9

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

PROGRAMACION DE LA INTERFACE DE USUARIO ENTRE UNA MAQUINA DE MEDICION POR COORDENADAS Y UNA PC.

T E S I S

QUE PARA OBTENER EL GRADO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

GLICONSTENER BERMUDEZ SANTANA

DIRECTOR DE TESIS: M.I. JOSE SANCHEZ VIZCAINO

CODIRECTOR DE TESIS: M.I. BENJAMIN VALERA OROZCO

SINODALES: M.I. ADOLFO MILLAN NAJERA
M.I. HUMBERTO GOMEZ NARANJO
M.I. SERGIO QUINTANA THIERRY

MEXICO, D.F. MARZO DE 2001

A mis padres por todo el apoyo y comprensión que me han brindado, por haberme enseñado el camino del trabajo y del esfuerzo que me han llevado a la meta que hoy tengo ante mí.

A mi abuelita, por todo su apoyo, cariño y comprensión, por mostrarme siempre el respeto hacia las demás personas y por ser ejemplo para mí.

A mis hermanos Clara, Guadalupe, Beatriz, Omar y José, por las etapas de la vida que hemos compartido y por las diferencias que hemos tenido.

A todos mis tíos, por el cariño y el apoyo que me han brindado.

A mis compañeros y amigos Erika, Huicho, Rodolfo, Ricardo y a todos mis compañeros de primer semestre, por las horas de diversión y de estudio que hemos compartido.

A mi codirector de tesis y amigo Benjamín Valera, por su confianza, apoyo y paciencia.

Al Ingeniero José Sánchez, por su ayuda y por diseñar la MMC. Al jefe del departamento de ingeniería y desarrollo tecnológico, Ingeniero Gerardo Ruiz. Al coordinador del laboratorio de metrología Ingeniero Rigoberto Nava.

Indice

INTRODUCCIÓN

Objetivo

Automatizar la captura y procesamiento de lecturas mediante una PC en una máquina de medición por coordenadas.

Definición del problema

Entorno actual

Recientemente en el Centro de Instrumentos UNAM, particularmente en la Sección de Metrología, se ha desarrollado y construido un prototipo mecánico de una Máquina de Medición por Coordenadas, MMC. El uso extendido de tales instrumentos de medición dimensional ha cobrado un gran auge como parte de los procesos industriales de control de calidad. El amplio mercado para las MMC ha motivado el desarrollo de una versión económica que ofrezca ventajas adicionales sobre los productos disponibles en el mercado. Nuestra idea no es competir en complejidad

con las MMCs comerciales, sino rescatar las necesidades industriales en un diseño realista de MMC para los consumidores nacionales. En éste sentido, estamos en la posición de ofrecer un producto competitivo en exactitud y software específico que no abrume al operador con sistemas y software altamente complejos.

La figura 1 muestra el estado actual de la MMC

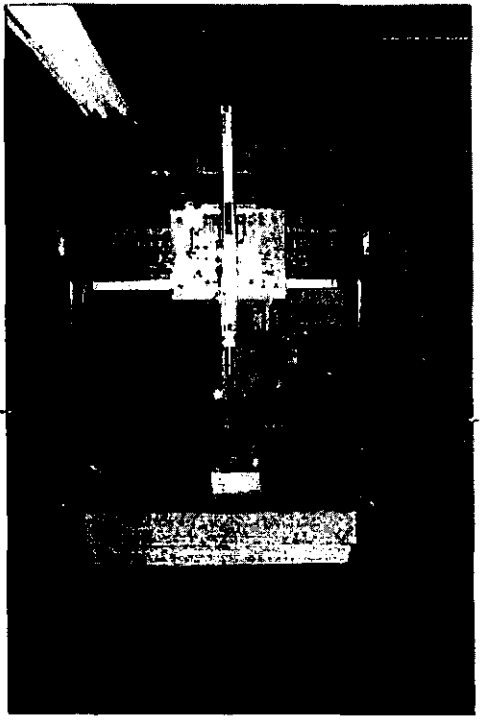


Figura 1. MMC del Centro de Instrumentos UNAM.

