cable DTE a DCE

El cable requerido por la PC (o DTE) para comunicarse con un dispositivo DTE tiene la siguiente configuración (Ver Figura 1.11):

(PC)	Conexión de pines	(Dispositivo)
TxD*	2 con 3	RxD
RxD	3 con 2	TxD*
RTS*	4 con 5	CTS
CTS	5 con 4	RTS*
DSR	6 con 20	DTR
DTR*	20 con 6	DSR*
common	7 con 7	common

Figura 1.11 Cable DTE a DTE

1.7.1.3.3 Librería DDE

Describe las funciones para el intercambio dinámico de datos. Estas librerías solamente soportan DDE's de Microsoft Windows. Cada función DDE genera una o más llamadas de funciones DDE.

Un servidor DDE puede ejecutar comandos enviados desde otra aplicación y enviar datos a una aplicación Cliente. Un Cliente DDE puede enviar comandos para ser ejecutado o solicitar datos desde un Servidor.

Con LabWindows/CVI se puede escribir programas que actúen como un cliente ó servidor DDE. Para conectar un servidor DDE desde un programa de LabWindows/CVI, se debe conocer alguna información acerca de la aplicación a la cual se va a conectar. Todo servidor DDE tiene un nombre y un tópico que define la conexión. Por ejemplo si se desea conectar Microsoft Excel como un servidor se puede realizar de dos formas por medio de la función ConnectToDDEServer. Si se desea enviar comandos para ser ejecutados por Excel, tales como abrir hojas de cálculo o crear gráficas, se debe especificar a Excel como servidor y System como el tópico en la función ConnectToDDEServer. Pero si se desea enviar datos a una hoja de Excel, se debe especificar como el servidor y el nombre del archivo con los datos como el tópico.

Si el programa actúa como un servidor DDE, donde otras aplicaciones bajo Windows envía y recibe comandos o datos, es necesario llamar la función RegisterDDEServer dentro del programa. La función RegisterDDEServer establece que el programa sea un servidor DDE tal que otras aplicaciones pueden conectarse para intercambiar información.

En el Apéndice D se da una lista detalla de las funciones de librería DDE.

1.7.1.3.4 Librería TCP

Describe las funciones para el protocolo de control de transmisión de datos y la red de comunicaciones usando las funciones de la librería TCP.TCP las cuales proveen una interface de plataforma independiente para hacerla:

- Fiable
- Orientada a la conexión
- Flujo de bytes
- Protocolo de comunicación con una red de trabajo.

La comunicación en una red de trabajo usando la librería TCP incluye un cliente y un servidor en cada conexión. Un servidor TCP pueden enviar y recibir información desde un cliente a través de una red de trabajo. Un cliente TCP puede enviar y solicitar datos desde un servidor.

Con LabWindows/CVI se puede escribir programas que actúan como un cliente o servidor TCP. Bajo Windows se puede correr un servidor y un cliente sobre la misma PC. El procedimiento para escribir un programa usando TCP es similar al usado con DDE

Para conectar un programa en LabWindows/CVI a un servidor se debe tener alguna información sobre la aplicación a la cual se va a conectar. Todo servidor TCP debe correr sobre un host especificado, el cual tiene un nombre (por ejemplo aaa.bbb.ccc) o una dirección conocida IP (por ejemplo 123.456.78.90). En adición cada servidor especifica su propio y único número de puerto. Estas dos partes de información (nombre de host y número de puerto del servidor) identifican al servidor sobre una misma o diferentes máquinas.

Si el programa en LabWindows/CVI actúa como servidor se debe llamar dentro del programa a la función RegisterTCPServer. La función RegisterTCPServer hace que el programa actúe como un servidor. Los clientes se pueden conectar al programa usando el nombre del host o la dirección IP y el número del puerto asociado a la PC.

En el Apéndice F se da una lista detalla de las funciones de libreria TCP.

1.7.1.4 Análisis de Datos

Son librerias que sirven para analizar la Adquisición de los Datos. A continuación se describen las diferentes clases de librerias que la componen:

- Generación de Señales. Panel de funciones que inicializa los arreglos con patrones predefinidos.
- Operaciones con Arreglos. Panel de funciones que llevan a cabo las operaciones aritméticas con arreglos 1D y 2D
- Operaciones con Complejos. Panel de funciones que llevan a cabo las operaciones aritméticas con números complejos. Este panel puede realizar operaciones de arreglos 1D complejos. La parte real e imaginaria de los números complejos es procesado separadamente.
- Procesamiento de Señales. Panel de funciones que llevan a cabo el análisis de los datos en el dominio de la frecuencia, dominio del tiempo y realiza el filtrado digital.
- Medición. Panel de funciones que llevan a cabo el análisis espectral usando unidades reales tales como hertz y segundos.

- Estadística. Panel de funciones que llevan a cabo las funciones básicas de estadística.
- Ajuste de Curvas. Panel de funciones para el ajuste las curvas usando al *menos cuatro* técnicas. Esta disponibles el ajuste lineal, exponencial, polinomial y no lineal.
- Interpolación. Panel de funciones que toman un conjunto de puntos de una función y determina el valor de un punto intermedio.
- Algebra de Vectores y Matrices. Panel de funciones que llevan a cabo las operaciones de vectores y matrices 1D y 2D.
- Utilidades de Arreglos. Panel de funciones que copia, inicializa y borra los arreglos.

En el Apéndice F se da una lista detalla de las funciones de Librería de Análisis de Datos.

MOTORES DE PASO

2.1 Definición

Es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos discretos llamados pasos.

Este tipo de motor puede ser fácilmente controlado por circuitos electrónicos los cuales convierten pulsos eléctricos en movimientos mecánicos proporcionales. Cada fracción de revolución del motor es hecha por una secuencia de pasos discretos idénticos. El diseño del motor de pasos usualmente provee rotación en el sentido de las manecillas del reloj así como en contra, lo que hace al motor de pasos idealmente conveniente para aplicaciones de posición y de control. El hecho de que los motores puedan ser fácilmente controlados por pulsos eléctricos permite el uso de señales digitales para controlar el movimiento mecánico o la posición. Además el torque relativamente grande asociado con cada paso permite a un motor de pasos reemplazar aparatos como embragues y frenos haciendo más seguros los sistemas.

2.2. Ventajas y Desventajas

Ventajas

- Control por medio de secuencias de señales digitales TTL.
- El ángulo de rotación del motor es proporcional al número de pulsos y la señal de dirección.
- Precisión y exactitud de posicionamiento.
- Excelente respuesta para el arranque, frenado y reversa del movimiento.
- Se pueden trabajar en lazo abierto, proporcionando un fácil control y menor costo.
- Es posible alcanzar muy bajas velocidades de rotación de una carga.
- Amplio rango de velocidades de rotación. Su velocidad es proporcional a la frecuencia de los pulsos.

Desventajas

- Torque variable.
- Resonancias
- No se puede operar a altas velocidades.

2.3 Tipos

2.3.1 Magneto Permanente (MP)

El estator es una estructura dentada donde se distribuye los devanados de las fases. El rotor es un imán permanente que contribuye con flujo magnético. Cuando se excita una fase individual el rotor tiende a alinearse con ella.

En la siguiente Figura 2.1 se muestra un motor MP de 4 fases compuesta de 2 devanados cada una. El mecanismo de funcionamiento se basa en excitar una fase a la vez. Para girar completamente el motor es necesario alimentar con pulsos de corriente positivos y negativos (Ver Figura 2.2).

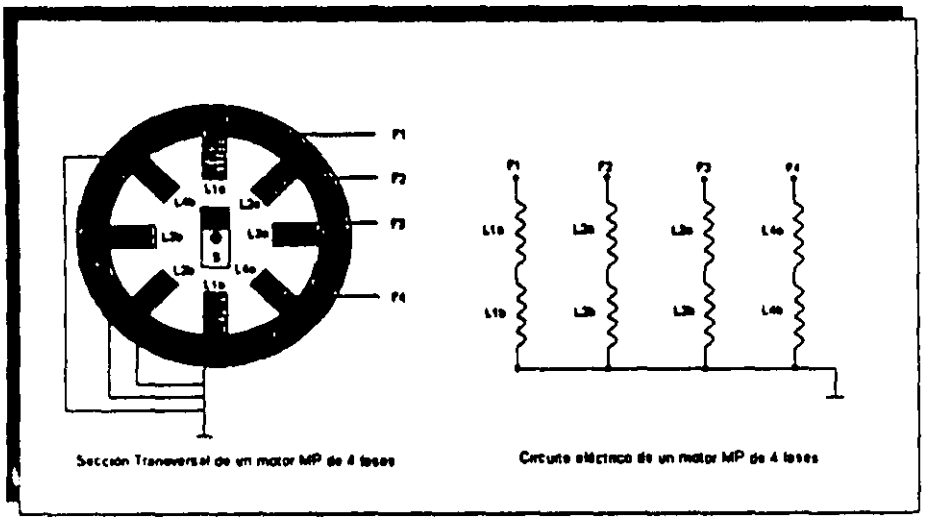


Figura 2.1 Esquema y Modelo de MP de 4 fases

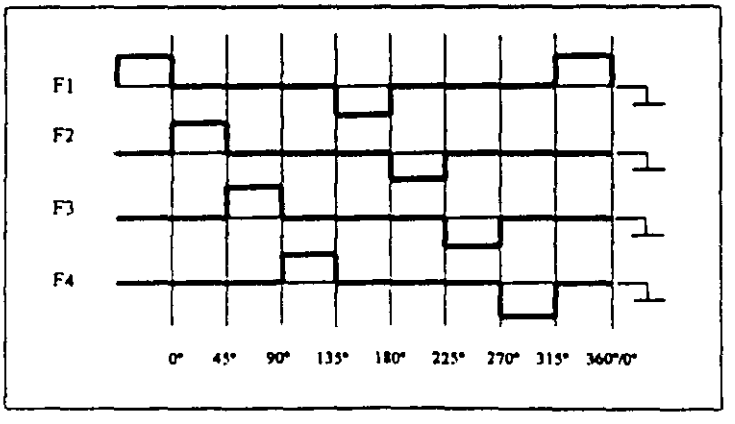


Figura 2.2 Tren de pulsos para un motor MP de 4 fases

Si *Ns* es el número de devanados del estator, por lo tanto el ángulo de paso Φs es:

$$\Phi_S = \left(\frac{360}{N_S}\right)$$

2.3.2 Reluctancia Variable (RV)

Tanto el rotor como el estator cuentan con determinado número de dientes. Los devanados de las fases se distribuyen en el estator. El rotor esta constituido de material ferromagnético (no es un magneto permanente).

Los motores de paso RV se subdividen a su vez en dos tipos. Una de estas configuraciones de motores de paso RV consta de un rotor y un estator único con fases múltiples, a este se le llama motor de paso RV y conjunto único. La otra configuración de motor de paso RV está constituida por un conjunto de motores monofásicos de RV desplazados axialmente y montados en el mismo eje, a este se le llama motor de paso RV y varios conjuntos. La resolución angular de un motor de RV se puede determinar por el número de dientes del estator y se puede aumentar mucho mediante técnicas como el dentado.

En la Figura 2.3 se presenta un motor de pasos RV y conjunto único de tres fases. Las secuencias de control: se muestra en la Figura 2.4:

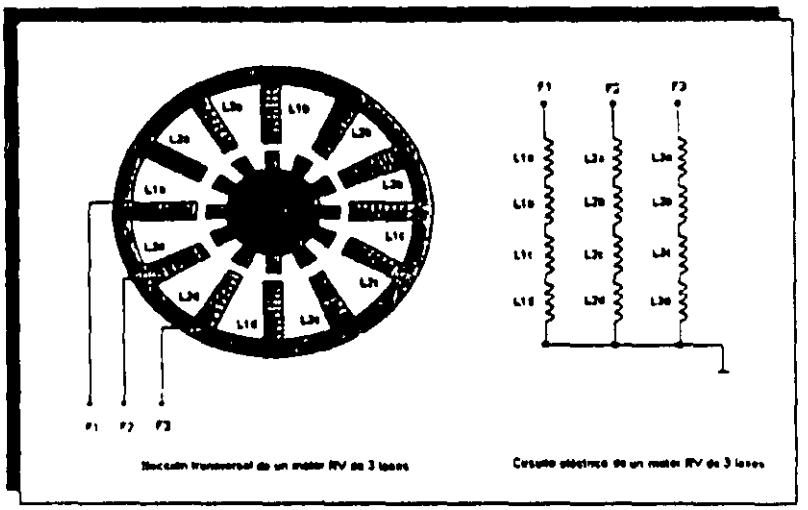


Figura 2.3 Esquema y modelo de un motor RV de 3 fases

Sea *Ps* el angulo entre dos dientes del estator, *PR* el ángulo de inclinación entre dos dientes del rotor, *Ns* el numero de dientes del estator y *NR el número de dientes en el rotor:*

$$P_S = \left(\frac{360}{N_S}\right) \qquad P_R = \left(\frac{360}{N_R}\right)$$

El número de fases se designa por *NP*, el ángulo de paso se calcula por:

$$\Phi s = \left(\frac{P_R}{N_P}\right)$$

El stepping rate *RS* es el total de pasos para completar una vuelta:

$$R_S = \left(\frac{360}{\Phi_S}\right) \qquad R_S = N_P \times N_R$$

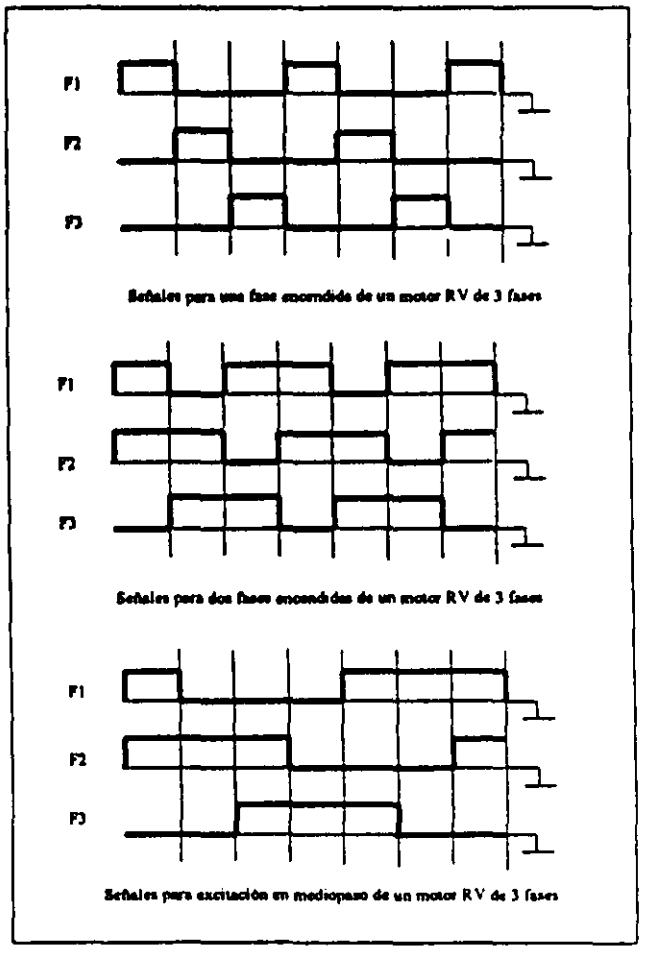


Figura 2.4 Secuencias de control para un motor RV de 3 fases

2.3.3 Hibridos (HB)

Son los motores más utilizados en aplicaciones industriales. Como su nombre lo dice, estos motores combinan características de operación de los motores de MP y RV de varios conjuntos (Ver Figura 2.5).

Consisten de un ROTOR de dos polos con un cierto número de dientes. Entre los polos existe un magneto concéntrico que esta magnetizado a lo largo del eje del rotor (formando entre sus extremos los polos Norte y Sur). El ESTATOR es un caparazón fijo de cierto número de dientes, estos dientes constituyen los devanados y están conectados en pares.

El magneto permanente sirve para mantener el rotor en reluctancia *mínima* cuando no existe secuencia de pasos. Es decir, provoca un torque pequeño que fija el rotor en una posición determinada. Este torque se le llama " detem torque". La reluctancia mínima se da cuando un par de polos del rotor se alinean con dos polos del estator.

Cuando existe *nuevamente* secuencia de pasos, entonces se requiere un torque mayor para mover el rotor a otra posición determinada. Este torque es llamado "holding torque".

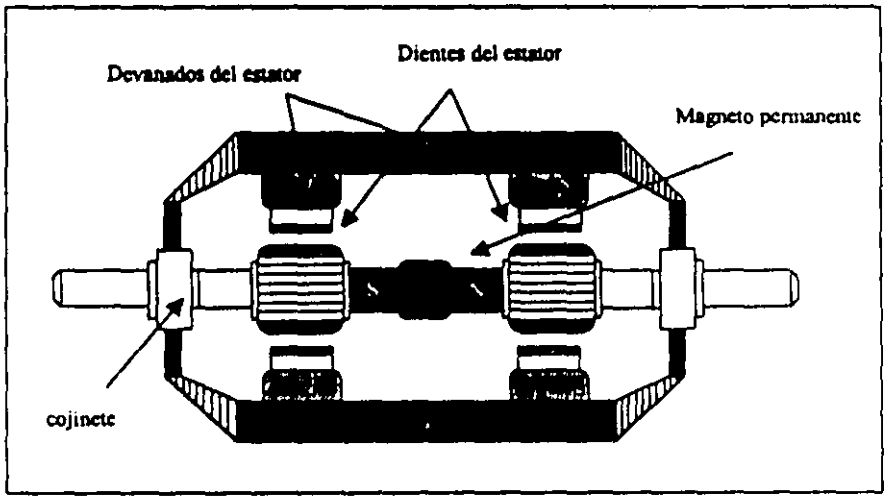


Figura 2.5 Motor de pasos híbrido

Los motores de paso tienen gran importancia debido a que pueden ser fácilmente controlados por señales digitales, lo que permite construir sistemas muy simples de control de movimiento y posición los cuales poseen gran exactitud. Una de las aplicaciones donde podemos observar el grado de exactitud de los motores de paso es la del manejo de las cabezas de los discos duros y flexibles de almacenamiento de datos. También los motores de paso encontramos presente en el mecanismo que controla el avance del papel y la posición de la cabeza de una impresora. En la industria la mayoría de los equipos de maniobra X-Y estan construidos con motores de paso. Una de las aplicaciones más interesantes es su uso en robótica. En este campo los motores de paso son utilizados en el control de posición de cada una de las junturas de los robots logrando con esto una amplia gama de movimientos de gran exactitud.

2.4 Modos de Secuencia

2.4.1 Paso completo (una fase encendida)

Se lleva a cabo cuando se enciende una fase durante un determinado tiempo. Las posiciones alcanzadas se muestran en la Figura 2.6.

Esta secuencia produce un movimiento liso y consume menos potencia. El torque generado tiene una corriente del 41% menos que la generada cuando se encienden dos fases a la vez. Por lo tanto el torque generado es menor.

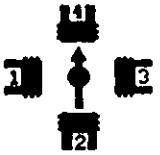
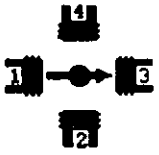
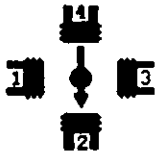
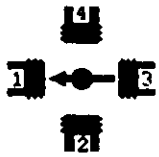
PASO	Bobina4	Bobina3	Bobina2	Bobina1	
1	1	0	0	0	
2	0	1	0	0	
3	0	0	1	0	
4	0	0	0	1	

Figura 2.6 Secuencia paso completo (una fase encendida)

2.4.2 Paso completo (Dos fases encendidas)

Se lleva a cabo cuando se encienden dos fases a la vez. Las posiciones que pueden alcanzar se muestran en la Figura 2.7.

Esta secuencia produce un movimiento no muy liso, usa más potencia pero produce un gran torque.

- La bobina 4 es siempre inversa a la bobina 2.
- La bobina 1 es siempre inversa a la bobina 3.

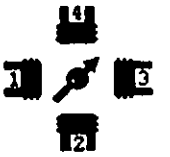
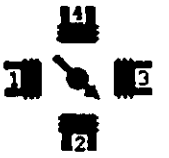
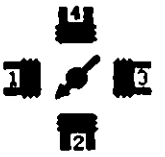
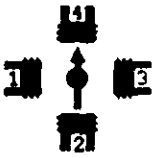
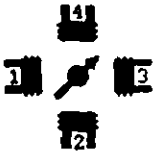
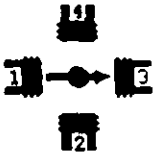
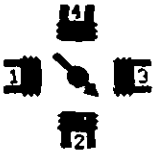
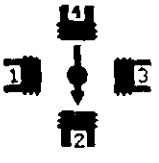
PASO	Bobina4	Bobina3	Bobina2	Bobina1	
1	1	1	0	0	
2	0	1	1	0	
3	0	0	1	1	
4	1	0	0	1	

Figura 2.7 Secuencia paso completo (dos fases encendidas)

2.4.3 Mediopaso

Se logra intercalando los tipos de secuencias de paso completo de una y dos fases encendidas. Las posiciones alcanzadas se muestran en la Figura 2.8.

La ventaja de esta secuencia es que la resolución se duplica, logrando tener el doble de pasos para realizar una vuelta completa del motor. Entre la desventaja es que se produce un torque variable, por los diferentes niveles de corriente generados al encender una y dos fases secuencialmente. Este problema se soluciona trabajando en el modo mediopaso modificado cuyo objetivo principal es el cambio inteligente del nivel de referencia de voltaje.

Paso	Bobina4	Bobina3	Bobina2	Bobina1	
1	1	0	0	0	
2	1	1	0	0	
3	0	1	0	0	
4	0	1	1	0	
5	0	0	1	0	

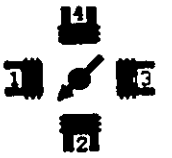
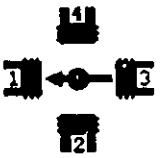
6	0	0	1	1	
7	0	0	0	1	
8	1	0	0	1	

Figura 2.8 Secuencia medio paso

2.4.4 Micropasos

Es una forma de movimiento de los motores de paso donde se pueden alcanzar posiciones entre las posiciones alcanzadas en el modo paso completo. Su principio de funcionamiento se basa en la variación continua de la corriente en los devanados del motor.

2.5 Circuitos para la generación de secuencias

Para la generación de las secuencias de paso de un motor de pasos se pueden realizar por circuitos TTL o por circuitos integrados especiales llamados circuitos drivers para motores de paso. Las señales comunes de control son:

- Step. Señal de reloj que determina la velocidad angular de giro del motor.
- Direccion. Señal que determina la dirección de giro del motor.
- Step mode. Señal que determina el tipo de secuencia usada para el giro del motor.

2.5.1 Generadores de secuencia paso completo

En la Figura 2.9 se muestra un circuito de un generador de secuencia de paso completo en una sola dirección y en la Figura 2.10 se muestra un circuito de un generador de paso completo en ambas direcciones.

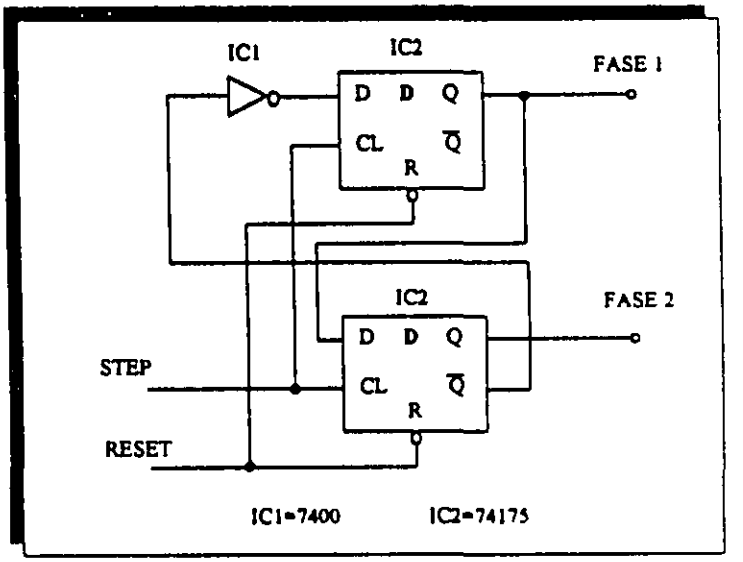


Figura 2.9 Generador paso completo en una dirección

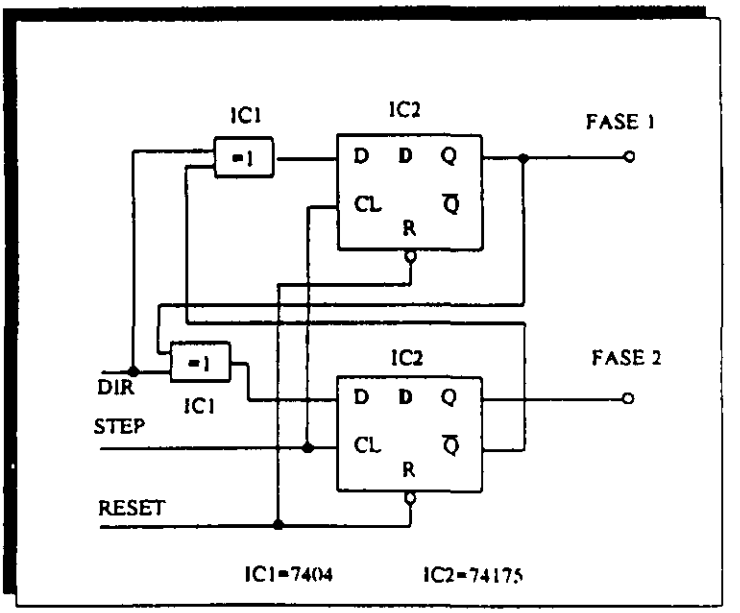


Figura 2.10 Generador paso completo en dos direcciones

2.5.2 Generador de secuencia de paso completo, mediopaso y mediopaso modificado en ambas direcciones

Un Generador de mediopaso en ambas direcciones barato y fácil de implementar con lógica TTL se muestra en la figura 2.11.

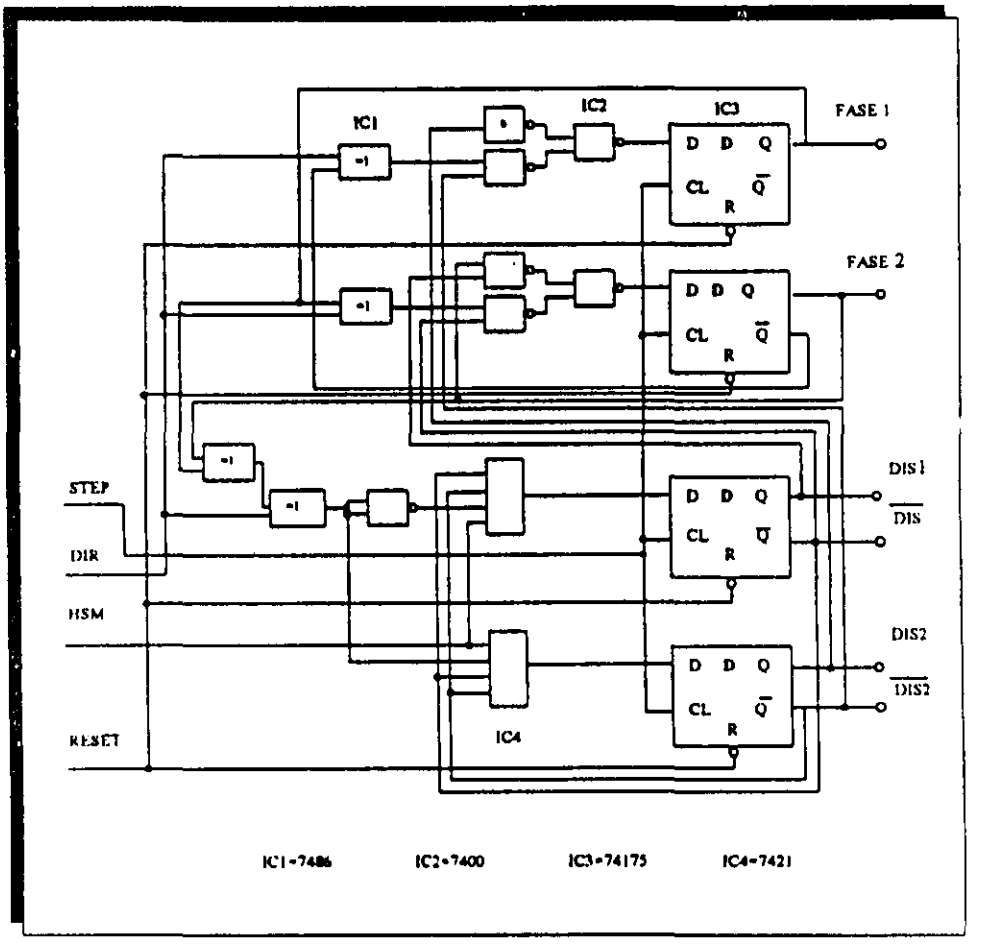


Figura 2.11 Generador paso completo, mediopaso, mediopaso modificado en ambas direcciones

PBD 3517 unipolar stepper motor driver:

El Driver mostrado en la Figura 2.12 es un circuito integrado producido por Ericson para manejar un motor de pasos híbrido unipolar.

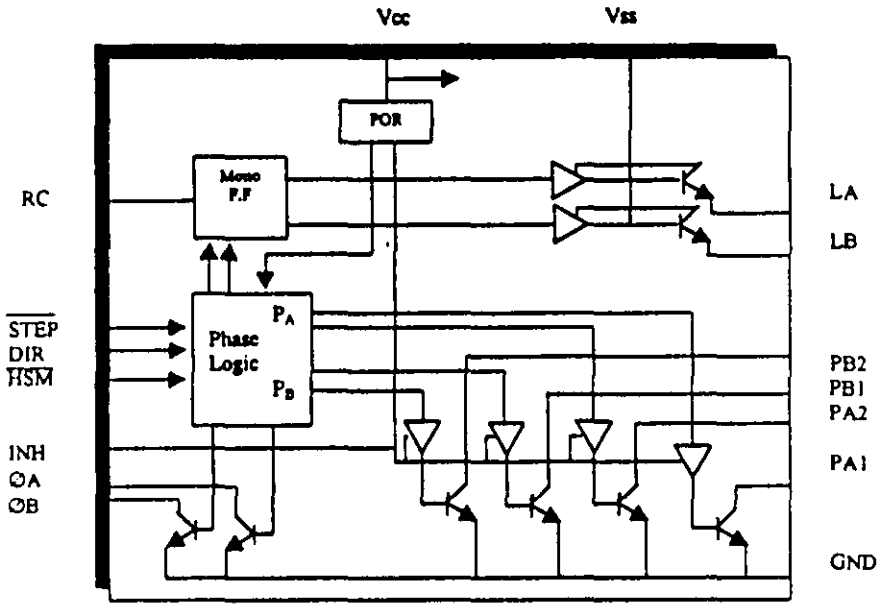


Figura 2.12 Diagrama de bloque del PBD 3517

2.5.3 Circuitos para el cambio de nivel en la secuencia mediopaso modificado

Los circuitos mostrados en la Figura 2.13 son cambiadores de nivel útiles cuando se trabajan los motores de paso en el modo mediopaso modificado. Es decir con cualquiera de ellos podremos graduar la corriente en los devanados del motor cuando se active la secuencia paso completo con dos fases encendidas y evitar así variaciones del torque durante su funcionamiento.

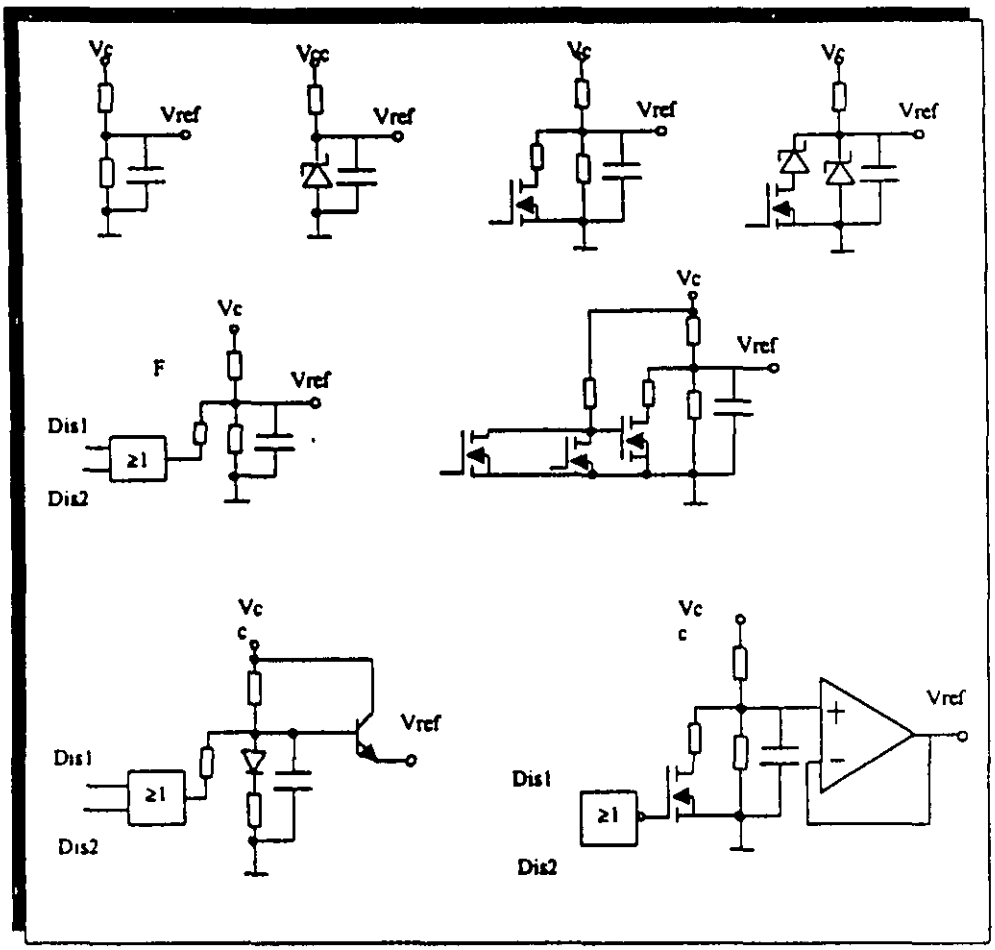


Figura 2.13 Ejemplos de circuitos de cambio de nivel

A. Un ordinario circuito divisor de voltaje por resistencias.
B Se usa un diodo zener para colocar el nuevo voltaje de referencia. El diodo zener reduce la influencia de la variación de la resistencia interna.
C Este circuito usa un MOS FET para switchear una resistencia en paralelo con el divisor de voltaje. Un capacitor en la salida mejora el rechazo al ruido.
D Usando diodos zener en lugar de resistencias reducen la influencia de la resistencia interna.
E Circuito que sirve para ser usado junto con el PBD 3517.
F Este circuito usa transistores FET para formar una función OR.
G Es el mejor circuito usado con el PBD 3517, en el cual el buffer es un transistor usado en el modo emisor abierto.
H Circuito donde el buffer es un amplificador operacional.

2.6 Drivers para motores de paso

2.6.1 Definición

El driver de un motor de pasos entrega potencia eléctrica como respuesta a la entrada de señales digitales desde un sistema de control (ver Figura 2.14).

El driver debe actuar como una fuente de corriente. La corriente proporcionada al motor de pasos recorre sus fases generando su torque. La fuerza magnetomotriz del estator es igual al producto de la corriente por el número de vueltas del devanado (Amp-Vueltas).

Las señales de entrada a un driver de motores de paso son los pulsos de paso y una señal de dirección de giro. El motor gira un paso por cada pulso de entrada. Esto es indiferente del modo de trabajo. Un driver puede requerir de 200 a 101,600 pulsos para producir una revolución de la flecha. Para una velocidad de flecha de 1800 rpm, se requiere una frecuencia de pulsos de 20KHz. 25,000 pasos por revolución requieren una frecuencia de paso de 750KHz, por lo tanto los drivers de micropasos deben de manejar altas frecuencias de paso.

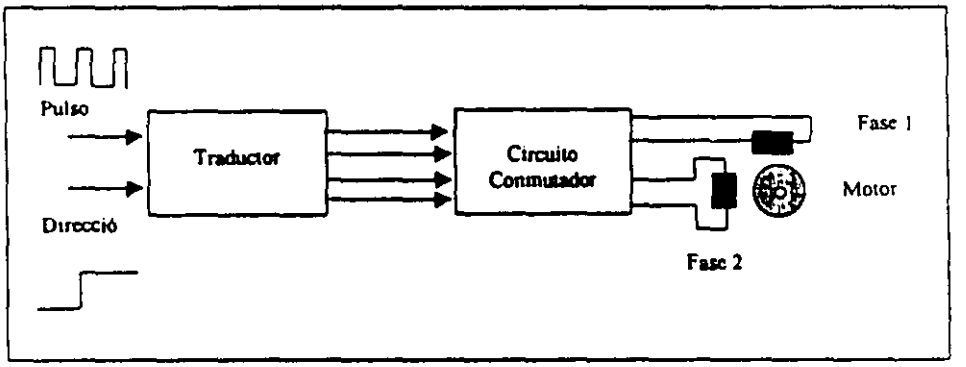


Figura 2.14 Elementos de un driver para motores de paso

La función de la sección lógica del driver (traductor), es trasladar las señales de control (pulso y dirección) en un conjunto de señales que llevan a cabo la secuencia de paso del circuito conmutador.

2.6.2 Tipos de drivers

2.6.2.1 Driver Unipolar

Son utilizados para los motores unipolares con tap central, es decir aquellos que tienen 5 ó 6 hilos y su característica es que la corriente sólo circula en una dirección por cada medio devanado. La corriente se puede graduar colocando una resistencia en serie por cada fase.

El arreglo de transistores de la Figura 2.15 constituye el elemento conmutador de un driver unipolar. Los transistores son controlados de acuerdo a la secuencia de paso utilizada por medio de otros circuitos que aceptan las señales de pulso y dirección o por la generación de estas secuencias por medio de software

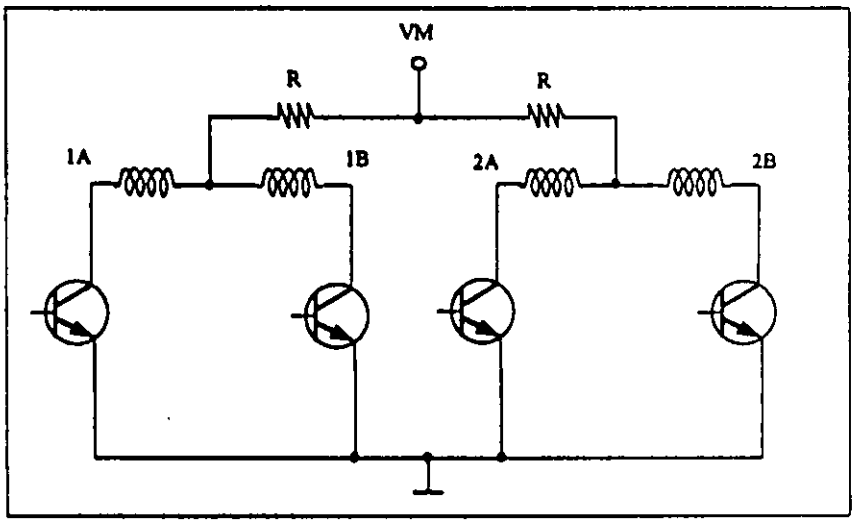


Figura 2.15 Driver Unipolar

La corriente sólo circula medio devanado de la fase por cada paso. Para la secuencia de paso completo sólo se activa una fase y para secuencias de mediopaso se activan alternativamente una o dos fases a la vez. De ello se nota que para este tipo de driver no se pueden utilizar todos los devanados al mismo tiempo.

Este driver nos da más precisión pero nos genera una variación del torque, ya que cuando se activan dos fases a la vez, la corriente se incrementa en un 40% aproximadamente.

2.6.2.2 Driver Bipolar

Son utilizados en motores de pasos bipolares híbridos y su característica es que la corriente puede circular en ambas direcciones por los devanados.

2.6.2.2.1 Driver Bipolar Simple

Como se muestra en la Figura 2.16 utilizada dos transistores por cada fase. La primera desventaja es que utiliza dos fuentes y una de ellas no hace da cuando la otra esta activa. La segunda desventaja es que los transistores deben tener la capacidad de ser conmutados al doble del voltaje que se aplica al motor

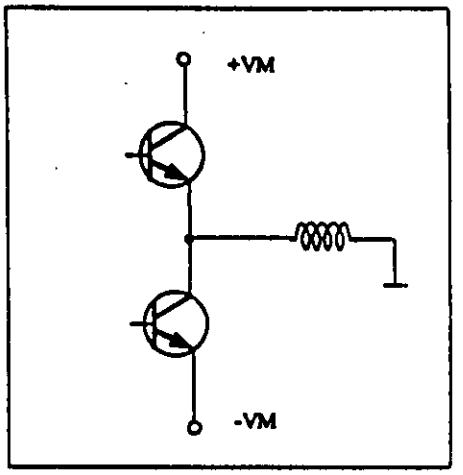


Figura 2.16 Driver Bipolar Simple

2.6.2.2.2 Driver Bipolar Bridge

Utiliza cuatro transistores por cada fase, la ventaja es que sólo es necesario una sólo fuente. Este arreglo se usa para aplicaciones de alto torque (Ver Figura 2.17).

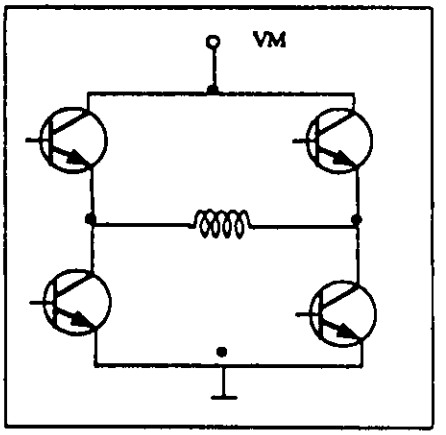


Figura 2.17 Driver Bipolar Bridge

2.6.2.2.3 Driver Bipolar R-L

Es una modificación del driver bipolar bridge al incluir una resistencia en serie con el devanado. Esta resistencia tiene el objetivo de graduar la corriente y son usadas en aplicaciones de bajo torque (Ver Figura 2.18)

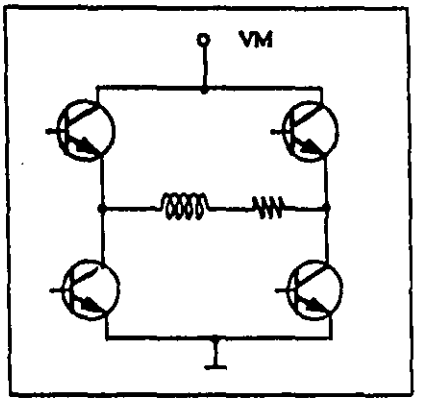


Figura 2.18 Driver Bipolar R-L

2.6.2.2.4 Driver de Recirculación Chopper

Es el método más usado de control de la corriente. Como se muestra en la Figura 2.19 utiliza cuatro transistores en arreglo bridge, diodos de recirculación y una resistencia para sensado. La resistencia es de bajo valor (típicamente 0.1 Ohm) y provee una retroalimentación de voltaje proporcional a la corriente en el motor.

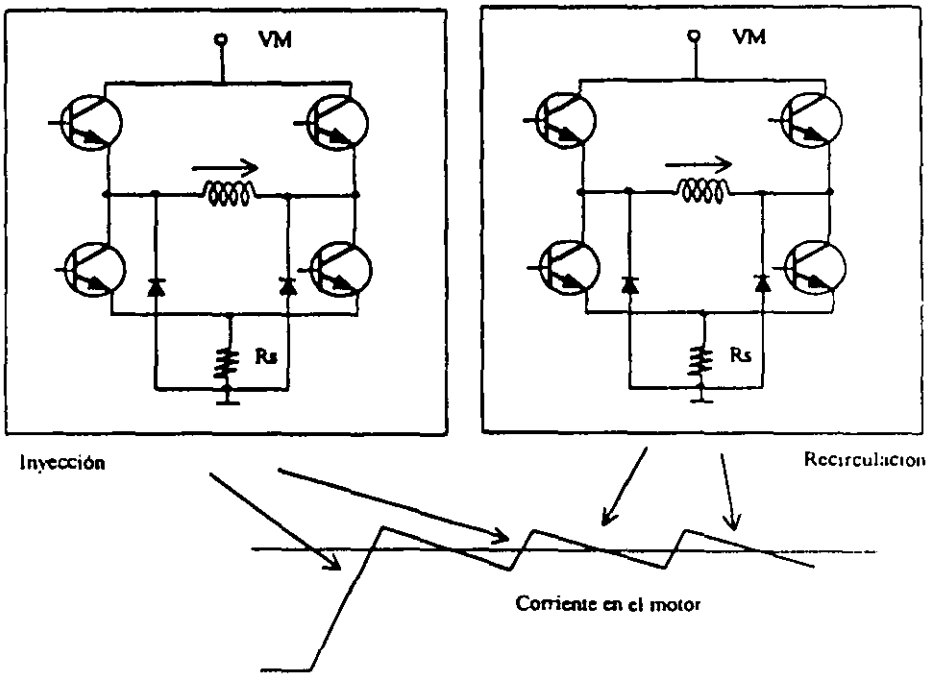


Figura 2.19 Driver de Recirculación Chopper

La corriente es inyectada al devanado del motor al activar un transistor superior y otro inferior. La corriente se incrementa linealmente y se puede monitorear a través de la resistencia Rs. Cuando el nivel de corriente a sido alcanzado, el transistor superior se apaga y la corriente queda atrapada circulando entre el transistor inferior y el diodo volante. La corriente decae y cuando alcanza un nivel de umbral, se enciende nuevamente el transistor superior y el ciclo se repite. La corriente es mantenida por lo tanto a un valor promedio conmutando o "chopping" la fuente.

Una variación de este circuito es el driver chopper regenerativo (Ver Figura 2.20). En este driver la fuente es aplicada en los devanados en direcciones alternas, causando rampas de corriente ascendentes y descendentes con igual pendiente. Esta técnica requiere pocos componentes y es de bajo costo, sin embargo la corriente de rizo en el motor es usualmente grande causando un incremento de calor en el motor.
El motor de pasos al girar actúa como un generador. La energía dada a la carga durante la aceleración es retornada al driver en la desaceleración. Esto incrementa la corriente en el motor y puede dañar los transistores de potencia. Un detector en el driver sensa el incremento en la corriente y apaga momentáneamente todos los transistores.