Descripción del problema a resolver

Planteamos el problema de automatizar la captura, procesamiento y administración de lecturas provenientes de la MMC construida en el CI UNAM que hasta el momento su operación es completamente manual. Los aspectos de interés en la automatización son:

- Instalación de equipo comercial para la interface entre la MMC y una computadora personal.
- Captura mediante computadora de posiciones x, y y z.
- Creación de un software específico para la captura de lecturas y medición de geometrías típicas.

Relevancia

El trabajo planteado en esta tesis ofrece características similares a las propias de los equipos comerciales. No obstante la relevancia y justificaciones del proyecto son las siguientes:

1. Experimentar con modelado de instrumentos para observar los efectos de deformación en mediciones de alta exactitud.
2. Resulta una opción económica, viable de ser transferida a la industria nacional.
3. Cumple con los objetivos del CI UNAM en cuanto a desarrollo de instrumentación.
4. Se podrá prestar el servicio de esta herramienta de medición a la industria.

Que contemplan labores académicas tanto en el nivel de investigación como en el de desarrollo tecnológico.

Relación con otras áreas

Existe una gran relación con las áreas de instrumentación electrónica, instrumentación virtual, metrología dimensional, control de calidad, diseño mecánico, y programación de computadoras.

Metodo

Los métodos involucrados en este proceso de automatización son los siguientes:

1. Programación de computadoras en ambiente Windows para el diseño de la interface con el usuario.
2. Diseño electrónico para la interconexión de transductores ópticos a computadora PC.
3. Graficación por computadora para la ayuda visual y analítica en la medición de geometrías típicas.

El esquema planteado se puede apreciar en la figura 2.

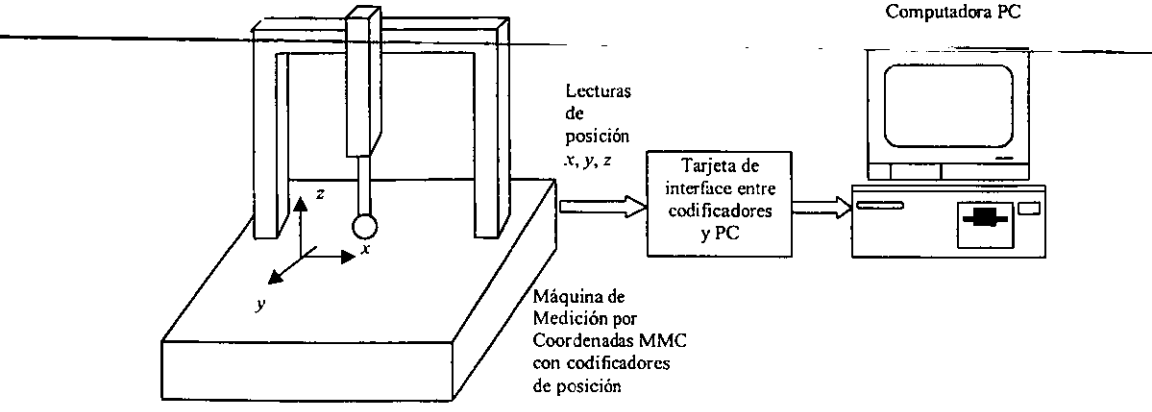


Figura 2. Automatización de una MMC.

En el esquema de la figura 1, una computadora captura las lecturas de posición x, y, z provenientes de tres transductores ópticos de posición solidarios a la estructura de la MMC. Se puede apreciar una tarjeta de interface entre los transductores y la PC. Las tareas a realizar entonces consisten en lo siguiente:

- Interconectar electrónicamente la MMC con la PC.

- Programar la interface con el usuario para administrar lecturas.
- Programar algoritmos específicos para medición de geometrías típicas.
- Realizar la programación anterior en un ambiente visual para Windows.

Resultados esperados

Automatizar la adquisición de lecturas de posición x, y, z con resolución de 2 μm en un alcance de medición de 500 mm^3.

Implementar algoritmos para la interpretación de geometrías básicas en una interfaz amigable para el usuario.

Resumen de la tesis

La presente tesis consta de cinco capítulos divididos de la siguiente forma: (1) Metrología dimensional, (2) Introducción a la programación, (3) Automatización del proceso de adquisición de lecturas, (4) Ambiente de operación para la máquina de medición por coordenadas, (5) Resultados y conclusiones.

En el capítulo uno se verán algunos conceptos básicos de metrología dimensional que fueron fundamentales en el desarrollo de esta tesis. Se da una descripción de los transductores de posición más comunes, con la finalidad de comprender el funcionamiento de los transductores ópticos empleados en la MMC. Este capítulo también hace referencia a las interfaces más usuales entre los transductores y las PCs.

El capítulo dos hace referencia a las bases de la programación en lenguaje Visual C++, describe también la organización de la PC y el modo de programación general de los periféricos más comunes en las computadoras personales.

El capítulo tres contiene la parte sustancial del desarrollo de la tesis. En este se describe el esquema a seguir para el desarrollo de nuestro sistema de medición, así como los componentes de hardware y software mínimos

que lo conforman, haciendo referencia a las interfaces empleadas entre la MMC y la PC, así como entre la PC y el usuario.

Durante el capítulo cuatro se hará referencia al ambiente de operación de la MMC; es decir, se dará una descripción del software desarrollado para la adquisición e interpretación de las lecturas provenientes de los codificadores ópticos incrementales que están asociados a la MMC. En este capítulo se proporciona también un manual de operación para el usuario.

Finalmente, el capítulo cinco resalta los resultados obtenidos en nuestro sistema de medición y la evaluación del mismo, en el que se pueden apreciar las mediciones hechas sobre diferentes cuerpos geométricos. Este capítulo contempla también el trabajo que se pretende desarrollar a futuro a partir del sistema desarrollado.

CAPITULO 1

Metrología dimensional

A través del tiempo, y prácticamente desde la historia de la humanidad, el hombre ha tenido la necesidad de contabilizar y dimensionar cualquier objeto, animal o incluso a sí mismo. Debido a esta necesidad, el hombre ha desarrollado una gran cantidad de herramientas para poder lograr su objetivo, dando origen así a la *metrología*, que al igual que toda ciencia, evoluciona y se enriquece día a día con el desarrollo de técnicas de medición más exactas, ayudándose de la tecnología.

En este capítulo se pretende dar un panorama general de la metrología mostrando las causas que dieron origen al nacimiento de la misma remontándonos un poco en la historia. Pretendemos mostrar las bases en las que se sustenta y dar a conocer las actividades básicas que esta ciencia desarrolla, así como sus aplicaciones en la industria. Sin embargo, para poder entender la metrología es necesario conocer algunos conceptos que son utilizados frecuentemente en este campo, y que fueron esenciales para el desarrollo de ésta tesis. Al final del capítulo se exponen algunos conceptos de transductores comunes en la metrología dimensional e interfaces para PC's.

1.1. Introducción general

Todos nosotros utilizamos y dependemos de la calibración todos los días en los aspectos mas comunes de nuestra vida. Desde ajustar nuestro reloj con la hora de la radio, hasta revisar el estado del tiempo. Para el éxito, todo depende de la propia calibración y de la adecuación a los estándares nacionales.

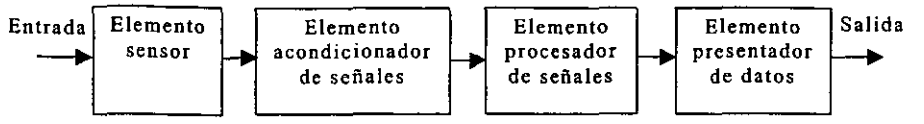
Debido a los efectos del tiempo, a los cambios que experimentan los equipos en temperatura o al estrés mecánico, el desempeño o rendimiento de los mismos se deteriora gradualmente. Esto es llamado *tendencia*. Cuando esto pasa, los resultados de nuestras pruebas llegan a ser no fiables y tanto el diseño y la producción se ven afectados. Aunque la tendencia no puede ser eliminada, ésta puede ser detectada y corregida a través de procesos de calibración.