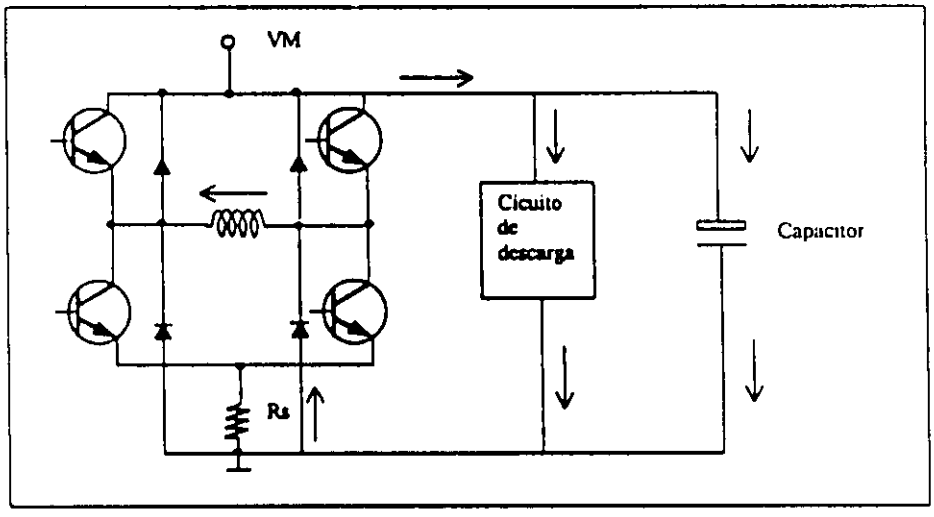


Figura 2.20 Regeneración durante el flujo de corriente

Hay ahora una ruta para la corriente regenerada hacia el capacitor, lo que incrementa el voltaje suministrado. Durante esta fase no esta fluyendouna gran corriente a través de la resistencia de sensado, los transistores deben ser activados nuevamente después de un corto periodo (típicamente 30uS). Si la corriente es aún alta, el driver retorna al estado regenerativo.

Un pequeño incremento en la fuente es aceptable, pero si es demasiado grande los transistores pueden ser dañados por sobre voltaje en lugar de altas corrientes. Para solucionar este problema se usa un circuito de descarga para disipar la potencia regenerada.

Un diodo rectificador y un capacitor alimentados con AC desde un transformador, provee un voltaje de referencia igual al valor pico de la señal AC. Bajo condiciones normales este debe ser el mismo que el voltaje de la fuente. Durante la regeneración, el voltaje del driver llega a estar arriba de esta referencia y enciende el transistor de descarga conectando la resistencia de 33Ω con la fuente (Ver Figura 2.21).

Cuando el voltaje de la fuente a decrecido lo suficiente, el transistor es nuevamente apagado. La corriente disipada puede ser alta, pero la potencia disipada es usualmente mediana puesto que el periodo de descarga es muy corto. En aplicaciones donde la potencia regenerada es alta, causado frecuentemente por una rápida desaceleración de la inercia de la carga, puede ser necesario otra resistencia de descarga.

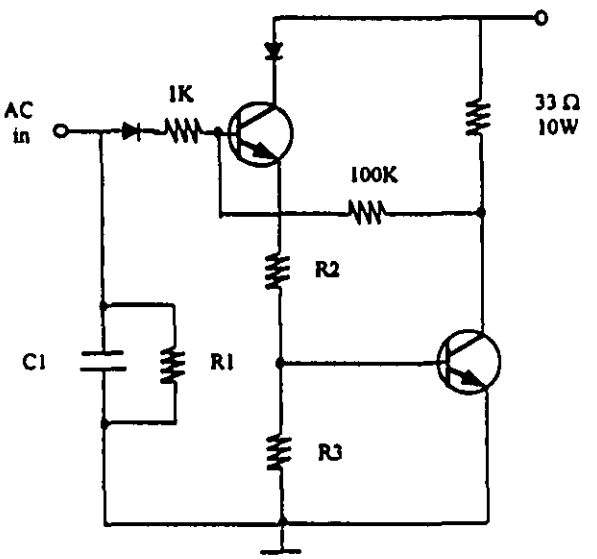


Figura 2.21 Circuito de Descarga

- Driver Unipolar: Dispositivo electrónico de potencia que entrega corriente a los devanados de los motores de paso del posicionador bidimensional.
- Posicionador Bidimensional: Dispositivo mecánico que realiza el movimiento real.

3.2 Diseño e Implementación de componentes de hardware

3.2.1 Tarjeta I/O de Propósito General Programable

En cualquier PC la capacidad de entrada/salida es limitada, para demostrarlo basta unos simples ejemplos: salida serie, salida impresora, mouse, joystick y no posee otras alternativas. Para realizar el montaje del posicionador bidimensional se hizo necesario tener algo más que estas salidas/entradas. Por lo tanto se pensó desarrollar una tarjeta de interface al bus de expansión de la PC para poder ocupar los puertos externos libres y las señales de alimentación.

La PC utilizada en el presente trabajo es una 486 DX, por lo cual se diseño una Tarjeta de Propósito General I/O Programable para ser conectada al Bus ISA XT. Esta Tarjeta tiene capacidad para manejar 6 puertos de 8 bits con opción a ser ampliada hasta 24 puertos de 8 bits. Mas adelante se presenta el circuito electrónico de la Tarjeta I/O y una breve explicación de su diseño.

3.2.1.1 Conector Bus ISA XT

Ranura de expansión para comunicación con sistemas de 8 bits de información. En la Figura 3.2 se muestra el esquema con sus terminales.

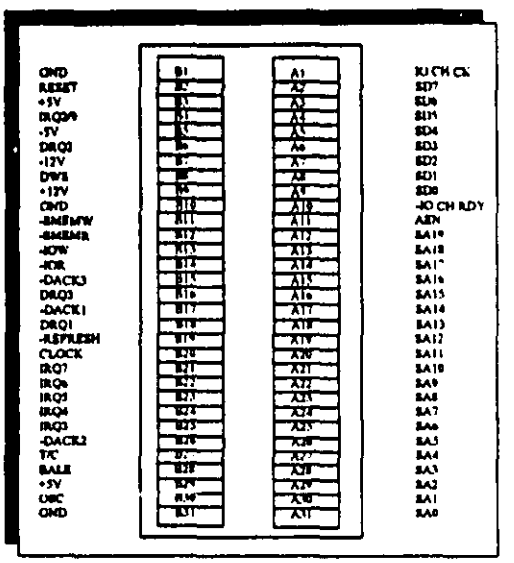


Figura 3.2 Terminales Bus ISA XT

Dentro de una PC por lo general se encuentran de 5 a 8 conectores bus, pudiéndose insertar tarjetas en cualquiera de estas.

3.2.1.2 Expansión del Bus ISA XT para AT

Ranura de expansión del Bus ISA XT. En el se encuentra básicamente los 8 bits restantes para lograr el bus AT de 16 bits (Ver Figura 3.3).

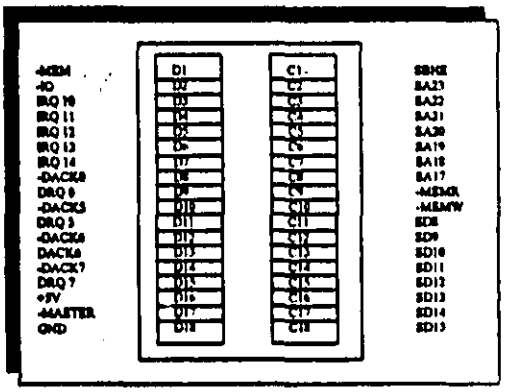


Figura 3.3 Terminales Bus ISA AT

3.2.1.3 Descripción de los Terminales del Bus XT

SD0-SD7: Líneas de Datos Bidireccionales usadas para transmitir datos entre el microprocesador, memoria, puertos. La dirección de los datos depende de las señales del microprocesador -IOR (leer) y -IOW (escribir).

OSC: Es una salida de frecuencia usada internamente para el reloj de tiempo real, interfaces RS232. No tiene relación con la señal de CLOCK.

CLOCK: Es una salida de la frecuencia que usa el microprocesador.

RESET: Es una salida provista por el mecanismo interno de reset para reiniciar las tarjetas conectadas al microprocesador.

BALE: Es una salida del controlador de bus (8288) usada para indicar cuando las señales A0-A19 son correctas. Esta señal se usa para sincronizar las direcciones con los datos.

I/O CH CK: Es una entrada para reportar al microprocesador una condición de error sobre el bus que atañe a la interface conectada al bus. Produce una interrupción del microprocesador del más alto nivel (NMI)

-I/O CH RDY: Es una salida usada para extender el ciclo de espera del microprocesador sobre una tarjeta conectada al bus (wait State 1,2,3....).

IRQ2-IRQ7: Entradas de interrupción al microprocesador. Como el microprocesador tiene una sola entrada de interrupción se le adiciona un controlador que posee 8 entradas, de las cuales 2 ya están usadas en la tarjeta madre (IRQ0 e IRQ1). Se usan para informar al microprocesador que requerimos su atención para pedirle o mandarle datos ejecutando un subprograma específico a cada IRQ.

-IOR: Salida sincronizada con los datos SD0-SD7 para su ingreso al microprocesador. Sólo para puertos
-IOW: Salida sincronizada con los datos SD0-SD7 para su egreso del microprocesador. Sólo para puertos
-SMEMW - SMEMR: Idénticas a las anteriores, pero para direccionar memoria.

DRQ1-DRQ3: Entradas para pedir un ciclo de DMA (Acceso directo a memoria), el método más rápido de acceso a memoria.

DACK0-DACK3: Salida del 8253 (Controlador DMA) para el reconocimiento de un ciclo DMA.

AEN: Señal de salida que indica en nivel bajo, la dirección válida de acceso a memoria o puerto a través del bus. En nivel alto indica que se está realizando un ciclo DMA.

T/C: Salida que indica la terminación de un ciclo DMA, ya sea de un bloque o de un caracter.

3.2.1.4 Direcciones de Puertos Externos

En general en las PC's el direccionamiento esta limitado a 512 puertos y muchos de los cuales ya son usados por los periféricos. En la Figura 3.4 se muestra un mapa de direcciones que nos permite conocer la dirección de los puertos externos libres que podemos ocupar sin interferir en la desconexión de cualquier periférico.

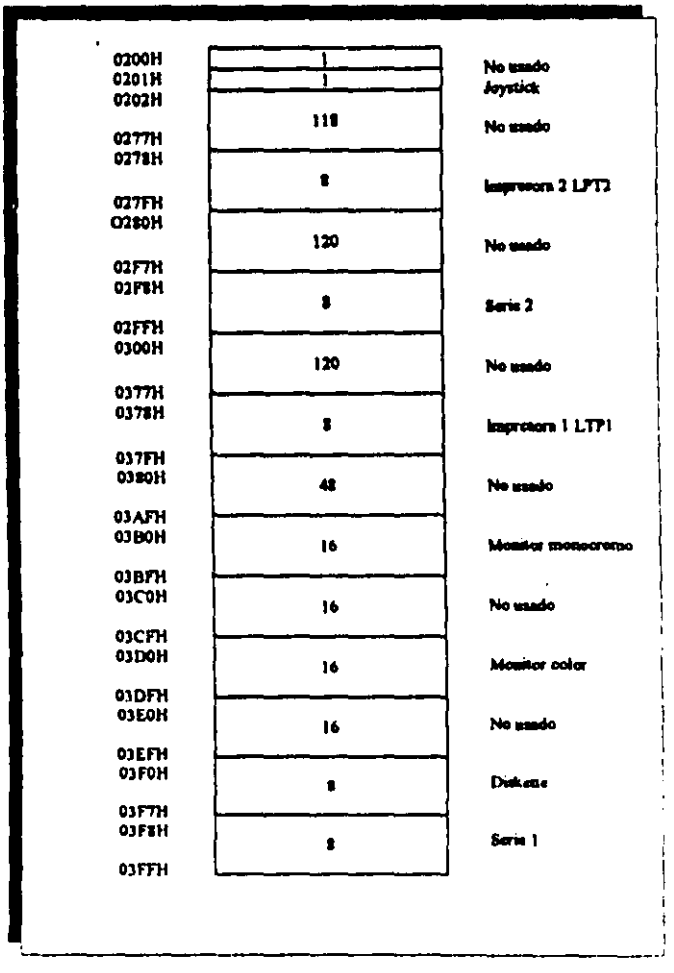


Figura 3.4 Direcciones de puertos

3.2.1.5 Diagrama de conexiones de la Tarjeta I/O (Ver Figura 3.5):

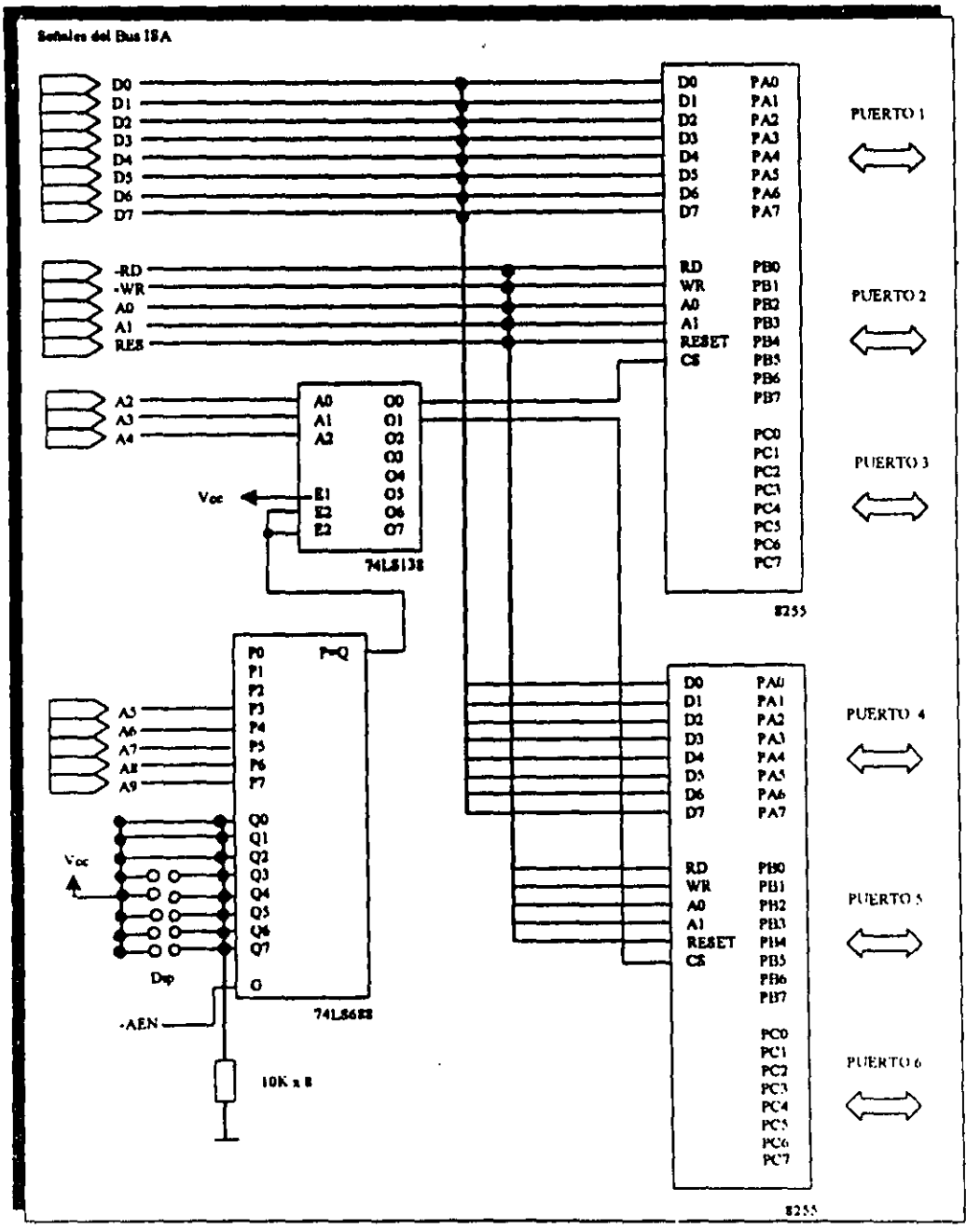


Figura 3.5 Circuito electronico de la Tarjeta I/O de proposito general programable

El bus de datos de la PC es conectado al bus de datos del PPI. Las señales de control RD, WR, RESET y las líneas de dirección A0 y A1 son las señales para programar y controlar los puertos del PPI 8255.Las líneas de habilitación A0 y A1 sirven para seleccionar el puerto A, B, C o el registro de control del PPI 8255. El registro de control se programa para trabajar en el Modo 0, haciendo que el puerto A sea definido como salida y el puerto B como entrada.

Para poder manejar más de un PPI se incorporó un LS138. Las líneas A2, A3 y A4 controlan la señal de habilitación del multiplexor LS138, el cual habilita al PPI correspondiente. Por lo tanto la tarjeta I/O de propósito general tiene la capacidad de ser expandida a 8 PPI's (llegando a tener una arquitectura de 24 puertos I/O).

El dispositivo comparador LS688 tiene como objetivo asegurar el uso de los puertos externos a partir de la dirección 0300H y además recibe la señal de habilitación de la PC (AEN). Si lo anterior es correcto este dispositivo habilita el multiplexor LS138.

Cada PPI ocupa 4 puertos libres externos para su funcionamiento y la siguiente Figura 3.6 se muestra la dirección y el puerto respectivo para la expansión a 8 PPI's de la Tarjeta I/O.Para la Tarjeta I/O se ocupan los puertos que inician a partir de la dirección 0300H.

Dirección	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Descripción
0300H	0	0	1	1	0	0	0	0	0	0	0	0	Puerto A
0301H	0	0	1	1	0	0	0	0	0	0	0	1	Puerto B
0302H	0	0	1	1	0	0	0	0	0	0	1	0	Puerto C
0303H	0	0	1	1	0	0	0	0	0	0	1	1	Control
0304H	0	0	1	1	0	0	0	0	0	1	0	0	Puerto A
0305H	0	0	1	1	0	0	0	0	0	1	0	1	Puerto B
0306H	0	0	1	1	0	0	0	0	0	1	1	0	Puerto C
0307H	0	0	1	1	0	0	0	0	0	1	1	1	Control
0308H	0	0	1	1	0	0	0	0	1	0	0	0	Puerto A
0309H	0	0	1	1	0	0	0	0	1	0	0	1	Puerto B
030AH	0	0	1	1	0	0	0	0	1	0	1	0	Puerto C
030BH	0	0	1	1	0	0	0	0	1	0	1	1	Control
030CH	0	0	1	1	0	0	0	0	1	1	0	0	Puerto A
030DH	0	0	1	1	0	0	0	0	1	1	0	1	Puerto B
030EH	0	0	1	1	0	0	0	0	1	1	1	0	Puerto C
030FH	0	0	1	1	0	0	0	0	1	1	1	1	Control
0310H	0	0	1	1	0	0	0	1	0	0	0	0	Puerto A
0311H	0	0	1	1	0	0	0	1	0	0	0	1	Puerto B
0312H	0	0	1	1	0	0	0	1	0	0	1	0	Puerto C
0313H	0	0	1	1	0	0	0	1	0	0	1	1	Control
0314H	0	0	1	1	0	0	0	1	0	1	0	0	Puerto A
0315H	0	0	1	1	0	0	0	1	0	1	0	1	Puerto B
0316H	0	0	1	1	0	0	0	1	0	1	1	0	Puerto C
0317H	0	0	1	1	0	0	0	1	0	1	1	1	Control
0318H	0	0	1	1	0	0	0	1	1	0	0	0	Puerto A
0319H	0	0	1	1	0	0	0	1	1	0	0	1	Puerto B
031AH	0	0	1	1	0	0	0	1	1	0	1	0	Puerto C
031BH	0	0	1	1	0	0	0	1	1	0	1	1	Control
031CH	0	0	1	1	0	0	0	1	1	1	0	0	Puerto A
031DH	0	0	1	1	0	0	0	1	1	1	0	1	Puerto B
031EH	0	0	1	1	0	0	0	1	1	1	1	0	Puerto C
031FH	0	0	1	1	0	0	0	1	1	1	1	1	Control

Figura 3.6 Direcciones de puertos externos de la Tarjeta I/O de propósito general

3.2.1.6 PPI 8255 Programable Peripheral Interface

El circuito integrado 8255 es un controlador programable para la adaptación de periféricos que trabajan en paralelo. Tiene 24 pines de I/O que pueden ser programados individualmente en 2 grupos de 12 y usados en 3 modos de operación.

3.2.1.6.1 Descripción funcional

En la Figura 3.7 se muestra en diagrama de bloques la descripción interna de PPI 8255.

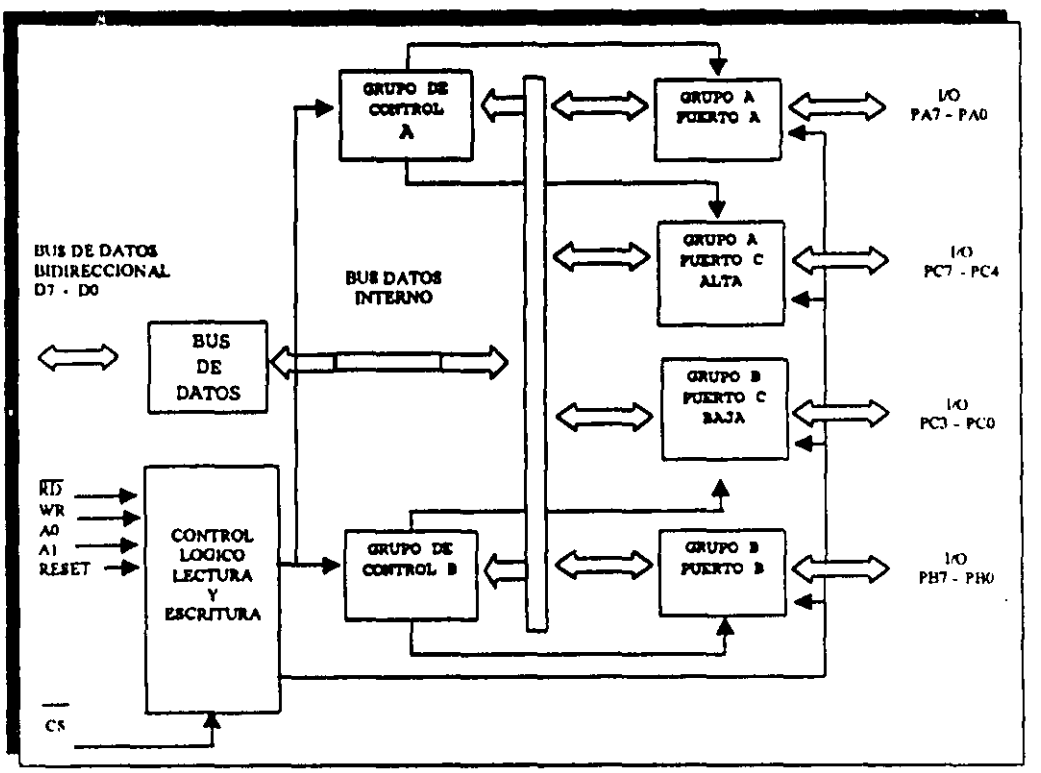


Figura 3.7 Descripción interna del PPI 8255

Bus de Datos: es de 8 bits, 3 estados y sirve de comunicación con la computadora.

Control lógico de Lectura y Escritura: la función de este bloque es manejar todas las transferencias internas y externas de datos y palabras de control o status.

Controles del Grupo A y Grupo B: La configuración de cada puerto se programa por software. Las palabras de control que manda la computadora son recibidas por el bloque de Control Lógico Lectura y

Escritura, manda los comandos al bloque de control de cada grupo a través del bus de datos interno. Los registros de la palabra de control son de solo escritura.
Control Grupo A: contiene al puerto A y la parte superior del puerto C (C7-C4)

Control Grupo B: contiene el puerto B y la parte inferior del puerto C (C3-C0)

Puertos A, B y C: son de 8 bits y pueden configurarse de distintas formas mediante software, pero cada uno tiene una función particular.

Puerto A: 8 bits de salida con latch/buffer y 8 bits de entrada con latch.

Puerto B: 8 bits de entrada/salida con latch/buffer y 8 bits de entrada con buffer.

Puerto C: 8 bits de salida con latch/buffer y 8 bits de entrada con buffer (sin latch). Este puerto puede dividirse en dos puertos de 4 bits. Cada puerto de 4 bits contiene un latch de 4 bits y puede usarse para señales de control de salida y para señales de status de entrada en conjunto con los puertos A y B.