La calibración es simplemente la comparación del desempeño o rendimiento de instrumentos con un estándar o medida conocida. Esto puede involucrar simplemente la determinación de la desviación del valor nominal o incluir la corrección (ajuste) para minimizar los errores. Los equipos propiamente calibrados proveen la confianza de que sus productos o servicios conocen sus especificaciones. La calibración incrementa el rendimiento de la producción, optimiza los recursos, asegura la consistencia y garantiza que las mediciones sean compatibles con las hechas en cualquier lugar.

Medir, es evaluar una cantidad con respecto a otra de la misma especie tomada como unidad y como término de comparación; en otras palabras, medir una cantidad de cierta magnitud es compararla con otra cantidad que se adopta como referencia, a la cual se le denomina *unidad*. Desde los primitivos sistemas de medición, existía la tendencia hacia la definición de tales unidades; pero fue con el desarrollo de los métodos de fabricación producidos a partir de la Revolución Industrial cuando la búsqueda de estas medidas se hizo esencial y con el reciente avance de la ciencia y la tecnología en los siglos XIX y XX, se estimuló el desarrollo de la metrología.

Sistemas de medición

Un sistema de medición consta básicamente de cuatro elementos, tal como se muestra en la figura 1.1.



F igura 1.1. Estructura general de un sistema de medición.

El elemento sensor está en contacto con el proceso y genera una salida. El elemento acondicionador de señales toma la salida del elemento sensor y la convierte en una forma mas adecuada para un procesamiento adicional. El elemento procesador de señales toma la salida del elemento acondicionador y la convierte en una forma más adecuada para la presentación. Por último, el elemento procesador de datos presenta el valor medido en una forma que el observador puede reconocer fácilmente [4].

Historia de la metrología

Si revisamos un poco la historia de la metrología, nos damos cuenta de que en las antiguas civilizaciones las unidades de longitud tenían origen antropomorfo, tales como el *paso, el pie, codo, palmo, pulgada, dedo*, entre otros; los cuales se empleaban en la construcción de edificios, templos y otras obras. Mientras que la medida del tiempo estaba basada en el estudio de los astros. Las unidades de medición que se empleaban eran diferentes, dependiendo de cada civilización, por ejemplo, en Mesopotamia, las unidades principales eran el *codo* (utilizada para medir longitudes), el *qa* (volumen) y la *mina* (peso); donde el codo equivalía aproximadamente a 50 cm, el qa era equivalente a 840 cm^3 y la mina aproximadamente a 5 N.

Por otra parte, en las culturas griega y romana, aunque no tuvieron un gran desarrollo en éste ámbito, la metrología de esos tiempos aportó a la sociedad los parámetros necesarios para regular actividades cotidianas tales como la agricultura, religión, comercio, estadísticas, impuestos, tiempo y localización geográfica.

Evolución de la fabricación

Con el advenimiento de la revolución industrial y la producción en serie, llegó la necesidad de que las piezas fueran intercambiables, ya que

cualquier maquina o mecanismo es formado a partir de otras piezas y era necesario que las respectivas piezas se acoplaran entre sí. Hoy en día, la intercambiabilidad es indispensable, ya que nos permite por ejemplo adquirir un repuesto para cierto modelo de un automóvil con un distribuidor o con otro de manera indiferente.

El origen de intercambiabilidad proviene tal vez de la fabricación de armas de fuego, ya que era necesaria la adaptación de la munición, aparte de que periódicamente resultaba necesario sustituir alguna pieza dañada, y en ese tiempo la única forma de acoplar dos piezas era disponiendo de una de estas para poder fabricar la otra, ya que de hacerlo de otra manera normalmente ocasionaba una holgura entre ellas. Debido a esto, se introdujo aproximadamente en el siglo XV el concepto de *calibres*, los cuales materializaban las dimensiones máxima del eje y mínima del taladro en el caso de ensamblajes con fuego, con lo que se logró reducir el juego u holgura entre las piezas, sin la necesidad de tenerlas presentes, llegando a ser la base de lo que ahora conocemos como *producción en serie*.

Las exigencias de piezas cada vez mas precisas en la fabricación mecánica, la navegación y la astronomía fueron determinantes para el desarrollo de instrumentos de medida de longitud, tiempo y masa a partir del siglo XVI. Así, en 1525 se desarrollan por primera vez en Francia los calibres de artillería. Comienzan así a surgir los instrumentos de medición con alto grado de precisión tales como el micrómetro que permite hoy en día medir longitudes de 1 µm, apareciendo entonces de metrología de precisión [10].

El sistema Internacional de unidades (SI)

En 1960 se establece el Sistema Internacional de unidades (SI), el cual introduce varios tipos de unidades, a saber:

- Unidades básicas o fundamentales: metro, kilogramo, segundo, ampere, candela, kelvin y mole.
- Unidades derivadas, son establecidas a partir de las unidades fundamentales.
- Unidades suplementarias, éstas son las unidades del ángulo plano (radián) y ángulo sólido (estereoradián), las cuales el Comité Internacional de Pesas y Medidas (CIMP) ha precisado que deben considerarse como magnitudes derivadas sin precisión.

Aunque todas las unidades derivadas pueden expresarse en función de las fundamentales, algunas tienen interés especial tales como *newton, pascal, tesla, volt, etc.* Los símbolos de las unidades deben escribirse de acuerdo con las siguientes reglas [10]:

- Siempre en caracteres rectos y con minúsculas (por ejemplo mm), excepto cuando el nombre de la unidad procede del nombre o apellido de una persona, en cuyo caso la primer letra es mayúscula (N, Pa, Hz).
- Nunca en plural (se debe escribir 12 kg y no 12 kgs).
- Nunca con punto final (es decir, 12 kg y no 12 kg.).
- Se puede agregar un punto central para indicar el producto, pero debe cuidarse el orden de los símbolos para evitar otras interpretaciones, así puede escribirse Nm ó N·m pero no mN que se interpretaría como milinewton (10-3 N).
- El cociente se puede notar con una barra (m/s^2) pero se prefiere $\frac{m}{s^2}$ o aún mejor $m \cdot s^{-2}$.
- No deben emplearse dos o mas barras en un mismo símbolo, es decir, nunca se debe escribir m/s/s sino $m \cdot s^{-2}$.

A continuación se muestran las definiciones actuales (ya que a través del tiempo se han modificado) de las unidades básicas de SI [10]:

metro (m): Longitud recorrida por la luz en el vacío durante 1/299792458 s.

kilogramo (kg): Masa del prototipo internacional del kilogramo conservado en el Buró internacional de Pesas y Medidas (BIPM).

segundo (s): Duración de 9192631770 periodos de la radiación correspondiente a la transición entre los dos niveles hiperfinos del estado fundamental del átomo cesio 133.

ampere (A): Intensidad de una corriente eléctrica constante que mantenida en dos conductores paralelos rectilíneos, de longitud infinita, de sección circular despreciable y colocados en el vacío a una distancia de un metro uno del otro, produce sobre estos dos conductores una fuerza igual a 2×10^{-7} newton por metro de longitud.

kelvin (k): Fracción 1/273.16 de la temperatura del punto triple del agua.

mole (mol): Cantidad de sustancia o de materia de un sistema que contiene tantas entidades elementales como átomos hay en 0.012 kg de carbono 12

candela (cd): Intensidad luminosa, en una dirección dada, de una fuente que emite una radiación monocromática de frecuencia 540 X 10^{12} Hz y cuya intensidad energética en esta dirección es de 1/683 W/sr.

1.2. Metrología dimensional

La metrología es la ciencia de las mediciones. La manufactura moderna requiere de metrología avanzada para resolver complejos problemas. La metrología ha llegado a ser una ciencia más precisa debido al alto grado de calidad requerido por las compañías manufactureras y por los consumidores.