Los pines son:

$\overline{CS}$: Chip select. Capacidad para poner en tri estado el bus de datos
$\overline{RD}$: Read
$\overline{WR}$: Write
A0 y A1: bits para acceder a los puertos o registro de control
RESET: inicialización del PPI. Borra el registro de control.

3.2.1.6.2 Operación Básica

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operación Leer
0	0	0	1	0	Puerto A → Bus Datos
0	1	0	1	0	Puerto B → Bus Datos
1	0	0	1	0	Puerto C → Bus Datos
1	1	0	1	0	Control → Bus Datos

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Operación Escribir
0	0	1	0	0	Bus Datos → Puerto A
0	1	1	0	0	Bus Datos → Puerto B
1	0	1	0	0	Bus Datos → Puerto C
1	1	1	0	0	Bus Datos → Control

A1	A0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Función Desactivar
X	X	X	X	1	Bus Datos → Tri Estado
X	X	1	1	0	Bus datos → Tri Estado

3.2.1.6.3 Programación del Registro de Control

Este registro sirve para programar el funcionamiento del PPI. Cuando el bit más significativo tiene un 1, los bits restantes tienen la función de controlar los grupos A y B. En la Figura 3.8 se describe la función de cada bit y su selección depende del usuario. Pero cuando el bit más significativo tiene un 0, se puede realizar la función bit set/reset del puerto C. Su funcionamiento se explica a continuación.

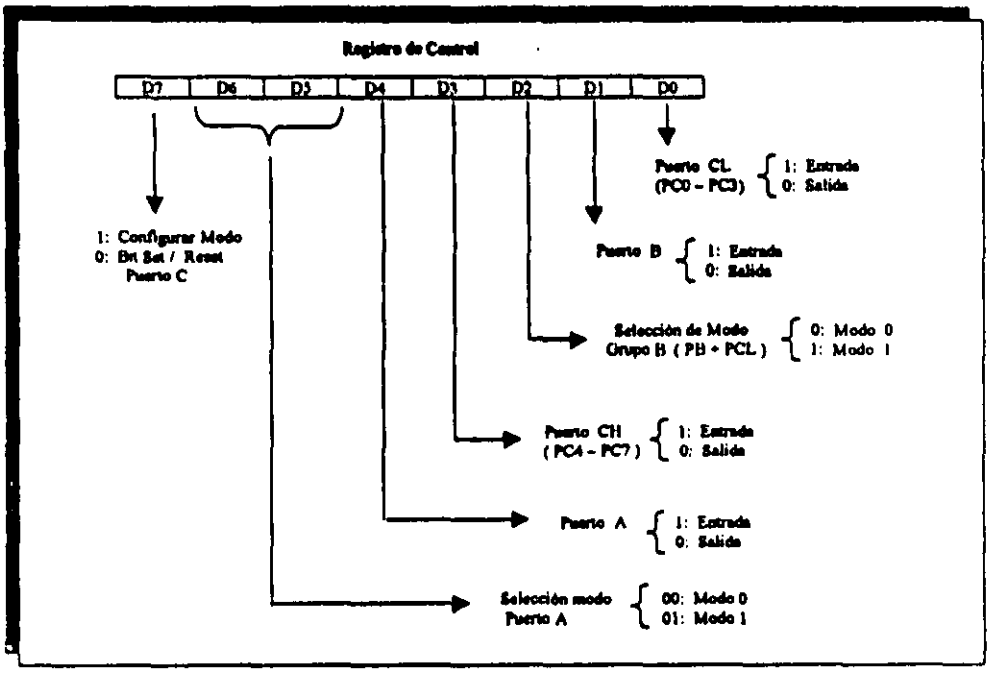


Figura 3.8 Registro de Control PPI 8255

3.2.1.6.4 Función Bit Set/Reset del Puerto C

Cuando el bit más significativo del registro de control tiene un 0, entonces cualquiera de los 8 bits del puerto C permite ser seteado o reseteado (Ver Figura 3.9).

Con D3 D2 y D1 se accede a determinado bit y en D0 se coloca su valor (set/reset).

Si el pin está como salida: el valor que tenga el bit set/reset será directamente enviado a la salida

Si el pin está como entrada: sirve para programar (enmascarar).

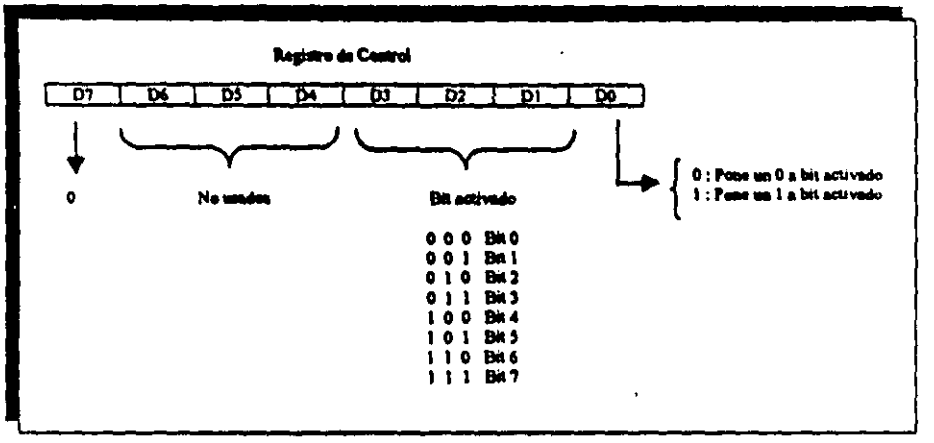


Figura 3.9 Función Set/Reset del Puerto C

3.2.1.6.5 Modos de Funcionamiento

Modo 0	Entrada / Salida Básica
Modo 1	Modo Pulsado de Entrada / Salida
Modo 2	Bus Bidireccional

Modo 0 (Basic Input/Output)

Provee operaciones de entrada o salida a cada uno de los tres puertos, sin Handshaking. En la Tabla 3.1 se muestra el valor del registro de control y su efecto en la configuración de los puertos A, B y C.

Características:

- 2 puertos de 8 bits y 2 puertos de 4 bits
- Cualquier puerto pude ser usado como entrada o como salida
- Las salidas se latchean
- Las entradas no se latchean
- Son posibles 16 configuraciones diferentes de entrada/salida

Tabla 3.1 Funcionamiento en el Modo 0

Registro de Control								Grupo A		n	Grupo B	
D7	D6	D5	D4	D3	D2	D1	D0	Puerto A	Puerto C (Alta)		Puerto B	Puerto C (Baja)
1	0	0	0	0	0	0	0	Salida	Salida	0	Salida	Salida
1	0	0	0	0	0	0	1	Salida	Salida	1	Salida	Entrada
1	0	0	0	0	0	1	0	Salida	Salida	2	Entrada	Salida
1	0	0	0	0	0	1	1	Salida	Salida	3	Entrada	Entrada
1	0	0	0	1	0	0	0	Salida	Entrada	4	Salida	Salida
1	0	0	0	1	0	0	1	Salida	Entrada	5	Salida	Entrada
1	0	0	0	1	0	1	0	Salida	Entrada	6	Entrada	Salida
1	0	0	0	1	0	1	1	Salida	Entrada	7	Entrada	Entrada
1	0	0	1	0	0	0	0	Entrada	Salida	8	Salida	Salida
1	0	0	1	0	0	0	1	Entrada	Salida	9	Salida	Entrada
1	0	0	1	0	0	1	0	Entrada	Salida	10	Entrada	Salida
1	0	0	1	0	0	1	1	Entrada	Salida	11	Entrada	Entrada
1	0	0	1	1	0	0	0	Entrada	Entrada	12	Salida	Salida
1	0	0	1	1	0	0	1	Entrada	Entrada	13	Salida	Entrada
1	0	0	1	1	0	1	0	Entrada	Entrada	14	Entrada	Salida
1	0	0	1	1	0	1	1	Entrada	Entrada	15	Entrada	Entrada

Comportamiento de las líneas

En la figura 3.10 se puede ver que los puertos de los grupos A y B pueden trabajar solamente como entrada o como salida del bus de datos.

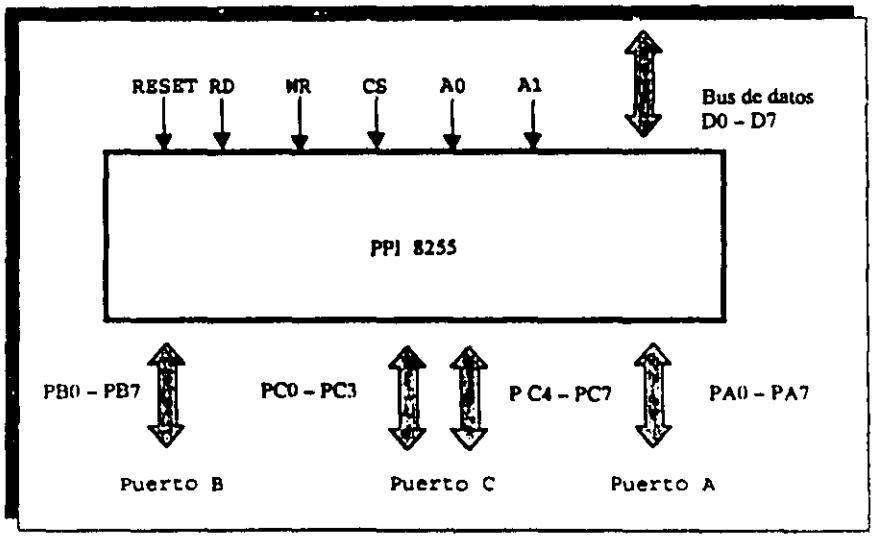


Figura 3.10 Lineas de control en el modo 0

Modo 1 (Strobed Input/Output)

El puerto A y el puerto B usan las líneas del puerto C para generar o aceptar señales de handshaking en modo de pulsos (strobes).

Características:

- 2 grupos (grupos A y B)
- Cada grupo contiene 8 bits de puerto de datos y 4 bits de puerto de control/datos.
- Cada puerto de datos de 8 bit puede usarse como entrada o como salida independientemente.
- Tanto las entradas como las salidas se latchean.
- El puerto de 4 bits se usa para control y status del puerto de datos de 8 bits.

Comportamiento de las líneas

En la figura 3.11 se muestra las señales de handshaking y su asignación a cada bit del puerto C.

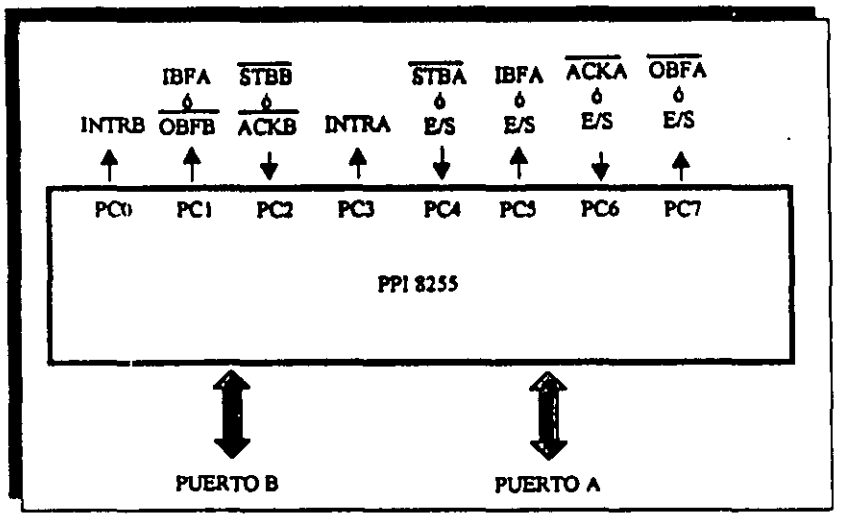


Figura 3.11 Líneas de control en el modo 1

Señales de Control cuando el Puerto (A o B) está definido como ENTRADA

STB (Strobe Input): Un nivel bajo en esta entrada, carga los datos en el latch de entrada.

IBF (Input Buffer Full F/F): Un nivel alto en esta salida indica que el dato ya fue cargado en el latch de entrada. IBF se pone en alto por un nivel bajo en STB y se pone en bajo debido al flanco positivo de la entrada RD.

INTR Un nivel alto en esta salida se puede usar para pedido de Interrupción. INTR se pone en alto cuando STB está en 1, IBF está en 1 e INTE está en 1. Se pone en 0 con el flanco negativo de RD

Señales de Control cuando el Puerto (A o B) está definido como SALIDA

OBF (Output Buffer F/F): esta salida se pone en bajo indicando que el 8088 escribió un dato en un puerto específico. El F/F de OBF se setea (nivel bajo) con el flanco positivo de la entrada WR y se resetea (nivel alto) cuando la entrada ACK tiene un nivel bajo.

ACK (Acknowledge Input): El periférico le informa al 8255 que leyó los datos provenientes del puerto A o B con un nivel bajo en esta entrada.

INTR (Interrupt Request): Un nivel alto en esta salida se puede usar para interrumpir al 8088 cuando el periférico aceptó el dato transmitido por el 8088. INTR se pone en 1 cuando ACK es un 1, OBF es un 1 e INTE es un 1. Se pone en cero con el flanco positivo de WR.
INTE A: está controlado por el bit set/reset de PC6
INTE B: está controlado por el bit set/reset de PC2

Modo 2 (Strobed Bidirectional Bus I/O)

Permite enviar datos a, o recibir datos de un periférico mediante un único bus de 8 bits bidireccional, con señales de handshaking para mantener el flujo de datos sobre el bus de datos de una manera similar al Modo 1. También se pueden generar interrupciones y dispone también de funciones de enable/disable.

Características:

- Usa sólo el Grupo A.
- Tiene un bus bidireccional de 8 bits (puerto A) y un puerto de control de 5 bits (puerto C).
- Se latchean tanto las entradas como las salidas.
- El puerto de control de 5 bits se usa para control y status del bus bidireccional de 8 bits.

Señales de Control

INTR (Interrupt Request): Un nivel alto en esta salida sirve para interrumpir al 8088 tanto en operaciones de entrada como de salida.

Operaciones de Salida

OBF (Output Buffer Full): Esta salida pasa a un nivel bajo para indicar que el 8088 escribió un dato en el puerto A.

ACK (Acknowledge): Un nivel bajo en esta entrada habilita el buffer de salida del puerto A para enviar datos al periférico (el periférico lee al 8255). En otro caso, el buffer de salida permanece en alta impedancia.

INTE 1 (El flip-flop INTE asociado con OBF) : controlado por el bit set/reset de PC6.

Operaciones de Entrada

STB (Strobe Input): un nivel bajo en esta entrada carga el dato en el latch de entrada.

IBF (Input Buffer Full F/F). Un nivel alto en esta salida indica que el dato fue cargado en el latch de entrada.

INTE 2 (El flip-flop INTE asociado con IBF): controlado por bit set/reset de PC4.

Consideraciones en Combinaciones de Modos

Se pueden combinar modos cuando no se usan todos los bits del puerto C para control o status.

Los bits restantes se pueden usar como:

Si se programan como entradas: Se puede acceder a todas las líneas de entrada durante una lectura normal del puerto C.

Si se programan como salidas: A los bits más significativos del puerto C (PC7-PC4) debe accederse individualmente usando el bit set/reset function. A los bits menos significativos del puerto C (PC3-PC0) puede accederse usando bit set/reset function o puede accederse como un trio escribiendo en el puerto C.

Capacidades de Corriente del Puerto B y C

Cualquier conjunto de 8 buffers de salida, seleccionados del puerto B y C puede suministrar 1 mA a 1.5 V.

Esto permite que el 8255 pueda alimentar directamente a drivers de tipo darlington y a displays de alto voltaje que requieran tal corriente.

3.2.1.6.6 Lectura del Estado del Puerto C

En el modo 0 el puerto C transfiere datos hacia o desde el periférico. Cuando el 8255 se programa para funcionar en modo 1 o 2, el puerto C genera o acepta señales de handshaking con el periférico. Leyendo el contenido del puerto C, le permite al programador testear o verificar el "estado" de cada periférico y cambiar el flujo del programa si es necesario.

Una operación de lectura normal del puerto C permite leer el estado.

Formato de la Palabra Estado Modo 1

Configuración para Entrada

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB

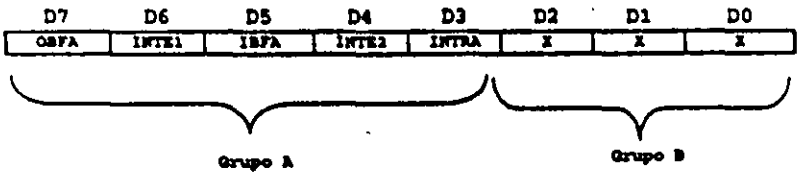
Grupo A Grupo B

Configuración para Salida

D7	D6	D5	D4	D3	D2	D1	D0
OBFA	INTEA	I/O	I/O	INTRA	INTEB	OBFB	INTRB

Grupo A Grupo B

Formato de la Palabra Estado Modo 2



(Definido por la selección del modo 0 o modo 1)

3.2.2 Modelado de los motores de paso usados

Durante el desarrollo de la tesis se llego a trabajar con dos tipos de motores de paso el KP4M4 y el SEE 00227.

3.2.2.1 MOTOR KP4M4

Motor de pasos usado en la fase de pruebas.

Fabricante:

Tandon
Servo Japan Motors

Caracteristicas:

Hibrido Unipolar
3.6 ° por paso completo
1.8 ° por medio paso
150 Ω por devanado
100 mA por devanado
12 V de alimentación

Esquema:

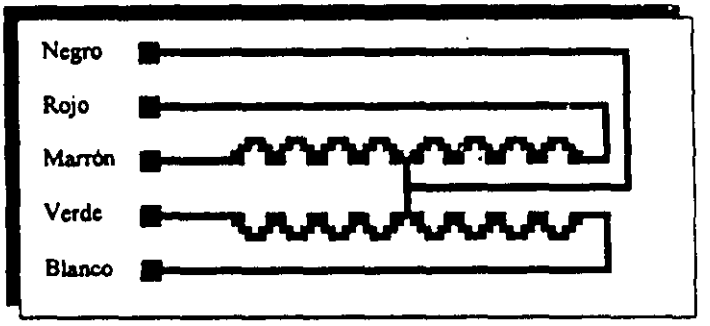


Figura 3.12 Modelo del KP4M4

Conector:

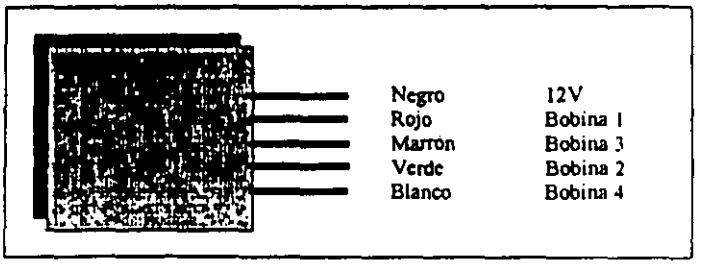


Figura 3-13 Conector del KP4M4

3.2.1.2 MOTOR SEE 00227

Motor de pasos usado para implementar el posicionador bidimensional.

Fabricante:

Japan

Características:

Híbrido Unipolar
7.5 ° por paso completo
3.75 ° por medio paso
12 Ω por devanado
200 mA por devanado
12 V de alimentación

Esquema:

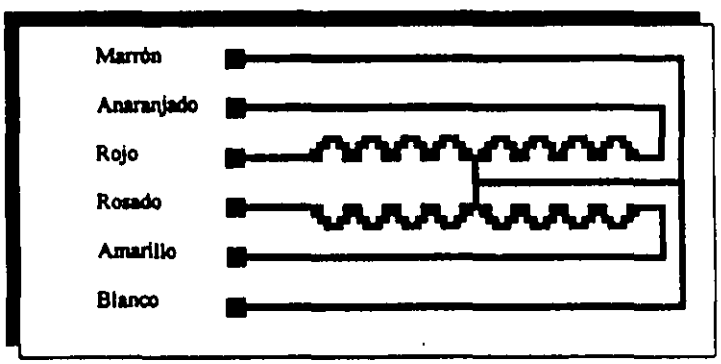


Figura 3.14 Modelo del SEE 00227

Conector:

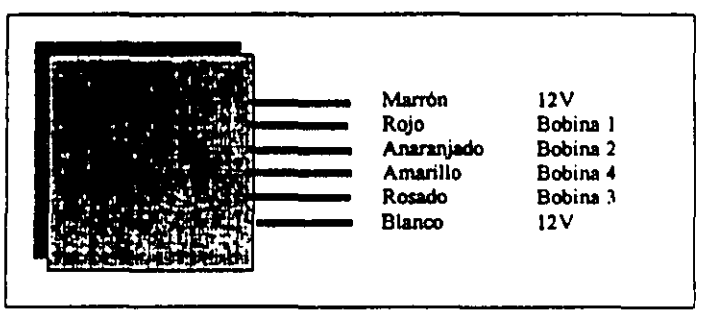


Figura 3.15 Conector del SEE 00227

3.2.3 Driver de potencia

Cada motor de pasos es manejado por medio de un driver unipolar. Los transistores tienen una configuración darlington y entregan pulsos de corriente del orden de 1.27A por cada devanado. Al finalizar la secuencia de movimiento se apagan todos los transistores para no dañar el motor de pasos por sobrecalentamiento.

En la Figura 3.16 se presenta el circuito del driver unipolar y las terminales del motor de pasos. Para la alimentación se puede ocupar una fuente externa (en el trabajo se utilizo además la fuente de la PC). La secuencia de paso es controlada por medio puerto. Por lo tanto se ocupo el puerto A para controlar los dos motores de paso del posicionador bidimensional.

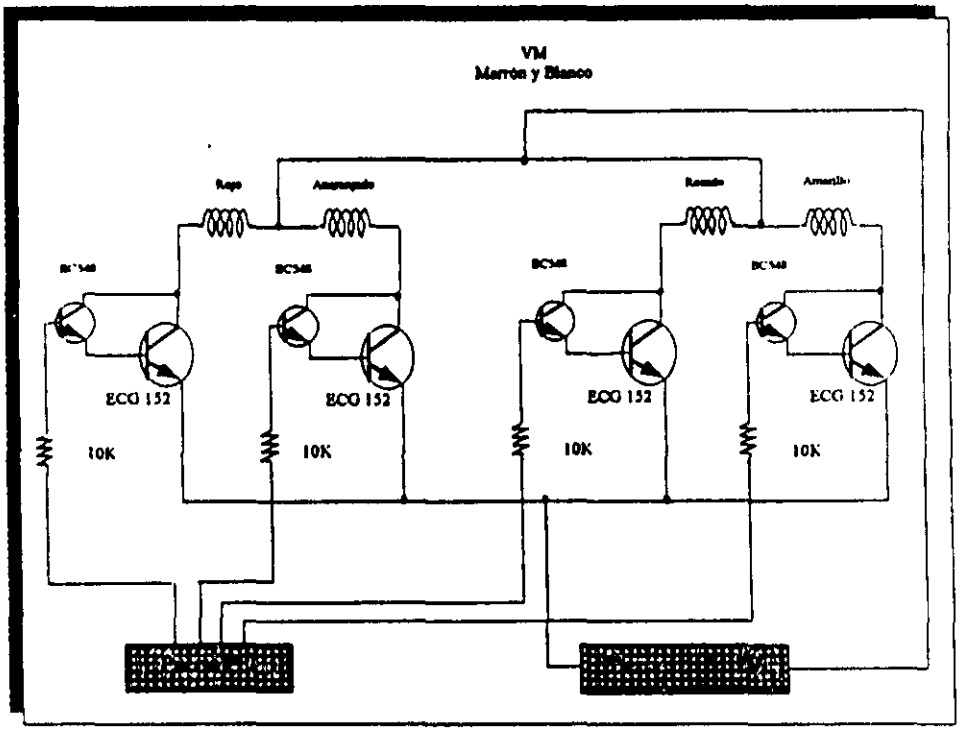


Figura 3.16 Driver Unipolar del posicionador bidimensional

Datos y Cálculos:

$\beta_{548} = 150$; $\beta_{152} = 150$; Por lo tanto $\beta_{darlington} = 22500$;
$V_{BE} = 1.4V$; $V_{BB} = 1.967V$; $V_{CEsat} = 0.82V$; $R_B = 10K$; $R_C = 6\Omega$ (Devanado);

$$I_B = (1.967 - 1.4) / 10 = 0.0567 \text{ mA}$$

$$I_C = 22500 \times 0.0567 \text{ mA} = 1.27 \text{ A}$$

$$V_{CE} = 12 - (1.27 \times 6) = 4.48V;$$

Como $V_{CE} > V_{CEsat}$ se esta trabajando en la región activa

3.2.4. Implementación del Posicionador Bidimensional

El posicionador bidimensional fue construido con material de reciclaje usado en impresoras de matriz de puntos. Los ejes son montados uno sobre el otro con un ángulo de 90° para formar un eje cartesiano bidimensional. Utilizan bandas dentadas para transmitir el movimiento de los motores de paso.