El empleo en la manufactura de estándares de referencia primarios certificados y utilizados de acuerdo a procedimientos establecidos, asegura que las partes manufacturadas cuentan con las condiciones de calidad que las manufactureras fijan para proveer productos a sus consumidores con las especificaciones que ellos buscan. El uso de estos estándares de referencia y de instrumentos precisos de metrología son un efectivo camino para asegurar la adherencia a las especificaciones y el proveer la calidad que los consumidores requieren.

Un metrologista "Desarrolla y evalúa los sistemas de calibración que miden las características de objetos o fenómenos, tales como longitud, masa, tiempo, temperatura, corriente eléctrica, intensidad luminosa y unidades derivadas de medidas físicas o químicas. Identifica la magnitud de los orígenes de error que contribuyen a la incertidumbre de resultados para determinar la fiabilidad de los procesos de medición en términos cuantitativos. Rediseña o ajusta la capacidad de medición para minimizar errores. Desarrolla métodos de calibración y técnicas basadas en principios de ciencias de medición, análisis técnico de problemas de medición y requerimientos de precisión. Dirige la ingeniería, calidad y personal de laboratorio en diseño, manufactura, evaluación y calibración de estándares de medición, instrumentos y sistemas de prueba para asegurar la selección de instrumentación aprobada. Aconseja a otros sobre métodos de resolución de problemas de medición e intercambia información con demás personal

metrologista a través de la participación en el gobierno y comités de estandarización y sociedades profesionales."1

Algunos de los conceptos que se manejan con mucha frecuencia en el campo de la metrología son los siguientes.

PATRÓN Patrón es la pieza o aparato representativo de una magnitud física de peso, longitud, capacidad, temperatura, presión atmosférica, inductancia, luz, etcétera, conocida con exactitud [11]. Los 'patrones son herramientas para la calibración de precisión, que tienen dimensiones preestablecidas. Se utiliza para calibrar herramientas de precisión o para efectuar mediciones muy precisas. El material con el que se fabrican estos patrones es acero inoxidable, carburo de tungsteno o acero para herramienta estabilizado y endurecido.

TRAZABILIDAD Este concepto tan utilizado en calibración hace referencia a que el equipo de medición debe ser probado contra un estándar de alta precisión. Por supuesto, esta calibración debe ser hecha de forma planificada, y con fundamentos periódicos con evidencia de los resultados de la comparación. Esta grabación debe incluir identificación de los estándares específicamente utilizados (los cuales deben estar dentro de su intervalo de calibración asignado) y algunos medios del conocimiento del método empleado así como otras condiciones de prueba. Examinando estas grabaciones, debe ser posible demostrar una cadena irrompible de comparaciones que terminan en el departamento responsable del mantenimiento y desarrollo de los estándares de medición de un país. Esta demostrable relación de los estándares nacionales, con exactitudes conocidas, representa la trazabilidad [15].

PERIODICIDAD Esta se refiere al intervalo de calibración asignado a una pieza del equipo —por ejemplo tres meses o dos años—. Una forma alternativa de expresar esto es el ciclo de calibración, usualmente el número de calibraciones que son requeridas por año. El equipo utilizado en cualquier situación metrológica debe tener una exactitud conocida; que es, una especificación asignada por el fabricante o por el usuario. Debido a que el desempeño de todo sobre la tierra se degrada con el tiempo o uso, la precisión esperada debe ser relativa a un periodo de tiempo dado [15].

CARACTERIZACIÓN La calibración puede ser solo la determinación de atributos o características o incluir realineación a nominal (con ciertos límites). La caracterización es comúnmente asociada con parámetros que no

1 Definición tomada del Dictionary of Occupational Titles.

son especificados o por aquellos que, por el uso de datos de medición, permiten al usuario incrementar su propia precisión de medición. Tales datos deben estar acompañados con información de incertidumbre para ser evaluada por el usuario deseando la calificación de su propia prueba de precisión [15].