A esta estructura se le agrego switches en cada eje con el fin de detectar el inicio y final de carrera. Además esto permite que el VI MOTOR pueda detectar automáticamente las dimensiones de los ejes y evitar durante el funcionamiento del posicionador bidimensional un descarrilamiento.

El posicionador bidimensional cuenta con tres conexiones externas. Dos de ellas van a los drivers unipolares para manejar cada motor de pasos (Ver Figura 3.15) y la otra sirve como retroalimentación. La señal de retroalimentación es conectada al puerto B de la Tarjeta I/O, la cual sirve para monitorear el inicio y final de carrera.

3.3 Diseño e implementación de componentes de software

3.3.1 Programación de la Tarjeta I/O

Para programar la Tarjeta I/O se diseño el módulo Configurar dentro del VI MOTOR. La definición del funcionamiento de los puertos esta determinado por el estado del registro de control del PPI 8255. En la Tabla 3.2 se da define el funcionamiento de la Tarjeta I/O en el modo 0 por medio del registro de control del PPI 8255, es decir que los puertos funcionen como entradas o salidas.

Tabla 3.2

Registro de Control								Grupo A		n	Grupo B	
D7	D6	D5	D4	D3	D2	D1	D0	Puerto A	Puerto C (Alta)		Puerto B	Puerto C (Baja)
1	0	0	0	0	0	0	0	Salida	Salida	0	Salida	Salida
1	0	0	0	0	0	0	1	Salida	Salida	1	Salida	Entrada
1	0	0	0	0	0	1	0	Salida	Salida	2	Entrada	Salida
1	0	0	0	0	0	1	1	Salida	Salida	3	Entrada	Entrada
1	0	0	0	1	0	0	0	Salida	Entrada	4	Salida	Salida
1	0	0	0	1	0	0	1	Salida	Entrada	5	Salida	Entrada
1	0	0	0	1	0	1	0	Salida	Entrada	6	Entrada	Salida
1	0	0	0	1	0	1	1	Salida	Entrada	7	Entrada	Entrada
1	0	0	1	0	0	0	0	Entrada	Salida	8	Salida	Salida
1	0	0	1	0	0	0	1	Entrada	Salida	9	Salida	Entrada
1	0	0	1	0	0	1	0	Entrada	Salida	10	Entrada	Salida
1	0	0	1	0	0	1	1	Entrada	Salida	11	Entrada	Entrada
1	0	0	1	1	0	0	0	Entrada	Entrada	12	Salida	Salida
1	0	0	1	1	0	0	1	Entrada	Entrada	13	Salida	Entrada
1	0	0	1	1	0	1	0	Entrada	Entrada	14	Entrada	Salida
1	0	0	1	1	0	1	1	Entrada	Entrada	15	Entrada	Entrada

El usuario dentro del módulo solamente escoge el puerto y escoge como debe funcionar. Para el CVPB el puerto A se define como salida y el puerto B como entrada.

3.3.2 Control manual de motores de paso

El VI MOTOR cuenta con un módulo donde se puede controlar manual y en forma independiente cada uno de los ejes del posicionador bidimensional. Se da el nombre de control manual porque básicamente se habilita controles de entrada para que desde teclado el usuario proporcione la distancia en pasos que debe recorrer cada eje.

La rotación de los motores de paso sigue la secuencia mediopaso mostrada en la Tabla 3.3:

Tabla 3.3

PASO	Bobina 1	Bobina 2	Bobina 3	Bobina 4
1	Apagado	Apagado	Apagado	Encendido
2	Apagado	Encendido	Apagado	Encendido
3	Apagado	Encendido	Apagado	Apagado
4	Apagado	Encendido	Encendido	Apagado
5	Apagado	Apagado	Encendido	Apagado
6	Encendido	Apagado	Encendido	Apagado
7	Encendido	Apagado	Apagado	Apagado
8	Encendido	Apagado	Apagado	Encendido

Para un movimiento hacia la derecha y derecha se sigue la secuencia mostrada en las Figura 3.17 y estas secuencias se implementaron con rutinas de software:

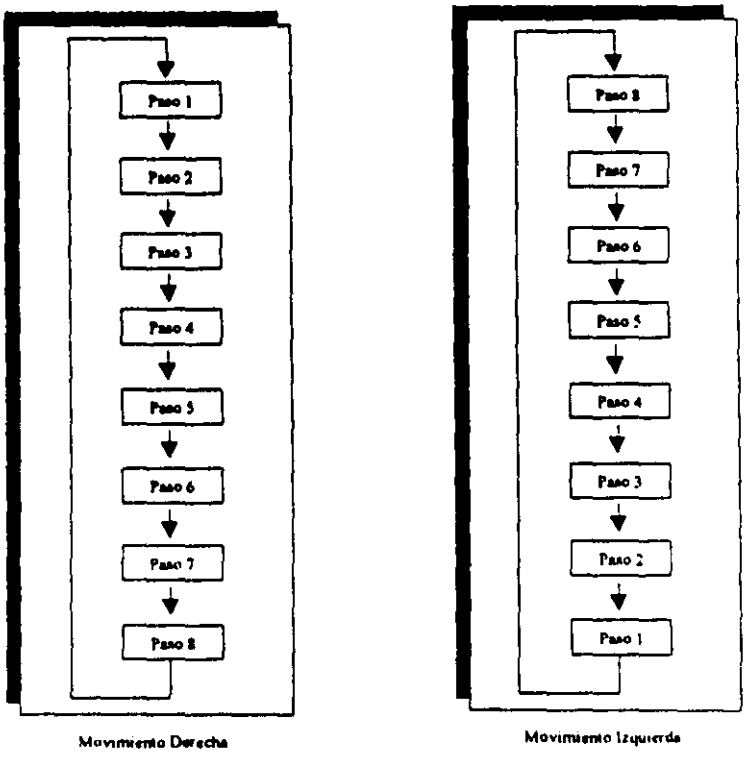


Figura 3.17

CONTROL VIRTUAL DE UN POSICIONADOR BIDIMENSIONAL

Descripción del Problema

El control del movimiento por medio de motores de paso ha tenido un auge importante en posicionadores y en robótica. Se han desarrollado herramientas integradas de software y hardware para el control exacto y preciso en posicionadores de una, dos y tres dimensiones cuya resolución llega a miles de pasos por revolución. El costo de estas herramientas es elevado y muchas veces su grado de resolución en el movimiento sobrepasa nuestras necesidades.

Ante este panorama el trabajo de tesis desarrollado tuvo como objetivo diseñar un instrumento virtual para el manejo de cualquier posicionador bidimensional trabajando los motores de paso en su modo mediopaso. Esto hizo necesario usar herramientas de software (LabWindows/CVI) y el hardware de un posicionador bidimensional. Se desarrollo e integro un instrumento virtual llamado VI MOTOR para llevar a cabo el control virtual del posicionador bidimensional desde una PC. La resolución alcanzada es suficiente como para construir un taladro electrónico, para lo cual es necesario agregar a este trabajo un eje ortogonal manejado por un motor en DC y reprogramar el VI MOTOR para llevar a cabo nuevas funciones. El costo de la circuiteria de interface y de potencia para el posicionador es relativamente barato, se usan circuitos integrados TTL que son fáciles de encontrar.

3.1 Estructura del Sistema CVPB

El sistema CVPB mostrado en la figura 3.1 tiene una conexión en lazo abierto y esta constituido por los siguientes elementos:

- Computadora : Donde reside el software del VI MOTOR.
- Tarjeta I/O de propósito general programable: Dispositivo electrónico que nos permite conectarnos con el bus de la PC.

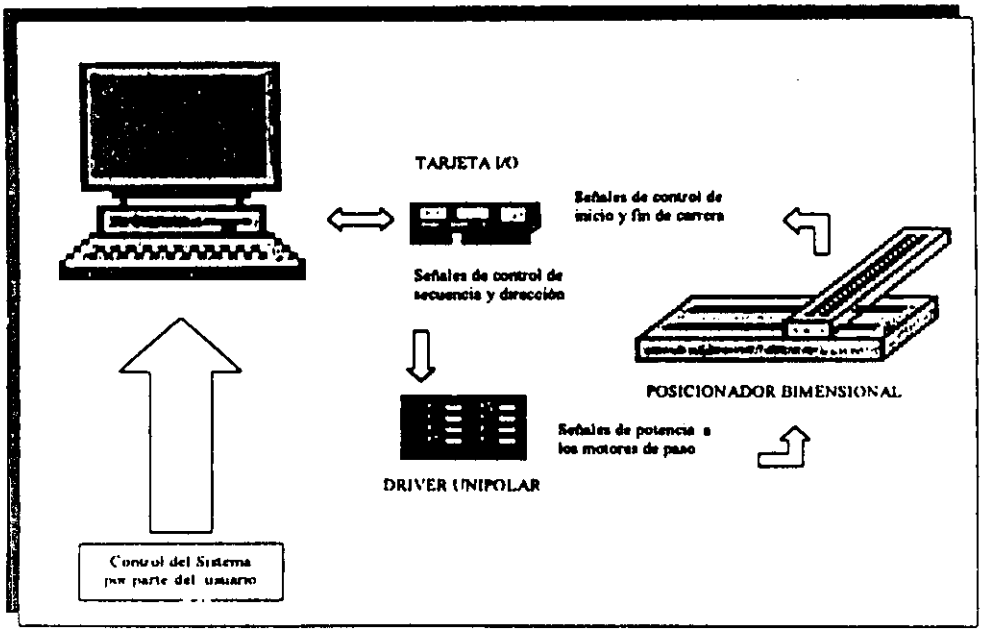


Figura 3.1 Sistema de Control Virtual del Posicionador Bidimensional

3.4.2 Control Manual del Posicionador Bidimensional

Este módulo se muestra en la Figura 3.20 y permite controlar independientemente cada eje del posicionador bidimensional. Desde teclado se ingresa el número de pasos a moverse. El procedimiento es el siguiente:

- Dar click sobre el botón Reset para mover el posicionador bidimensional a su origen. Este punto es el (0,0) del eje cartesiano.
- Con el botón superior izquierdo seleccionar el Motor Eje X o Motor Eje Y. Esto permite seleccionar el eje cartesiano donde nos moveremos.
- Posicionar el cursor del mouse dentro de las cajas de texto y dar desde teclado la distancia en pasos. Si el número de pasos sobrepasa los límites de los ejes, automáticamente el VI MOTOR detecta y simplemente no se mueve.
- Dar click sobre los botones Izquierda, Derecha, Arriba o Abajo colocados encima de las cajas de texto. Esto botones dan la orden de moverse en la dirección seleccionada del eje cartesiano.
- Opcionalmente para visualizar el movimiento y el estado de los devanados de los motores, se selecciona del control Rate la velocidad de movimiento de los ejes.
- Opcionalmente para un posicionamiento manual unitario se puede seleccionar los botones de dirección. Al dar click sobre estos botones de dirección, el movimiento de los ejes se realiza paso a paso.

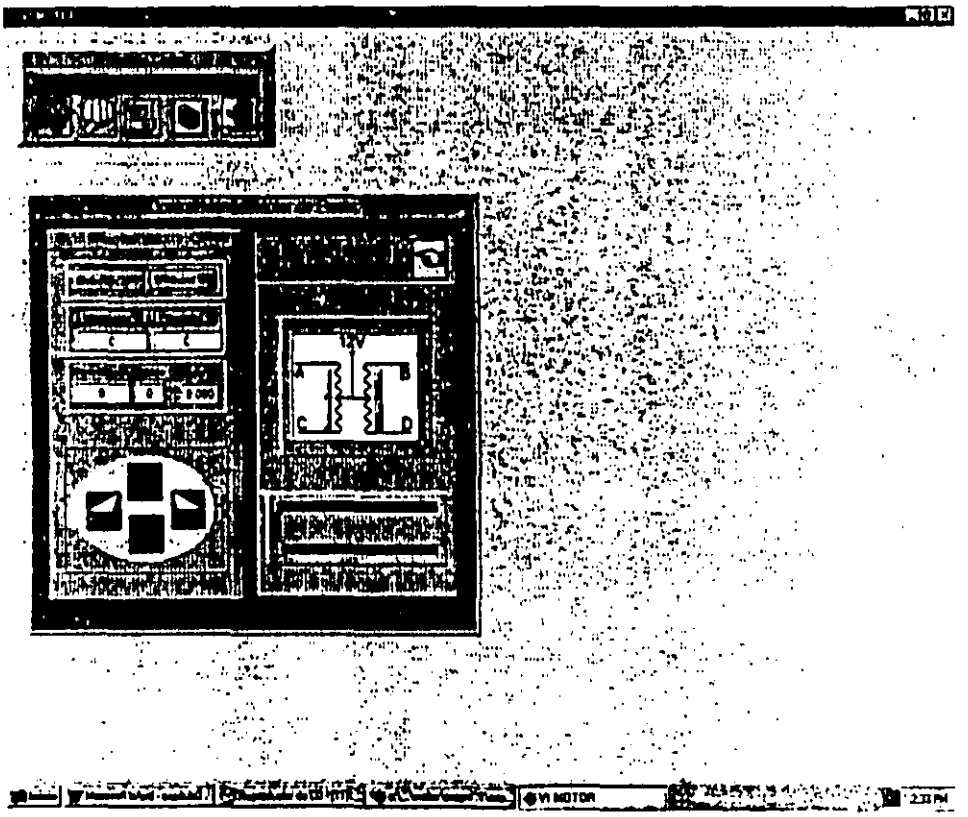


Figura 3.20 Modulo Control Manual del Posicionador Bidimensional

Como elementos de visualización del estado del movimiento y los motores se cuenta con:

- El indicador Pasos Eje X o Pasos Eje Y. Aquí se visualiza el número de pasos recorridos por el posicionador bidimensional a partir del origen.
- El indicador Estado. En el se puede apreciar el número de estado en el que se encuentra los devanados de los motores de paso.
- Un indicador de las fases encendidas. En el se puede apreciar gráficamente las fases encendidas del motor de pasos.
- Una barra de desplazamiento absoluto. En el se indica gráficamente el desplazamiento en pasos del eje a partir del origen.
- Una barra de desplazamiento relativo. En el se indica gráficamente el desplazamiento en pasos del posicionamiento actual.

3.4.3 Control Automático del Posicionador Bidimensional

Este módulo se muestra en la Figura 3.21 y permite el funcionamiento automático del posicionador bidimensional. Aquí se representa virtualmente el eje cartesiano del posicionador y con el cursor del ratón se puede seleccionar el punto destino. El procedimiento es el siguiente:

- Dar click sobre el botón Reset para mover el posicionador bidimensional a su origen. Este punto es el (0,0) del eje cartesiano. Esta acción permite además encontrar las dimensiones de los ejes del posicionador bidimensional y el área virtual del posicionador bidimensional se habilitará.

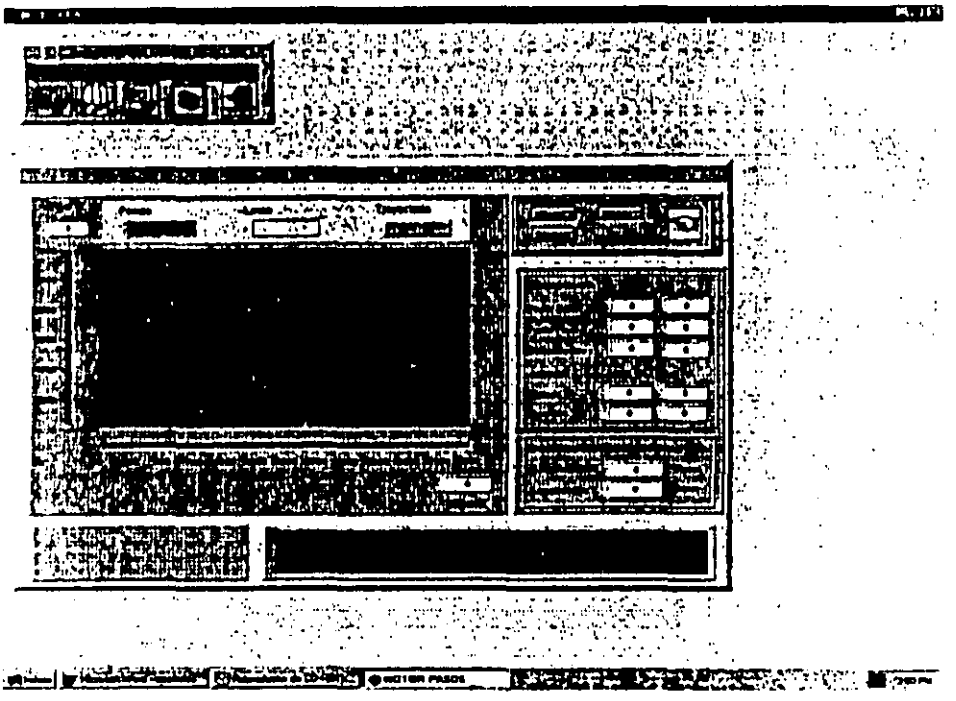


Figura 3.21 Módulo Control Automatico del Posicionador Bidimensional

- Desplazar el cursor del ratón sobre el área virtual del posicionador bidimensional y dar click cuando se haya seleccionado el punto destino a donde queremos movernos. El posicionador bidimensional real automáticamente se moverá hasta alcanzar el punto destino.
- Dar Click sobre el botón Origen cuando queremos regresar al punto (0,0) de nuestro posicionador sin tener que nuevamente resetear.

Como elementos de visualización se cuenta con:

- Indicadores de Reset. En ellos se muestran mensajes que nos da el VI MOTOR mientras encuentra el origen y las dimensiones del posicionador en pasos de los ejes X y Y.
- Indicadores de las coordenadas. En ellos se muestra las coordenadas hacia donde nos estamos moviendo.
- Indicadores Punto Inicial y Punto Final. En ellos se muestra el punto origen y el punto destino del movimiento que se va a realizar a escala.
- Indicadores Distancia Abs. En ellos se muestra la distancia absoluta en dirección de los ejes entre el punto inicial y el punto a donde queremos movernos a escala.
- Indicadores de Estado. En ellos se sigue el seguimiento de las fases encendidas en los devanados de los motores de paso.
- Indicadores de Posición. En ellos se presenta el punto real donde nos encontramos sobre el posicionador bidimensional.

3.4.4. Ayuda del Sistema CVPB

Este módulo se muestra en la Figura 3.22 y sirve como información de ayuda acerca del sistema explicando detalladamente los alcances y el funcionamiento del sistema. El procedimiento es el siguiente:

- Dar click sobre cualquiera de los iconos. La información sobre el tema seleccionado aparece inmediatamente explicando y dando los alcances del sistema.

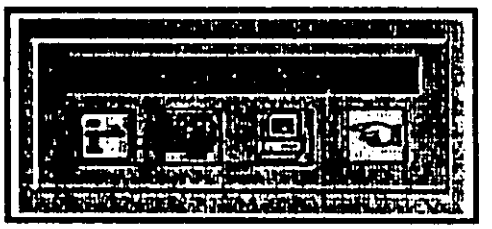


Figura 3.22 Módulo Ayuda del Sistema

Todo el código del VI MOTOR se da en el Apéndice G.

Resultados

- El posicionador bidimensional implementado con motores de paso tiene una elevada precisión y exactitud de movimiento. Estos dispositivos electromecánicos además nos reducen circuiteria, ya que tienen la capacidad de ser trabajados en lazo abierto y no necesariamente requieren retroalimentación.

- La precisión y exactitud del posicionador bidimensional es función de la velocidad y la carga conectada a la flecha de los motores de paso. Es decir mientras el torque generado por los devanados del motor de pasos sea mayor al torque para el arranque de la carga no se perderán pasos.

- El trabajar los motores de paso en el modo mediopaso, duplican la resolución de estos dispositivos. Para el posicionador bidimensional se logro una resolución de 3.75° por paso.

- La generación de la secuencia de movimiento en mediopaso es implementada por software. Es decir una rutina de software viene a ser el "instrumento" ya que realiza las funciones de un generador driver de secuencia de mediopaso en ambas direcciones (Figura 3.17).

- En el modo mediopaso la resolución del motor de pasos SEE 00227 es de 96 pasos por revolución. Es decir que por cada paso gira 3.75°. Con un arreglo de engranes se puede modificar la equivalencia en distancia longitudinal por cada paso.

- La velocidad de giro de los motores de paso es dependiente de la velocidad del microprocesador de la PC y de la velocidad de respuesta del dispositivo del motor de pasos. En el posicionador las frecuencias de paso fue de aproximadamente 2000 Hz.

Conclusiones

- La instrumentación virtual es un nuevo paradigma de cambio que ha venido a revolucionar la instrumentación. La PC puede ser convertida de una manera rápida en un elemento de medición y a la vez de control sobre cierto proceso.

- Para la realización de un instrumento virtual en su manera básica solo es necesario tener una tarjeta DAQ digital que nos sirva como elemento de interface hacia la PC. Para aplicaciones mayores se deben de agregar elementos acondicionadores de señal.

- Los motores de paso tienen su máxima resolución cuando son trabajados en el modo micropasos. Su campo de aplicación esta en medicina donde la abertura y cerrado de válvulas es crítico. También tiene un auge dentro de la robótica para un mejor control en sus grados de movimiento.

- El posiconador bidimensional puede ser aplicado para el movimiento de objetos dentro de ambientes peligrosos para el ser humano. El control se puede llevar a cabo desde una PC localizada fuera de la zona crítica y contar con un panel de control y un área virtual que simule el ambiente de trabajo

- El posicionador bidimensional mejora su precisión y exactitud de movimiento si en lugar de las bandas dentadas se usara ejes en forma de tornillo. Estos ejes permiten que al girar una revolución la flecha del motor se avance la distancia entre muescas adyacentes, que por lo general son fracciones de pulgadas.

Bibliografía

(1) STEPPING MOTORS: A guide to modern theory and practice
P.P. Acarnley
IEE CONTROL ENGINEERING SERIES 1984

(2) DATA ACQUISITION SEMINAR
National Instruments
1995

(3) ELECTRONICA & COMPUTADORAS
Publicaciones CEKIT
Buenos Aires, Argentina 1997

(4) INDUSTRIAL CIRCUITS APPLICATION NOTE
ERICSSON 1998

(5) STEPPER MOTORS CONTROL UNITS POSITIONING SYSTEMS
Phytron 1998

(6) MICROPROCESADORES
Angulo 1984

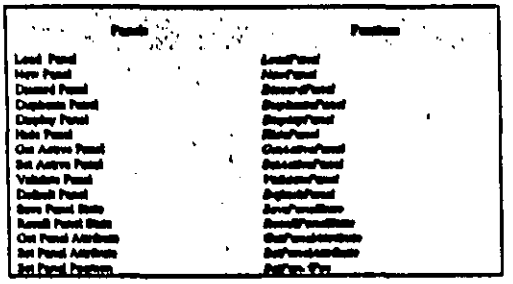
(7) MANUALES DE LABWINDOWS/CVI
National Instruments 1995

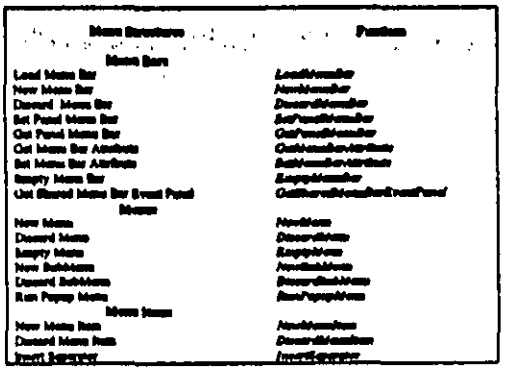
(8) CIRCUITOS ELECTRONICOS
Schilling - Belove
Mc Graw Hill 1994

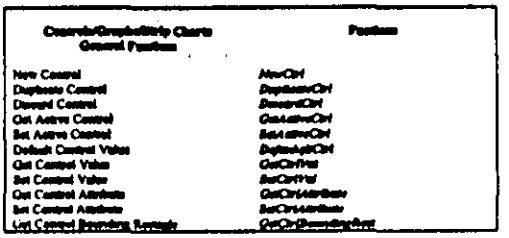
(9) DISEÑO ELECTRONICO
Savant - Roden - Carpenter
Addison - Wesley Iberoamericana 1992

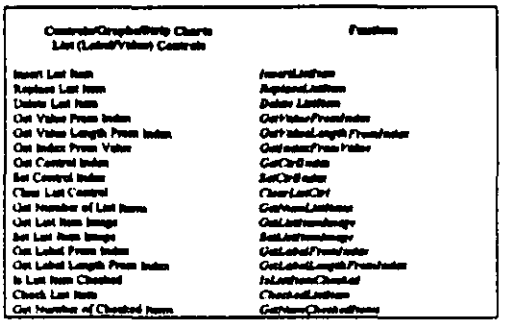
(10) EL LENGUAJE DE PROGRAMACION C
Kernighan - Ritchie
Prentice Hall 1991

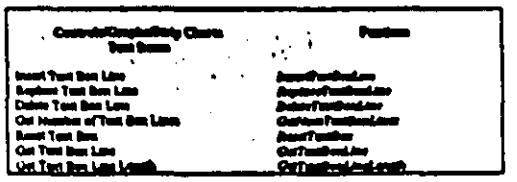
(11) C A REFERENCE MANUAL
Harbison - Steele
Prentice Hall 1995

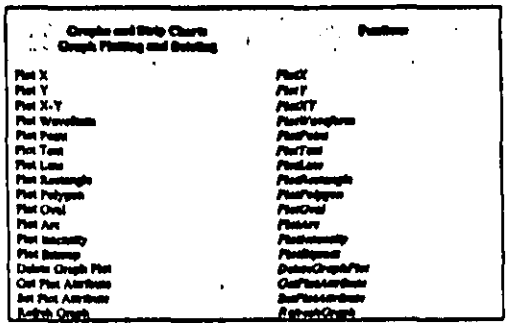

Panels
Functions
Load Panel
New Panel
Discard Panel
Duplicate Panel
Display Panel
Hide Panel
Get Active Panel
Set Active Panel
Validate Panel
Default Panel
Save Panel State
Recall Panel State
Get Panel Attribute
Set Panel Attribute
Set Panel Position

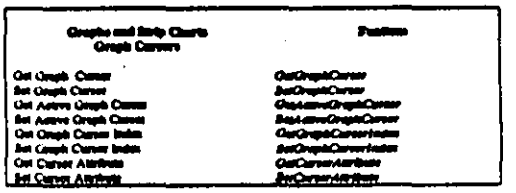

Menu Structure
Functions
Menu Bars
Load Menu Bar
New Menu Bar
Discard Menu Bar
Set Panel Menu Bar
Get Panel Menu Bar
Get Menu Bar Attribute
Set Menu Bar Attribute
Empty Menu Bar
Get Shared Menu Bar Event Panel
Menus
New Menu
Discard Menu
Empty Menu
New SubMenu
Discard SubMenu
Run Popup Menu
Menu Items
New Menu Item
Discard Menu Item
Insert Separator

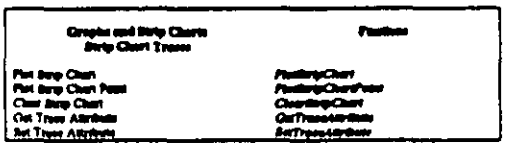

Controls/Graphs/Strip Charts
General Functions
Functions
New Control
Duplicate Control
Discard Control
Get Active Control
Set Active Control
Default Control Value
Get Control Value
Set Control Value
Get Control Attribute
Set Control Attribute
List Control Procedure Formats

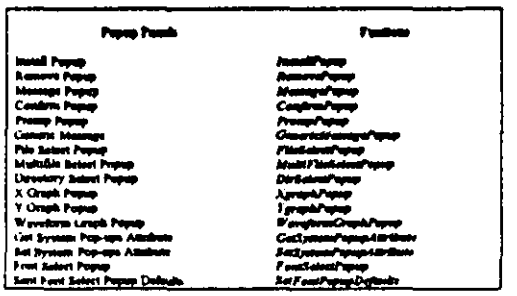

Controls/Graphs/Strip Charts
List (Label/Value) Controls
Functions
Insert List Item
Replace List Item
Delete List Item
Get Value From Index
Get Value Length From Index
Get Index From Value
Get Control Index
Set Control Index
Clear List Control
Get Number of List Items
Get List Item Image
Set List Item Image
Get Label From Index
Get Label Length From Index
Is List Item Checked
Check List Item
Get Number of Checked Items



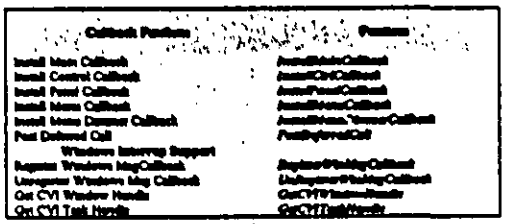




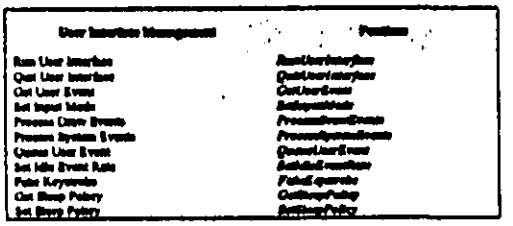




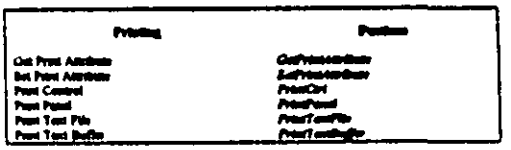
Graphs and Strip Charts Axis Scales	Functions
Get Axis Range	GetAxisRange
Set Axis Range	SetAxisRange



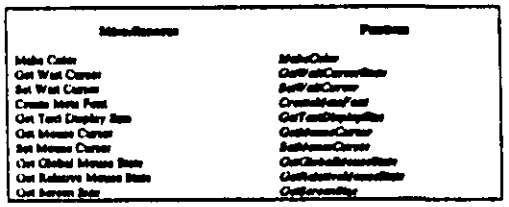
Callback Functions	Functions
Install Main Callback	InstallMainCallback
Install Control Callback	InstallCtrlCallback
Install Panel Callback	InstallPanelCallback
Install Menu Callback	InstallMenuCallback
Install Menu Dimmer Callback	InstallMenuDimmerCallback
Post Deferred Call	PostDeferredCall
Windows Interop Support	
Register Windows Msg Callback	RegisterWinMsgCallback
Unregister Windows Msg Callback	UnregisterWinMsgCallback
Get CVI Window Handle	GetCVIWindowHandle
Get CVI Task Handle	GetCVITaskHandle



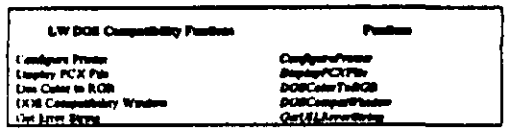
User Interface Management	Functions
Run User Interface	RunUserInterface
Quit User Interface	QuitUserInterface
Get User Event	GetUserEvent
Set Input Mode	SetInputMode
Process Draw Events	ProcessDrawEvents
Process System Events	ProcessSystemEvents
Queue User Event	QueueUserEvent
Set Idle Event Rate	SetIdleEventRate
Poke Keystroke	PokeKeystroke
Get Sleep Policy	GetSleepPolicy
Set Sleep Policy	SetSleepPolicy



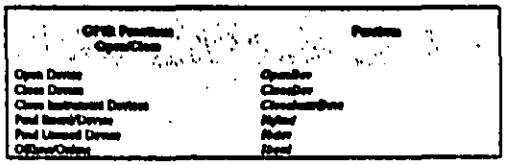
Printing	Functions
Get Print Attribute	GetPrintAttribute
Set Print Attribute	SetPrintAttribute
Print Control	PrintCtrl
Print Panel	PrintPanel
Print Text File	PrintTextFile
Print Text Buffer	PrintTextBuffer

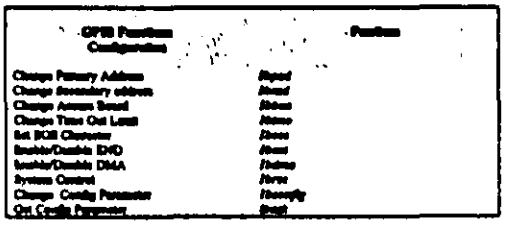


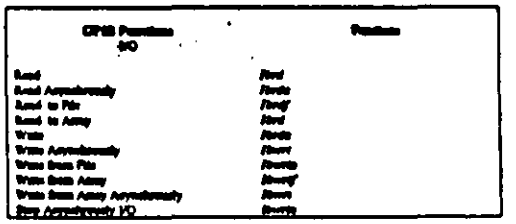
Miscellaneous	Functions
Make Color	MakeColor
Get Wait Cursor	GetWaitCursorState
Set Wait Cursor	SetWaitCursor
Create Metric Font	CreateMetricFont
Get Text Display Size	GetTextDisplaySize
Get Mouse Cursor	GetMouseCursor
Set Mouse Cursor	SetMouseCursor
Get Global Mouse State	GetGlobalMouseState
Get Relative Mouse State	GetRelativeMouseState
Get Screen Size	GetScreenSize

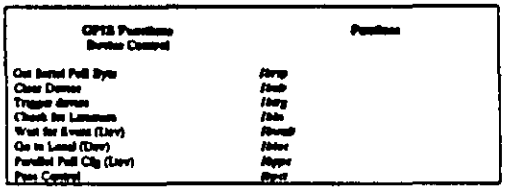


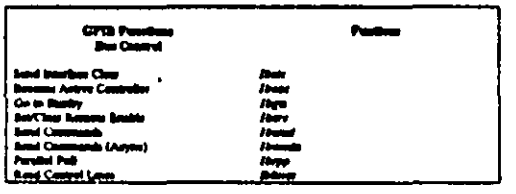
LW DOS Compatibility Functions	Functions
Configure Printer	ConfigurePrinter
Display PCX File	DisplayPCXFile
Dos Color to RGB	DOSColorToRGB
DOS Compatibility Window	DOSCompatWindow
Get Error String	GetDLLErrorString

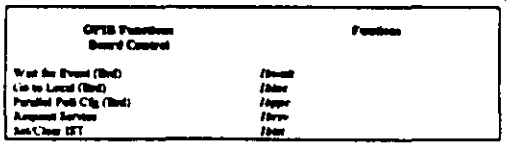




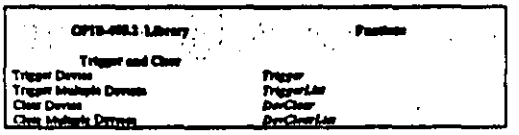




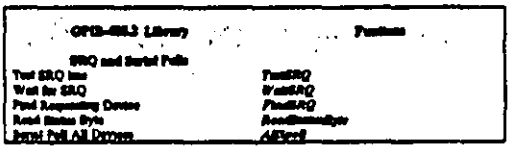




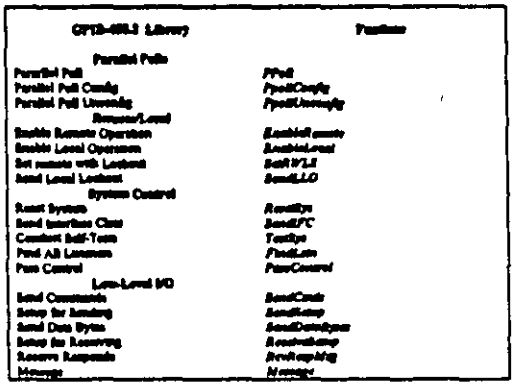
GPIB-488.2 Library	Function
Device I/O	
Send	Send
Send to Multiple Devices	SendList
Receive	Receive



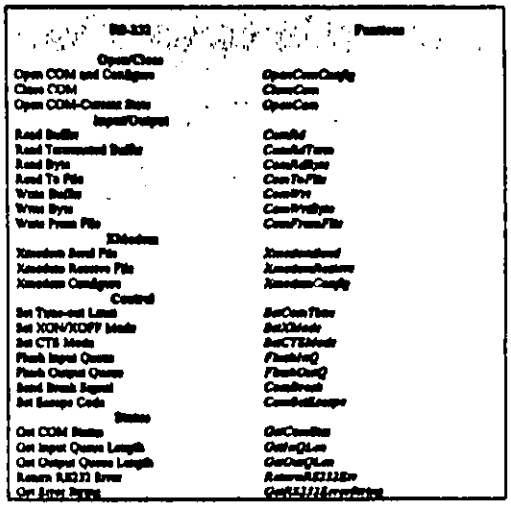
GPIB-488.2 Library	Function
Trigger and Clear	
Trigger Device	Trigger
Trigger Multiple Devices	TriggerList
Clear Device	DevClear
Clear Multiple Devices	DevClearList



GPIB-488.2 Library	Function
SRQ and Serial Polls	
Test SRQ line	TestSRQ
Wait for SRQ	WaitSRQ
Find Requesting Device	FindRQS
Read Status Byte	ReadStatusByte
Serial Poll All Devices	AllSpoll



GPIB-488.2 Library	Function
Parallel Polls	
Parallel Poll	PPoll
Parallel Poll Config	PpollConfig
Parallel Poll Unconfig	PpollUnconfig
Remote/Local	
Enable Remote Operation	EnableRemote
Enable Local Operation	EnableLocal
Set remote with Lockout	SetRWLS
Send Local Lockout	SendLLO
System Control	
Reset System	ResetSys
Send Interface Clear	SendIFC
Constant Self-Tests	TestSys
Find All Listeners	FindLstn
Pass Control	PassControl
Low-Level I/O	
Send Commands	SendCmds
Setup for Sending	SendSetup
Send Data Bytes	SendDataBytes
Setup for Receiving	RcvRespMsg
Receive Response	RcvRespMsg
Message	Message

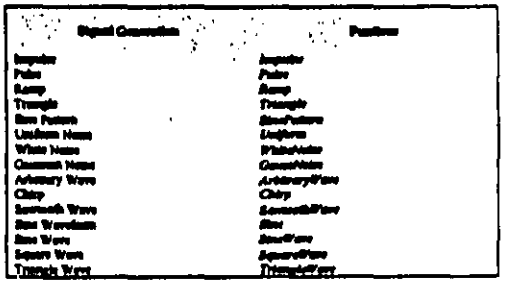


Server Routines	Function
Register DDE Server	*RegisterDDEServer*
Server DDE Write	*ServerDDEWrite*
Advise DDE Data Ready	*AdviseDDEDataReady*
Unregister DDE Server	*UnregisterDDEServer*

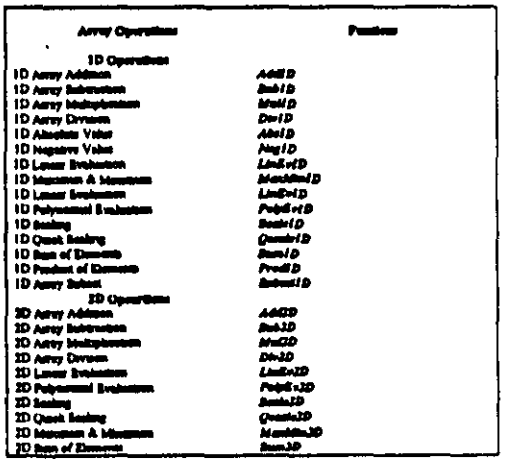
Client Routines	Function
Client DDE Execute	*ClientDDEExecute*
Client DDE Read	*ClientDDERead*
Client DDE Write	*ClientDDEWrite*
Connect To DDE Server	*ConnectToDDEServer*
Set Up DDE Hot Link	*SetUpDDEHotLink*
Set Up DDE Warm Link	*SetUpWarmLink*
Disconnect From DDE Server	*DisconnectFromDDEServer*
Get Error String	*GetDDEErrorString*

Server Routines	Routines
Register TCP Server	RegisterTCPServer
Server TCP Write	ServerTCPWrite
Server TCP Write	ServerTCPServer
Unregister TCP Server	UnregisterTCPServer

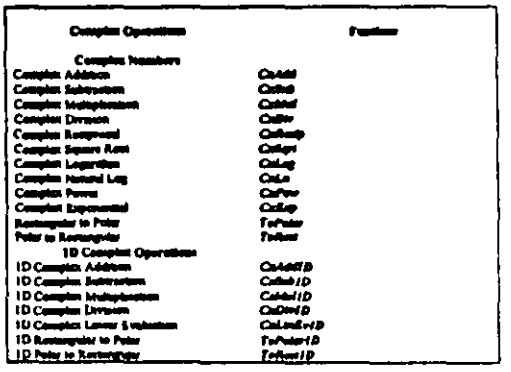
Client Routines	Routines
Connect To TCP Server	ConnectToTCPServer
Client TCP Read	ClientTCPRead
Client TCP Write	ClientTCPWrite
Disconnect From TCP Server	DisconnectFromTCPServer
Get Error Status	GetTCPErrorStatus



Signal Generation	Function
Impulse	*Impulse*
Pulse	*Pulse*
Ramp	*Ramp*
Triangle	*Triangle*
Sine Pattern	*SinePattern*
Uniform Noise	*UniNoise*
White Noise	*WhiteNoise*
Gaussian Noise	*GaussNoise*
Arbitrary Wave	*ArbitraryWave*
Chirp	*Chirp*
Sawtooth Wave	*SawtoothWave*
Sine Waveform	*Sine*
Sine Wave	*SineWave*
Square Wave	*SquareWave*
Triangle Wave	*TriangleWave*

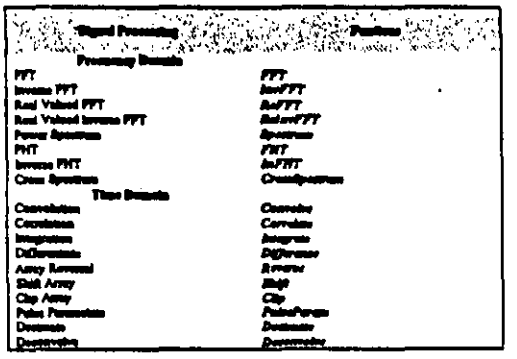


Array Operations	Function
1D Operations	
1D Array Addition	*Add1D*
1D Array Subtraction	*Sub1D*
1D Array Multiplication	*Mul1D*
1D Array Division	*Div1D*
1D Absolute Value	*Abs1D*
1D Negative Value	*Neg1D*
1D Linear Evaluation	*LinEval1D*
1D Maximum & Minimum	*MaxMin1D*
1D Linear Evaluation	*LinEval1D*
1D Polynomial Evaluation	*PolyEval1D*
1D Scaling	*Scale1D*
1D Quick Scaling	*QuickScale1D*
1D Sum of Elements	*Sum1D*
1D Product of Elements	*Prod1D*
1D Array Subset	*Subset1D*
2D Operations	
2D Array Addition	*Add2D*
2D Array Subtraction	*Sub2D*
2D Array Multiplication	*Mul2D*
2D Array Division	*Div2D*
2D Linear Evaluation	*LinEval2D*
2D Polynomial Evaluation	*PolyEval2D*
2D Scaling	*Scale2D*
2D Quick Scaling	*QuickScale2D*
2D Maximum & Minimum	*MaxMin2D*
2D Sum of Elements	*Sum2D*

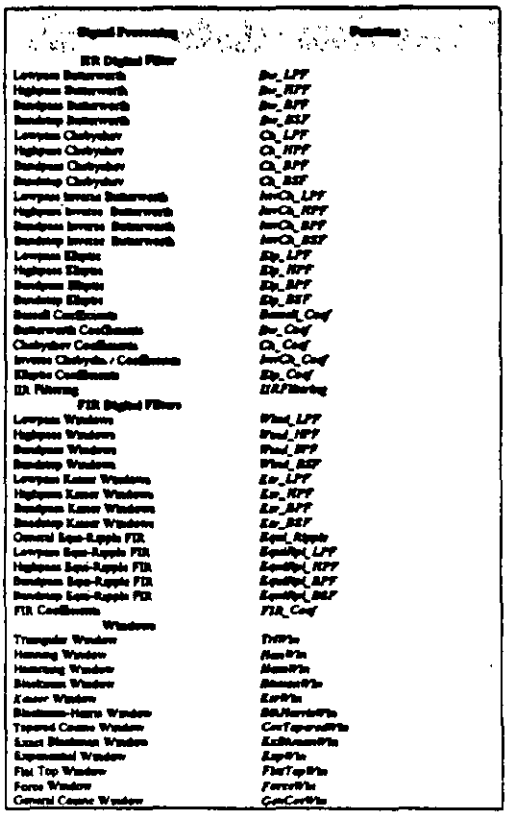


Complex Operations	Function
Complex Numbers	
Complex Addition	*CpxAdd*
Complex Subtraction	*CpxSub*
Complex Multiplication	*CpxMul*
Complex Division	*CpxDiv*
Complex Reciprocal	*CpxRecip*
Complex Square Root	*CpxSqrt*
Complex Logarithm	*CpxLog*
Complex Natural Log	*CpxLn*
Complex Power	*CpxPow*
Complex Exponential	*CpxExp*
Rectangular to Polar	*ToPolar*
Polar to Rectangular	*ToRect*
1D Complex Operations	
1D Complex Addition	*CpxAdd1D*
1D Complex Subtraction	*CpxSub1D*
1D Complex Multiplication	*CpxMul1D*
1D Complex Division	*CpxDiv1D*
1D Complex Linear Evaluation	*CpxLinEval1D*
1D Rectangular to Polar	*ToPolar1D*
1D Polar to Rectangular	*ToRect1D*

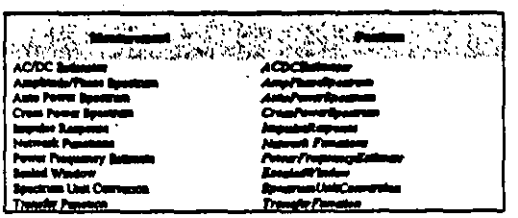
Array Utilities	Function
Clear Array	*Clear1D*
Set Array	*Set1D*
Copy Array	*Copy1D*
Unwrap Phase	*UnWrap1D*
Get Analysis Error	*GetAnalysisError*
Get Error String	*GetAnalysisErrorString*



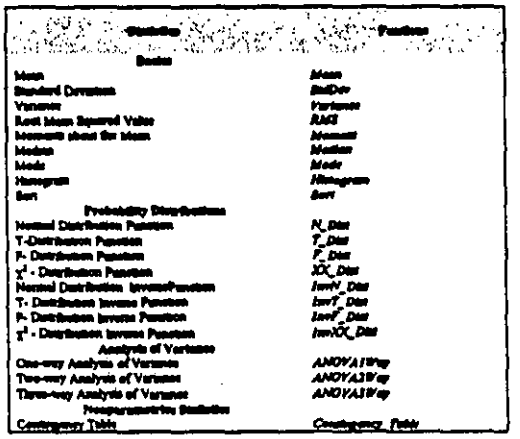
Signal Processing	Positivo
Frequency Domain	
FFT	*FFT*
Inverse FFT	*InvFFT*
Real Valued FFT	*RelFFT*
Real Valued Inverse FFT	*RelInvFFT*
Power Spectrum	*Spectrum*
FHT	*FHT*
Inverse FHT	*InFHT*
Cross Spectrum	*CrossSpectrum*
Time Domain	
Convolution	*Convolve*
Correlation	*Correlate*
Integration	*Integrate*
Differentiate	*Difference*
Array Reversal	*Reverse*
Shift Array	*Shift*
Clip Array	*Clip*
Pulse Parameters	*PulseParam*
Decimate	*Decimate*
Downsample	*Downsample*



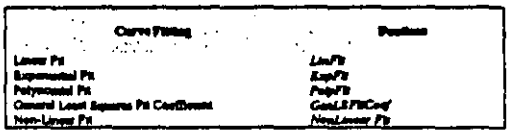
Signal Processing	Positivo
IIR Digital Filter	
Lowpass Butterworth	*Bw_LPF*
Highpass Butterworth	*Bw_HPF*
Bandpass Butterworth	*Bw_BPF*
Bandstop Butterworth	*Bw_BSF*
Lowpass Chebyshev	*Cb_LPF*
Highpass Chebyshev	*Cb_HPF*
Bandpass Chebyshev	*Cb_BPF*
Bandstop Chebyshev	*Cb_BSF*
Lowpass Inverse Butterworth	*InvCb_LPF*
Highpass Inverse Butterworth	*InvCb_HPF*
Bandpass Inverse Butterworth	*InvCb_BPF*
Bandstop Inverse Butterworth	*InvCb_BSF*
Lowpass Elliptic	*Ely_LPF*
Highpass Elliptic	*Ely_HPF*
Bandpass Elliptic	*Ely_BPF*
Bandstop Elliptic	*Ely_BSF*
Bessel Coefficients	*Bessel_Coef*
Butterworth Coefficients	*Bw_Coef*
Chebyshev Coefficients	*Cb_Coef*
Inverse Chebyshe. / Coefficients	*InvCb_Coef*
Elliptic Coefficients	*Ely_Coef*
IIR Filtering	*IIRFiltering*
FIR Digital Filter	
Lowpass Windowed	*Wind_LPF*
Highpass Windowed	*Wind_HPF*
Bandpass Windowed	*Wind_BPF*
Bandstop Windowed	*Wind_BSF*
Lowpass Kaiser Windowed	*Kai_LPF*
Highpass Kaiser Windowed	*Kai_HPF*
Bandpass Kaiser Windowed	*Kai_BPF*
Bandstop Kaiser Windowed	*Kai_BSF*
General Equi-Ripple FIR	*Equi_Ripple*
Lowpass Equi-Ripple FIR	*EquiRpl_LPF*
Highpass Equi-Ripple FIR	*EquiRpl_HPF*
Bandpass Equi-Ripple FIR	*EquiRpl_BPF*
Bandstop Equi-Ripple FIR	*EquiRpl_BSF*
FIR Coefficients	*FIR_Coef*
Windows	
Triangular Window	*TriWin*
Hamming Window	*HamWin*
Hamming Window	*HannWin*
Blackman Window	*BlackmanWin*
Kaiser Window	*KaiWin*
Blackman-Harris Window	*BlkHarrisWin*
Tapered Cosine Window	*CosTaperedWin*
Least Blackman Window	*LeBlackmanWin*
Exponential Window	*ExpWin*
Flat Top Window	*FlatTopWin*
Force Window	*ForceWin*
General Cosine Window	*GenCosWin*



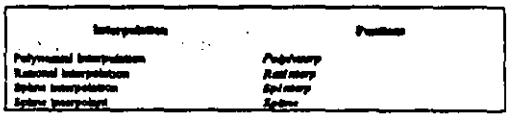
Measurement	Function
AC/DC Estimator	*ACDCEstimator*
Amplitude/Phase Spectrum	*AmpPhaseSpectrum*
Auto Power Spectrum	*AutoPowerSpectrum*
Cross Power Spectrum	*CrossPowerSpectrum*
Impulse Response	*ImpulseResponse*
Network Functions	*Network Functions*
Power Frequency Estimator	*PowerFrequencyEstimator*
Scaled Window	*ExtendWindow*
Spectrum Unit Conversion	*SpectrumUnitConversion*
Transfer Function	*TransferFunction*



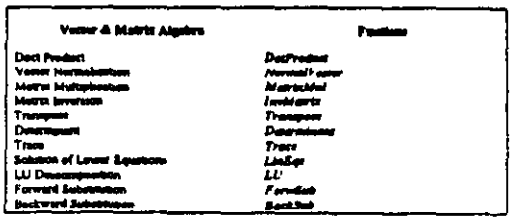
Statistics	Function
Basics	
Mean	*Mean*
Standard Deviation	*StdDev*
Variance	*Variance*
Root Mean Squared Value	*RMS*
Moments about the Mean	*Moment*
Median	*Median*
Mode	*Mode*
Histogram	*Histogram*
Sort	*Sort*
Probability Distributions	
Normal Distribution Function	*N_Dist*
T-Distribution Function	*T_Dist*
F- Distribution Function	*F_Dist*
χ^2 - Distribution Function	*XX_Dist*
Normal Distribution Inverse Function	*InvN_Dist*
T- Distribution Inverse Function	*InvT_Dist*
F- Distribution Inverse Function	*InvF_Dist*
χ^2 - Distribution Inverse Function	*InvXX_Dist*
Analysis of Variance	
One-way Analysis of Variance	*ANOVA1Way*
Two-way Analysis of Variance	*ANOVA2Way*
Three-way Analysis of Variance	*ANOVA3Way*
Nonparametric Statistics	
Contingency Table	*Contingency_Table*



Curve Fitting	Function
Linear Fit	*LinFit*
Exponential Fit	*ExpFit*
Polynomial Fit	*PolyFit*
General Least Squares Fit Coefficient	*GenLSFitCoef*
Non-Linear Fit	*NonLinear Fit*



Interpolation	Function
Polynomial Interpolation	*PolyInterp*
Rational Interpolation	*RatInterp*
Spline Interpolation	*SplineInterp*
Spline Interpolant	*Spline*



Vector & Matrix Algebra	Function
Dot Product	*DotProduct*
Vector Normalization	*NormalVector*
Matrix Multiplication	*MatrixMul*
Matrix Inversion	*InvMatrix*
Transpose	*Transpose*
Determinant	*Determinant*
Trace	*Trace*
Solution of Linear Equations	*LinEqn*
LU Decomposition	*LU*
Forward Substitution	*ForwSub*
Backward Substitution	*BackSub*

```
/**** UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO****/
/**** FACULTAD DE INGENIERIA ****/
/**** PROGRAMA DEL POSICIONADOR BIDIMENSIONAL ****/


#include <ansi_c.h>
#include <utility.h>
#include <userint.h>
#include "tesis.h"

static int handle, panelHandle,handle_ayuda,handle_manual,handle_automatico,handle_configurar;

int numero_botones_panel_control=9,numero_botones_panel_ayuda=4;

int boton[6],x,y,x1,y1,i,j,an,al,ancho[6],alto[6],fondo_visible,estado_x,NumPasX;
int mod_paso[9],num_pasos_x,halfstep;
char *mensaje[6];
int ayuda_boton[5],an_ayuda,al_ayuda,ancho_ayuda[5],alto_ayuda[5],CajaAyuda[4],NumPasDerX,NumPasIzqX;
char *ayuda_mensaje[5];
float rata;
int  pos_der_x,pos_izq_x;
int ejex,ejey,punto_ini_x,punto_ini_y,punto_fin_x,punto_fin_y,distancia_x,distancia_y;
double xx1,xx2,yy1,yy2;
int w1,h1,w2,h2,j_x,j_y;
int bandera_x,bandera_y,pasos_x,pasos_y;
int halfstep_x,halfstep_y;
int dis_x,dis_y,posicionx,posiciony;
unsigned char limite;
int eje_x,eje_y;
int trayectoria, linea,superficie;
int configuracion,A,B,CH,CL;
int motor;
int pos_arri_y,NumPasArriY,NumPasY,NumPasAbaY,dimensionX,dimensionY;
INT pasos_x1,pasos_h1,pasos_x2,pasos_h2,posx,posy;
int pasos_x_escala,pasos_y_escala;

/*funciones panel control automatico */
void mov_derecho_x(void);
void mov_izquierda_x(void);
void mov_derecho_un_paso_x(void);
void mov_izquierda_un_paso_x(void);
void monitoreo(int j);
void apaga_monitoreo(void);
void resetea_monitoreo(void);
void movimiento_automatico(void);
void encontrar_origen(void);
void colocar_origen_real(void);
void mov_arriba_y(void);
void mov_abajo_y(void);
void resetea_posicionador_automatico(void);


void main ()
{

        panelHandle = LoadPanel (0, "tesis.uir", fondo);
        handle = LoadPanel (panelHandle, "tesis.uir", comando);
        handle_ayuda=LoadPanel (panelHandle, "tesis.uir", ayuda);
        handle_manual=LoadPanel (panelHandle, "tesis.uir", manual);
        handle_automatico=LoadPanel (panelHandle, "tesis.uir", automatico);
        handle_configurar=LoadPanel (panelHandle, "tesis.uir", configura);
        DisplayPanel (handle);
        HidePanel (handle_automatico);
        HidePanel (handle_ayuda);
        HidePanel (handle_manual);
        DisplayPanel (panelHandle);

        mod_paso[1]=1;
        mod_paso[2]=5;
        mod_paso[3]=4;
        mod_paso[4]=6;
        mod_paso[5]=2;
        mod_paso[6]=10;
        mod_paso[7]=8;
        mod_paso[8]=9;


        dis_x=0;dis_y=0;
        posicionx=0,posiciony=0;
        num_pasos_x=0;



        mensaje[1]="Configuracion I/O de Tarjeta";
        mensaje[2]="Control Manual Motor de Pasos";
        mensaje[3]="Control Automatico Motor de Pasos";
        mensaje[4]="Información del Instrumento Virtual";
        mensaje[5]="Salir del Sistema";

        ayuda_mensaje[1]="Alcances del Sistema";
        ayuda_mensaje[2]="Configuración I/O de Tarjeta";
        ayuda_mensaje[3]="Control Virtual Motores de Paso";
        ayuda_mensaje[4]="Salir de la Ayuda";

        boton[1]=comando_boton1;
```

```
        boton[2]=comando_boton2;
        boton[3]=comando_boton3;
        boton[4]=comando_boton4;
        boton[5]=comando_boton5;

        ayuda_boton[1]=ayuda_ayuda_boton1;
        ayuda_boton[2]=ayuda_ayuda_boton2;
        ayuda_boton[3]=ayuda_ayuda_boton3;
        ayuda_boton[4]=ayuda_ayuda_boton4;

        CajaAyuda[1]=ayuda_ayuda1;
        CajaAyuda[2]=ayuda_ayuda2;
        CajaAyuda[3]=ayuda_ayuda3;
        for (i=1;i<=3;i++) SetCtrlAttribute(handle_ayuda,CajaAyuda[i],ATTR_VISIBLE,0);


        GetPanelAttribute(handle,ATTR_WIDTH,&an);
        GetPanelAttribute(handle,ATTR_HEIGHT,&al);
        GetCtrlAttribute(handle_ayuda,ayuda_tapiz_ayuda,ATTR_WIDTH,&an_ayuda);
        GetCtrlAttribute(handle_ayuda,ayuda_tapiz_ayuda,ATTR_HEIGHT,&al_ayuda);

        for(i=1;i<=numero_botones_panel_control;i++)
        {GetCtrlAttribute(handle,boton[i],ATTR_WIDTH,&ancho[i]);
         GetCtrlAttribute(handle,boton[i],ATTR_HEIGHT,&alto[i]);}
        for(i=1;i<=numero_botones_panel_ayuda;i++)
        {GetCtrlAttribute(handle_ayuda,ayuda_boton[i],ATTR_WIDTH,&ancho_ayuda[i]);
         GetCtrlAttribute(handle_ayuda,ayuda_boton[i],ATTR_HEIGHT,&alto_ayuda[i]);}


        SetPanelAttribute(handle,ATTR_TOP,80);
        SetPanelAttribute(handle,ATTR_LEFT,20);

        SetCtrlAttribute(handle_manual,manual_abajo,ATTR_VISIBLE,0);
        SetCtrlAttribute(handle_manual,manual_arriba,ATTR_VISIBLE,0);

        punto_ini_x=0; punto_ini_y=0;x1=0;h1=0;
        dimensionX=1800; dimensionY=1000;

        RunUserInterface ();
}


/* FUNCIONES PARA BOTONES DEL PANEL PRINCIPAL */

int mouse_sobre_panel_control (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_TIMER_TICK:

                        for(i=1;i<=numero_botones_panel_control;i++)
                        {
                        GetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,&fondo_visible);
                        if(fondo_visible==VAL_MINIMIZE) break;
                        else
                                GetRelativeMouseState(handle,0,&x,&y,0,0,0);
                                GetRelativeMouseState(handle,boton[i],&x1,&y1,0,0,0);
                                        if ((x>=0 && y>=0) && (x<=an && y<=al)){

        SetPanelAttribute(handle,ATTR_MOUSE_CURSOR,VAL_POINTING_FINGER_CURSOR);
                                                                        if ((x1>=0 && y1>=0) && (x1<=ancho[i] &&
y1<=alto[i]))

        SetCtrlVal(handle,comando_mensaje,mensaje[i]);}
                                                        else{
SetPanelAttribute(handle,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                                                                SetCtrlVal(handle,comando_mensaje,"");}
                        }


                        break;
        }
        return 0;
}


int configurar_tarjeta (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        SetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,VAL_MAXIMIZE);
                        DisplayPanel (handle_configurar);
                        SetPanelAttribute(handle_configurar,ATTR_MOUSE_CURSOR,VAL_POINTING_FINGER_CURSOR);
                        break;
        }
        return 0;
}


int control_manual (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
```

```
        switch (event) {
                case EVENT_COMMIT:
                        SetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,VAL_MAXIMIZE);
                        DisplayPanel (handle_manual);
                        /*SetCtrlAttribute(handle,comando_TimerPrincipal,ATTR_ENABLED,0);*/

                        break;
        }
        return 0;
}

int control_automatico (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        SetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,VAL_MAXIMIZE);
                        DisplayPanel (handle_automatico);
                        SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,1);
                        break;
        }
        return 0;
}


int informacion_general (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        SetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,VAL_MAXIMIZE);
                        DisplayPanel (handle_ayuda);
                        SetCtrlAttribute(handle_ayuda,ayuda_TimerAyuda,ATTR_ENABLED,1);
                        break;
        }
        return 0;
}

int salir (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        QuitUserInterface(0);
                        break;
        }
        return 0;
}

int salir_configurar (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        HidePanel (handle_configurar);

                        SetPanelAttribute(handle_configurar,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                        break;
        }
        return 0;
}

int salir_control_manual (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        HidePanel (handle_manual);
                        SetCtrlAttribute(handle,comando_TimerPrincipal,ATTR_ENABLED,1);
                        break;
        }
        return 0;
}

int salir_control_automatico (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        HidePanel (handle_automatico);
                        SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,0);
                        break;

        }
        return 0;
}

int salir_informacion_ayuda (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        for (i=1;i<=3;i++) SetCtrlAttribute(handle_ayuda,CajaAyuda[i],ATTR_VISIBLE,0);
                        HidePanel(handle_ayuda);
                        SetCtrlAttribute(handle_ayuda,ayuda_TimerAyuda,ATTR_ENABLED,0);
                        break;
```

```
        }
        return 0;
}


/* FUNCIONES PARA BOTONES DE PANEL AYUDA */

int mouse_sobre_panel_ayuda (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_TIMER_TICK:
                        for(i=1;i<=numero_botones_panel_ayuda;i++)
                        {
                        GetPanelAttribute(panelHandle,ATTR_WINDOW_ZOOM,&fondo_visible);
                        if(fondo_visible==VAL_MINIMIZED) break;
                        else
                                GetRelativeMouseState(handle_ayuda,ayuda_tapiz_ayuda,&x,&y,0,0,0);
                                GetRelativeMouseState(handle_ayuda,ayuda_boton[i],&x1,&y1,0,0,0);
                                        if ((x>=0 && y>=0) && (x<=an_ayuda && y<=al_ayuda)){

        SetPanelAttribute(handle_ayuda,ATTR_MOUSE_CURSOR,VAL_POINTING_FINGER_CURSOR);
                                                                        if ((x1>=0 && y1>=0) && (x1<=ancho_ayuda[i] &&
y1<=alto_ayuda[i])){

        SetCtrlVal(handle_ayuda,ayuda_mensaje_ayuda,ayuda_mensaje[i]);}
                                                                        else{
SetPanelAttribute(handle_ayuda,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                                                                SetCtrlVal(handle_ayuda,ayuda_mensaje_ayuda,"");}
                        }

                        break;
        }
        return 0;
}

int boton_alcances (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        for (i=1;i<=3;i++) SetCtrlAttribute(handle_ayuda,CajaAyuda[i],ATTR_VISIBLE,0);
                        SetCtrlAttribute(handle_ayuda,CajaAyuda[1],ATTR_VISIBLE,1);
                        break;
        }
        return 0;
}

int boton_configuracion (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        for (i=1;i<=3;i++) SetCtrlAttribute(handle_ayuda,CajaAyuda[i],ATTR_VISIBLE,0);
                        SetCtrlAttribute(handle_ayuda,CajaAyuda[2],ATTR_VISIBLE,1);
                        break;
        }
        return 0;
}

int boton_control (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        for (i=1;i<=3;i++) SetCtrlAttribute(handle_ayuda,CajaAyuda[i],ATTR_VISIBLE,0);
                        SetCtrlAttribute(handle_ayuda,CajaAyuda[3],ATTR_VISIBLE,1);
                        break;
        }
        return 0;
}


/* FUNCIONES PARA EL PANEL CONTROL MANUAL MOTOR DE PASOS */

int reseteo_manual (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        outp(0x380,170);
                        SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
                        SetCtrlVal(handle_manual,manual_CtrlManual5,0);
                        SetCtrlVal(handle_manual,manual_CtrlManual6,8);
                        SetCtrlVal(handle_manual,manual_CtrlManual7,0);
                        SetCtrlVal(handle_manual,manual_CtrlManual8,0);
                        NumPasX=0;NumPasY=0;
                        apaga_monitoreo();
                        encontrar_Origen();
                        Colocar_origen_real();
                        SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                        break;
        }
        return 0;
```

```
int seleccionar_motor (int panel, int control, int event,
                       void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        GetCtrlVal(handle_manual,manual_RING,&motor);
                        if (motor==0) {SetCtrlAttribute(handle_manual,manual_abajo,ATTR_VISIBLE,1);

SetCtrlAttribute(handle_manual,manual_arriba,ATTR_VISIBLE,1);

SetCtrlAttribute(handle_manual,manual_CtrlManual3,ATTR_VISIBLE,0);

SetCtrlAttribute(handle_manual,manual_CtrlManual4,ATTR_VISIBLE,0);
                                                                SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasY);
}
                        else            {SetCtrlAttribute(handle_manual,manual_abajo,ATTR_VISIBLE,0);

SetCtrlAttribute(handle_manual,manual_arriba,ATTR_VISIBLE,0);

SetCtrlAttribute(handle_manual,manual_CtrlManual3,ATTR_VISIBLE,1);

SetCtrlAttribute(handle_manual,manual_CtrlManual4,ATTR_VISIBLE,1);
                                                                SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasX);}

                        break;
        }
        return 0;
}

int apagar_eje_x (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        outp(0x300,0);
                        apago_monitores();
                        break;
        }
        return 0;
}

int secuencia_derecha_x (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_derecha_x();
                        break;
        }
        return 0;
}

int secuencia_izquierda_x (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_izquierda_x();
                        break;
        }
        return 0;
}

int secuencia_abajo_y (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_abajo_y();
                        break;
        }
        return 0;
}

int secuencia_arriba_y (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_arriba_y();
                        break;
        }
        return 0;
}


int reta_eje_x (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
```

```
                        break;
        }
        return 0;
}

int secuencia_izquierda_un_paso_x (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}

int secuencia_derecha_un_paso_x (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}


int reset_eje_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}


int apagar_eje_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}

int secuencia_derecha_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}

int secuencia_izquierda_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}


int rate_eje_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;


int secuencia_derecha_un_paso_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)

        switch (event) {
                case EVENT_COMMIT:

                        break;
```

```
        }
        return 0;
}

int secuencia_izquierda_un_paso_y (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        break;
        }
        return 0;
}

void mov_abajo_y(void)
{halfstep_x=0;
 if (j_y==0) j_y=0;if (j_y==9) j_y=1;

 pos_der_x=0;
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
 GetCtrlAttribute(handle_manual,manual_CtrlManual8,ATTR_CTRL_VAL,&NumPasAbaY);

 for (i=1;i<=NumPasAbaY;i=i+1)

 {j_y=j_y+1;halfstep_y=mod_paso[j_y];
 pos_der_x++;


 NumPasY=NumPasY-1;

 if (NumPasY<0) {NumPasY=0;break;}
 SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasY);
 SetCtrlVal (handle_manual,manual_CtrlManual8,j_y);


 outp(0x300,halfstep_x+(halfstep_y<<4));
 monitoreo(j_y);

 if (j_y==8) j_y=0;}
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);

 outp(0x300,0);
}

void mov_arriba_y(void)
{halfstep_x=0;
 if (j_y==-1) j_y=9;if (j_y==0) j_y=0;

 pos_arri_y=0;
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
 GetCtrlAttribute(handle_manual,manual_CtrlManual8,ATTR_CTRL_VAL,&NumPasArriY);

 for (i=1;i<=NumPasArriY;i=i+1)
 {j_y=j_y-1;halfstep_y=mod_paso[j_y];
 pos_arri_y--;


 NumPasY=NumPasY+1;

 if (NumPasY<0) {NumPasY=0;GetCtrlVal(handle_manual,manual_CtrlManual8,&j_y);break;}
 SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasY);
 SetCtrlVal (handle_manual,manual_CtrlManual8,j_y);


outp(0x300,halfstep_x+(halfstep_y<<4));
 monitoreo(j_y);
 if (j_y==-1) j_y=9;}
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
 outp(0x300,0);

}

void mov_derecha_x(void)

{halfstep_y=0;
 if (j_x==0) j_x=0;if (j_x==9) j_x=1;

 pos_der_x=0;
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
 GetCtrlAttribute(handle_manual,manual_CtrlManual6,ATTR_CTRL_VAL,&NumPasDerX);

 for (i=1;i<=NumPasDerX;i=i+1)

 {j_x=j_x+1;halfstep_x=mod_paso[j_x];
 pos_der_x++;


 NumPasX=NumPasX+1;

 if (NumPasX<0) {NumPasX=0;break;}
 SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasX);
 SetCtrlVal (handle_manual,manual_CtrlManual8,j_x);
```

```
outp(0x300,halfstep_x+(halfstep_y<<4));
monitoreo(j_x);

if (j_x==8) j_x=0;}
SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);

outp(0x300,0);

}

void mov_izquierda_x(void)
{halfstep_y=0;
 if (j_x==-1) j_x=7;if (j_x==0) j_x=8;

 pos_izq_x=0;
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
 GetCtrlAttribute(handle_manual,manual_CtrlManual8,ATTR_CTRL_VAL,&NumPasIzqX);

 for (i=1;i<=NumPasIzqX;i=i+1)
 {j_x=j_x-1;halfstep_x=mod_paso[j_x];
 pos_izq_x--;

NumPasI=NumPasX-1;

 if (NumPasX<0) {NumPasX=0;GetCtrlVal(handle_manual,manual_CtrlManual8,&j_x);break;}
 SetCtrlVal (handle_manual,manual_CtrlManual7,NumPasX);
 SetCtrlVal (handle_manual,manual_CtrlManual8,j_x);


outp(0x300,halfstep_x+(halfstep_y<<4));
 monitoreo(j_x);
 if (j_x==-1) j_x=8;}
 SetPanelAttribute(handle_manual,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
 outp(0x300,0);

 }

void mov_derecha_un_paso_x(void)
{if (j_x==8) j_x=0;if (j_x==0) j_x=1;
 halfstep_y=0;
 j_x++;halfstep_x=mod_paso[j_x];
 outp(0x300,halfstep_x+(halfstep_y<<4));
 if (j_x==8) j_x=0;}

void mov_izquierda_un_paso_x(void)
{if (j_x==-1) j_x=7;if (j_x==0) j_x=8;
 halfstep_y=0;
 j_x--;halfstep_x=mod_paso[j_x];
 outp(0x300,halfstep_x+(halfstep_y<<4));
 if (j_x==-1) j_x=7;}


 void monitoreo(int j)
 {switch (j) {
 case 1:    SetCtrlVal(handle_manual,manual_AX,0);
                              SetCtrlVal(handle_manual,manual_BX,0);
                              SetCtrlVal(handle_manual,manual_CX,0);
                              SetCtrlVal(handle_manual,manual_DX,1);
                              break;
 case 2:  SetCtrlVal(handle_manual,manual_AX,0);
                              SetCtrlVal(handle_manual,manual_BX,1);
                              SetCtrlVal(handle_manual,manual_CX,0);
                              SetCtrlVal(handle_manual,manual_DX,1);
                              break;
 case 3:    SetCtrlVal(handle_manual,manual_AX,0);
                              SetCtrlVal(handle_manual,manual_BX,1);
                              SetCtrlVal(handle_manual,manual_CX,0);
                              SetCtrlVal(handle_manual,manual_DX,0);
                              break;
 case 4:    SetCtrlVal(handle_manual,manual_AX,0);
                              SetCtrlVal(handle_manual,manual_BX,1);
                              SetCtrlVal(handle_manual,manual_CX,1);
                              SetCtrlVal(handle_manual,manual_DX,0);
                              break;
 case 5:    SetCtrlVal(handle_manual,manual_AX,0);
                              SetCtrlVal(handle_manual,manual_BX,0);
                              SetCtrlVal(handle_manual,manual_CX,1);
                              SetCtrlVal(handle_manual,manual_DX,0);
                              break;
 case 6:    SetCtrlVal(handle_manual,manual_AX,1);
                              SetCtrlVal(handle_manual,manual_BX,0);
                              SetCtrlVal(handle_manual,manual_CX,1);
                              SetCtrlVal(handle_manual,manual_DX,0);
                              break;
 case 7:    SetCtrlVal(handle_manual,manual_AX,1);
                              SetCtrlVal(handle_manual,manual_BX,0);
                              SetCtrlVal(handle_manual,manual_CX,0);
                              SetCtrlVal(handle_manual,manual_DX,0);
                              break;
 case 8:    SetCtrlVal(handle_manual,manual_AX,1);
                              SetCtrlVal(handle_manual,manual_BX,0);
                              SetCtrlVal(handle_manual,manual_CX,0);
```

```
                        SetCtrlVal(handle_manual,manual_DE,1);
                        break;}
}

void apaga_monitoreo(void)
{
                        SetCtrlVal(handle_manual,manual_AI,0);
                        SetCtrlVal(handle_manual,manual_BI,0);
                        SetCtrlVal(handle_manual,manual_CE,0);
                        SetCtrlVal(handle_manual,manual_DE,0);
                        SetCtrlVal (handle_manual,manual_CtrlManual8,0);}
void resetea_monitoreo(void)
{
                        NumPasX=0;
                        j=0;
                        SetCtrlVal(handle_manual,manual_CtrlManual7,0);
                        SetCtrlVal(handle_manual,manual_AI,1);
                        SetCtrlVal(handle_manual,manual_BI,0);
                        SetCtrlVal(handle_manual,manual_CE,1);
                        SetCtrlVal(handle_manual,manual_DE,0);
                        SetCtrlVal (handle_manual,manual_CtrlManual8,0);}


/* PANEL DE CONTROL AUTOMATICO */

int Timer_Automatico (int panel, int control, int event,
                      void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_TIMER_TICK:
                        GetRelativeMouseState(handle_automatico,automatico_Grafico,&ejex,&ejey,0,0,0);
                        if ((ejex>=0 && ejex<=400) && (ejey>=0 && ejey<=200))
                                {SetCtrlVal(handle_automatico,automatico_slide_x,(dimensionX*ejex)/400);
                            SetCtrlVal(handle_automatico,automatico_slide_y,dimensionY -
((dimensionY*ejey)/200));
                            SetCtrlVal(handle_automatico,automatico_Xfin,ejex);
                            SetCtrlVal(handle_automatico,automatico_Yfin,200-ejey);
                            punto_fin_x=ejex; punto_fin_y=200-ejey;
                            distancia_x=abs(punto_fin_x - punto_ini_x); distancia_y=abs(punto_fin_y -
punto_ini_y);
                            SetCtrlVal(handle_automatico,automatico_AbsX,distancia_x);
                            SetCtrlVal(handle_automatico,automatico_AbsY,distancia_y);
                            GetCtrlVal(handle_automatico,automatico_posicionx,&pasos_x1);
                            GetCtrlVal(handle_automatico,automatico_posiciony,&pasos_h1);
                            GetCtrlVal(handle_automatico,automatico_slide_x,&pasos_x2);
                            GetCtrlVal(handle_automatico,automatico_slide_y,&pasos_h2);
                            GetCtrlVal(handle_automatico,automatico_linea,&linea);
                            GetCtrlVal(handle_automatico,automatico_Fondo,&superficie);
                                  SetCtrlAttribute (handle_automatico,automatico_Grafico, ATTR_PLOT_BGCOLOR,
superficie);
                                  GetCtrlVal(handle_automatico,automatico_trayectoria,&trayectoria);
                                }

                            else {SetCtrlVal(handle_automatico,automatico_slide_x,0);
                                   SetCtrlVal(handle_automatico,automatico_slide_y,0);
                                   SetCtrlVal(handle_automatico,automatico_Xfin,0);
                             SetCtrlVal(handle_automatico,automatico_Yfin,0);
                             SetCtrlVal(handle_automatico,automatico_AbsX,0);
                             SetCtrlVal(handle_automatico,automatico_AbsY,0);
                             GetCtrlVal(handle_automatico,automatico_Fondo,&superficie);
                             SetCtrlAttribute (handle_automatico,automatico_Grafico, ATTR_PLOT_BGCOLOR,
superficie);}


                            break;



        }
        return 0;
}

void resetea_posicionador_automatico(void)
{
        /* DESACTIVA OTROS CONTROLES */
        SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,0);
        SetPanelAttribute(handle_automatico,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);
        SetCtrlVal(handle_automatico,automatico_escalaX,0);
        SetCtrlVal(handle_automatico,automatico_escalaY,0);
        SetCtrlVal(handle_automatico,automatico_Xini,0);
        SetCtrlVal(handle_automatico,automatico_Yini,0);
        SetCtrlVal(handle_automatico,automatico_Xfin,0);
        SetCtrlVal(handle_automatico,automatico_Yfin,0);
        SetCtrlVal(handle_automatico,automatico_posicionx,0);
        SetCtrlVal(handle_automatico,automatico_posiciony,0);
        pasos_x1=0;pasos_h1=0;punto_ini_x=0;punto_ini_y=0;posx=0;posy=0;


                        /* RUTINA PARA ENCONTRAR ORIGEN */
                                bandera_x=1; bandera_y=1;
                                outp(0x300,170);
                                outp(0x300,0);j_x=0;j_y=0;
                                dimensionX=0; dimensionY=0;
```

```
                while (bandera_x || bandera_y)
                {
                        /* Checar limites Inico de Carrera */
                        limite = inp(0x300);
                        if (limite==1 || limite==5) {bandera_x=0;halfstep_x=0;}
                        if (limite==4 || limite==5) {bandera_y=0;halfstep_y=0;}

                        /* Movimiento eje X hacia la Izquierda */
                        if (bandera_x==1)
                        {
                                if (j_x==-1) j_x=9;if (j_x==0) j_x=0;
                                j_x--;halfstep_x=med_paso[j_x];
                                SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                        }

                        /** Movimiento eje Y hacia Abajo **/
                        if (bandera_y==1)
                        {
                                if (j_y==0) j_y=0;if (j_y==9) j_y=1;
                                j_y++;halfstep_y=med_paso[j_y];
                                SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                        }

                        /* Señal de salida por el Puerto A */
                        outp(0x300,halfstep_x+(halfstep_y<<4));
                        Delay(.008);

                }
        /* RUTINA PARA ENCONTRAR DIMENSIONES DEL POSICIONADOR */
                bandera_x=1; bandera_y=1;

                while (bandera_x || bandera_y)
                {
                        /* Checar limites Fin de Carrera */
                        limite = inp(0x301);
                        if (limite==2 || limite==10) {bandera_x=0;halfstep_x=0;}
                        if (limite==8 || limite==10) {bandera_y=0;halfstep_y=0;}

                        /* Movimiento eje X hacia la Derecha */
                        if (bandera_x==1)
                        {
                                if (j_x==-1) j_x=0;if (j_x==9) j_x=1;
                                j_x++;halfstep_x=med_paso[j_x];dimensionX++;
                                SetCtrlVal(handle_automatico,automatico_estadox,j_x);
SetCtrlVal(handle_automatico,automatico_escalaX,dimensionX);

                        }

                        /* Movimiento eje Y hacia Arriba */
                        if (bandera_y==1)
                        {
                                if (j_y==1) j_y=9;if (j_y==0) j_y=0;
                                j_y--;halfstep_y=med_paso[j_y];dimensionY++;
                                SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
SetCtrlVal(handle_automatico,automatico_escaleY,dimensionY);
                        }

                        /* Señal de salida Puerto A */
                        outp(0x300,halfstep_x+(halfstep_y<<4));
                        Delay(0.008);
                }
        /* RUTINA PARA ENCONTRAR ORIGEN  NUEVAMENTE */
                bandera_x=1; bandera_y=1;

                while (bandera_x || bandera_y)
                {
                        /* Checar limites Inicio de Carrera */
                        limite = inp(0x301);
                        if (limite==1 || limite==5) {bandera_x=0;halfstep_x=0;}
                        if (limite==4 || limite==5) {bandera_y=0;halfstep_y=0;}

                        /* Movimiento eje X hacia la Izquierda */
                        if (bandera_x==1)
                        {
                                if (j_x==-1) j_x=9;if (j_x==0) j_x=0;
                                j_x--;halfstep_x=med_paso[j_x];
                                SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                        }

                        /* Movimiento eje Y hacia Abajo */
                        if (bandera_y==1)
                        {
                                if (j_y==0) j_y=0;if (j_y==9) j_y=9;
                                j_y++;halfstep_y=med_paso[j_y];
                                SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                        }

                        /* Señal de salida Puerto A */
                        outp(0x300,halfstep_x+(halfstep_y<<4));
```

```
                                        Delay(.000);
                                }

                        /* RUTINA PARA COLOCAR ORIGEN REAL */
                        for (i=1; i<=10;i=i+1)
                        {
                                if (j_x==0) j_x=0;if (j_x==9) j_x=1;
                                        j_x++;halfstep_x=mod_paso[j_x];
                                        SetCtrlVal(handle_automatico,automatico_entradax,j_x);
                                if (j_y==1) j_y=9;if (j_y==0) j_y=0;
                                        j_y--;halfstep_y=mod_paso[j_y];
                                        SetCtrlVal(handle_automatico,automatico_entraday,j_y);

                                /* Señal de salida Puerto A */
                                outp(0x300,halfstep_x+(halfstep_y<<4));
                                Delay(0.01);
                        }
                        outp(0x300,0);

                        /* INICIALIZACION DE VARIABLES */
                        posicionx=0;posiciony=0;
                        punto_ini_x=0;punto_ini_y=0;
                        SetCtrlVal(handle_automatico,automatico_Xini,0);
                        SetCtrlVal(handle_automatico,automatico_Yini,0);

        /* ACTIVA OTROS CONTROLES */
                        /* SetCtrlAttribute(handle_automatico,automatico_boton1_automatico,ATTR_DIMMED,1); */
                        SetPanelAttribute(handle_automatico,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                        SetCtrlAttribute(handle_automatico,automatico_Grafico,ATTR_DIMMED,0);
                        DeleteGraphPlot(handle_automatico,automatico_Grafico,-1,VAL_IMMEDIATE_DRAW);

                        SetCtrlAttribute (handle_automatico,automatico_slide_x, ATTR_MAX_VALUE,dimensionX );
                        SetCtrlAttribute (handle_automatico,automatico_slide_y, ATTR_MAX_VALUE,dimensionY );


                        SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,1);
}

int Dibuje_Trayectoria (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)

{
        switch (event) {
                case EVENT_COMMIT:
                SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,0);
                SetPanelAttribute(handle_automatico,ATTR_MOUSE_CURSOR,VAL_HOUR_GLASS_CURSOR);

                GetCtrlVal(handle_automatico,automatico_Xini,&w1);
                GetCtrlVal(handle_automatico,automatico_Yini,&h1);

                SetCtrlVal(handle_automatico,automatico_Xini,ejex);
                SetCtrlVal(handle_automatico,automatico_Yini,200-ejey);


                GetCtrlVal(handle_automatico,automatico_Xini,&w2);
                GetCtrlVal(handle_automatico,automatico_Yini,&h2);


pasos_x_escala=(abs(w2-w1));pasos_y_escala=(abs(h2-h1));
pasos_x=(abs(pasos_x2-pasos_x1));pasos_y=(abs(pasos_h2-pasos_h1));
bandera_x=1;bandera_y=1;
if(pasos_x==0) bandera_x=0;if (pasos_y==0) bandera_y=0;

if (w2>w1) posicionx=posicionx+pasos_x_escala;
else posicionx=posicionx-pasos_x_escala;
if (h2>h1) posiciony=posiciony+pasos_y_escala;
else posiciony=posiciony-pasos_y_escala;

SetCtrlVal(handle_automatico,automatico_borrarx,posicionx);
SetCtrlVal(handle_automatico,automatico_borrary,posiciony);

while (bandera_x || bandera_y)
        {


                if (pasos_x>0)
                {

                        if (w2>w1)                      /** Movimiento a la Derecha **/
                                {
                                if (j_x==0) j_x=0;if (j_x==9) j_x=1;
                                j_x++;halfstep_x=mod_paso[j_x];dis_x++;posx++; /*posicionx++;*/

                                }
                        else if(w2<w1)                  /** Movimiento a la Izquierda **/
                                {
                                if (j_x==1) j_x=9;if (j_x==0) j_x=0;
                                j_x--;halfstep_x=mod_paso[j_x];dis_x--;posx--; /*posicionx--;*/

                                }

                        pasos_x=pasos_x-1;
                        if (pasos_x==0) {bandera_x=0;halfstep_x=0;punto_ini_x=posx;punto_ini_y=posy;}
                }
```

```
                if (pasos_y>0)
                {

                        if (h2>h1)                      /** Movimiento hacia Arriba **/
                                {
                                if (j_y==1) j_y=0;if (j_y==0) j_y=0;
                                j_y--;halfstep_y=mod_paso[j_y];dis_y--;posy++;/*posiciony--;*/
                                }
                        else if(h2<h1)                  /** Movimiento hacia Abajo **/
                                {
                                if (j_y==0) j_y=0;if (j_y==0) j_y=1;
                                j_y++;halfstep_y=mod_paso[j_y];dis_y++;posy--;/*posiciony++;*/

                                }

                        pasos_y=pasos_y-1;
                        if (pasos_y==0) {bandera_y=0;halfstep_y=0;punto_ini_x=posx;punto_ini_y=posy;}
                }

                SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                SetCtrlVal(handle_automatico,automatico_posicionx,posx);
                SetCtrlVal(handle_automatico,automatico_posiciony,posy);

                outp(0x300,halfstep_x+(halfstep_y<<4));


                if (trayectoria==0)

        PlotLine(handle_automatico,automatico_Grafico,(punto_ini_x*800)/dimensionX,(punto_ini_y*280)/dimensionY,
                        (posx*800)/dimensionX,(posy*280)/dimensionY,VAL_WHITE);
                Delay(0.000);
        }           /** final de while **/

                outp(0x300,0);

                PlotLine(handle_automatico,automatico_Grafico,w1,h1,w2,h2,linea);




                punto_ini_x=posx;punto_ini_y=posy;
                SetPanelAttribute(handle_automatico,ATTR_MOUSE_CURSOR,VAL_DEFAULT_CURSOR);
                SetCtrlAttribute(handle_automatico,automatico_TimerAutomatico,ATTR_ENABLED,1);
                break;

        }
        return 0;
}


int reseteo_automatico (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                reseteo_posicionador_automatico();
                }
        return 0;
}


int limpiar_automatico (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        DeleteGraphPlot (handle_automatico, automatico_Grafico,-1,VAL_IMMEDIATE_DRAW);

                        break;
        }
        return 0;
}


void encontrar_origen(void)
{
                /** RUTINA PARA ENCONTRAR ORIGEN **/
                        bandera_x=1; bandera_y=1;
                        outp(0x300,170);
                        outp(0x300,0);j_x=0;j_y=0;
                        dimensionX=0; dimensionY=0;

                        while (bandera_x || bandera_y)
                        {
                                /** Checar limites de ejes **/
                                limite = inp(0x301);
                                SetCtrlVal(handle_automatico,automatico_borrar,limite);

                                if (limite==1 || limite==5) {bandera_x=0;halfstep_x=0;}
                                if (limite==4 || limite==5) {bandera_y=0;halfstep_y=0;}
                                /** Movimiento eje X hacia la izquierda **/
                                if (bandera_x==1)
```

```
                                        {
                                        if (j_x==1) j_x=0;if (j_x==0) j_x=0;
                                        j_x--;halfstep_x=mod_paso[j_x];
                                        SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                                        }
                                /** Movimiento eje Y hacia Abajo **/
                                if (bandera_y==1)
                                        {
                                        if (j_y==0) j_y=0;if (j_y==9) j_y=1;
                                        j_y++;halfstep_y=mod_paso[j_y];
                                        SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                                        }
                                outp(0x300,halfstep_x+(halfstep_y<<4));
                                Delay(.008);

                        }

                /** RUTINA PARA ENCONTRAAR DIMENSIONES DEL POSICIONADOR **/
                                bandera_x=1; bandera_y=1;

                        while (bandera_x || bandera_y)
                        {
                                /** Checar limites de ejes **/
                                limite = inp(0x301);
                                if (limite==2 || limite==10) {bandera_x=0;halfstep_x=0;}
                                if (limite==8 || limite==21) {bandera_y=0;halfstep_y=0;}
                                /** Movimiento eje X hacia la Derecha **/
                                if (bandera_x==1)
                                        {
                                        if (j_x==0) j_x=0;if (j_x==9) j_x=1;
                                        j_x++;halfstep_x=mod_paso[j_x];dimensionX++;
                                        SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                                        SetCtrlVal(handle_automatico,automatico_escalaX,dimensionX);
                                        }
                                /** Movimiento eje Y hacia Arriba **/
                                if (bandera_y==1)
                                        {
                                        if (j_y==1) j_y=9;if (j_y==0) j_y=0;
                                        j_y--;halfstep_y=mod_paso[j_y];dimensionY++;
                                        SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                                        SetCtrlVal(handle_automatico,automatico_escalaY,dimensionY);
                                        }
                                outp(0x300,halfstep_x+(halfstep_y<<4));
                                Delay(0.008);

                        }

                /** RUTINA PARA ENCONTRAAR ORIGEN **/
                                bandera_x=1; bandera_y=1;


                        while (bandera_x || bandera_y)
                        {
                                /** Checar limites de ejes **/
                                limite = inp(0x301);
                                SetCtrlVal(handle_automatico,automatico_borrar,limite);

                                if (limite==1 || limite==5) {bandera_x=0;halfstep_x=0;}
                                if (limite==4 || limite==5) {bandera_y=0;halfstep_y=0;}
                                /** Movimiento eje X hacia la Izquierda **/
                                if (bandera_x==1)
                                        {
                                        if (j_x==1) j_x=9;if (j_x==0) j_x=0;
                                        j_x--;halfstep_x=mod_paso[j_x];
                                        SetCtrlVal(handle_automatico,automatico_estadox,j_x);
                                        }
                                /** Movimiento eje Y hacia Abajo **/
                                if (bandera_y==1)
                                        {
                                        if (j_y==0) j_y=0;if (j_y==9) j_y=1;
                                        j_y++;halfstep_y=mod_paso[j_y];
                                        SetCtrlVal(handle_automatico,automatico_estadoy,j_y);
                                        }
                                outp(0x300,halfstep_x+(halfstep_y<<4));
                                Delay(.008);

                        }



}


void colocar_origen_real(void)

                for (i=1; i<=10;i=i+1)
                {
                        if (j_x==0) j_x=0;if (j_x==9) j_x=1;
                        j_x++;halfstep_x=mod_paso[j_x];
                        /**SetCtrlVal(handle_automatico,automatico_estadox,j_x);**/
                        if (j_y==1) j_y=9;if (j_y==0) j_y=0;
                        j_y--;halfstep_y=mod_paso[j_y];
                        /**SetCtrlVal(handle_automatico,automatico_estadoy,j_y);**/
                        outp(0x300,halfstep_x+(halfstep_y<<4));
                        Delay(0.01);
```

```
                }
                outp(0x300,0);
}

int Auto_Derecha_Un_Paso (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_derecha_un_paso_a();
                        break;
        }
        return 0;
}

int Auto_Izquierda_Un_Paso (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:
                        mov_izquierda_un_paso_a();
                        break;
        }
        return 0;
}

int configuracion_tarjeta (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event) {
                case EVENT_COMMIT:

                        GetCtrlVal(handle_configurar,configurar_puertoA,&A);
                        GetCtrlVal(handle_configurar,configurar_puertoB,&B);
                        GetCtrlVal(handle_configurar,configurar_puertoCH,&CH);
                        GetCtrlVal(handle_configurar,configurar_puertoCL,&CL);
                        if (A==0 && CH==0 && B==0 && CL==0) configuracion=128;
                        if (A==0 && CH==0 && B==0 && CL==1) configuracion=129;
                        if (A==0 && CH==0 && B==1 && CL==0) configuracion=130;
                        if (A==0 && CH==0 && B==1 && CL==1) configuracion=131;
                        if (A==0 && CH==1 && B==0 && CL==0) configuracion=136;
                        if (A==0 && CH==1 && B==0 && CL==1) configuracion=137;
                        if (A==0 && CH==1 && B==1 && CL==0) configuracion=138;
                        if (A==0 && CH==1 && B==1 && CL==1) configuracion=139;
                        if (A==1 && CH==0 && B==0 && CL==0) configuracion=144;
                        if (A==1 && CH==0 && B==0 && CL==1) configuracion=145;
                        if (A==1 && CH==0 && B==1 && CL==0) configuracion=146;
                        if (A==1 && CH==0 && B==1 && CL==1) configuracion=147;
                        if (A==1 && CH==1 && B==0 && CL==0) configuracion=152;
                        if (A==1 && CH==1 && B==0 && CL==1) configuracion=153;
                        if (A==1 && CH==1 && B==1 && CL==0) configuracion=154;
                        if (A==1 && CH==1 && B==1 && CL==1) configuracion=155;

                        outp(0x303,configuracion);
                        outp(0x300,0);

                        break;
        }
        return 0;
}
```