INCERTIDUMBRE A pesar de lo que el vendedor pueda decir, ninguna medida puede ser garantizada como perfecta. Una incertidumbre es una cifra del mérito asociado con el valor medido actualmente; los límites fronterizos dentro de los cuales el valor "verdadero" miente. Contribuidores para esta inexactitud son el desempeño del equipo utilizado para hacer la medición, el proceso de prueba o la técnica misma y los efectos ambientales. Imprecisiones adicionales pueden resultar en el comportamiento del fenómeno o pieza medida [15].

PRUEBA DE RELACIÓN DE EXACTITUD Es generalmente considerada una buena práctica utilizar equipo de prueba y técnicas cuya incertidumbre combinada es de 3 a 10 veces menor que la unidad bajo prueba, lo cual representa la prueba de relación de exactitud : [15]

$$TAR = \frac{Especificación}{Incertidumbre}$$; TAR: Test Accuracy Ratio (prueba de relación de exactitud).

MARGEN DE ERROR Este es particularmente empleado cuando la prueba de relación de exactitud es baja, este es un margen de seguridad con el propósito de acercarse mas a un límite aceptable cuando se prueba un producto específico. El límite del margen de error puede simplemente, ser fijado a un punto igual a la especificación menos la incertidumbre, pero es normalmente fijada de tal forma que proporcione la confianza que resultaría de utilizar una prueba de relación de exactitud (TAR) de 4:1 con un punto aceptable fijado al límite de la especificación [15].

1.3. Transductores de posición

Mediante sistemas electrónicos adecuados es posible medir distancias de amplios márgenes, los cuales varían desde miles de kilómetros, para observaciones astronómicas, hasta fracciones de milímetros (para aplicaciones industriales), empleando distintas técnicas que dependen del margen de medida deseado.

Para las medidas en distancias cortas o desplazamientos, existe una gran cantidad de métodos, como el uso de reglas graduadas de lectura óptica o el empleo de transductores.

Transductor se define como todo dispositivo que convierte una señal de una forma física en una señal correspondiente pero de otra forma física; es decir, es un dispositivo que convierte un tipo de energía en otra.

1.3.1. Transductores ópticos

Uno de los tipos de transductores más utilizados es el denominado *codificador incremental*. Estos codificadores tienen un elemento lineal llamado también regla, o un disco de poca inercia que se desplaza solidario a la pieza que se desea medir. Este elemento posee dos tipos de zonas, arregladas de forma alternativa y equidistante (como se muestra en la figura 1.2), de tal forma que un incremento de posición es detectado mediante un cabezal de lectura fijo [9].

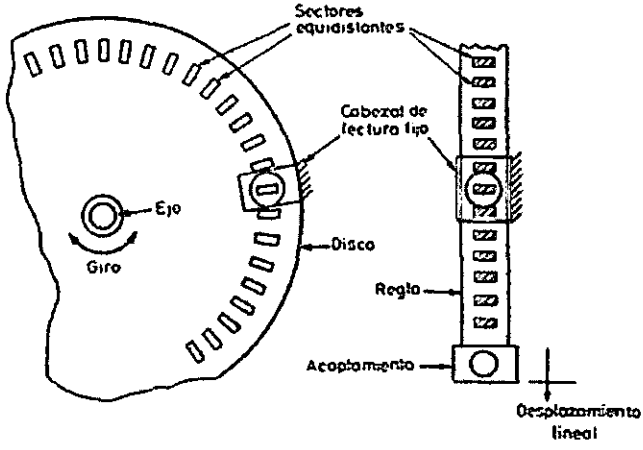


Figura 1.2. Codificador incremental.

El empleo de estos codificadores es muy económico, pero tiene algunos inconvenientes, como por ejemplo la pérdida de la posición cuando falla la alimentación del sistema, cuando se desconecta, o con la presencia de interferencias fuertes. Es necesario también un contador bidireccional, para poder tener una salida digital compatible con los elementos de entrada-

salida de un procesador, y por último no permite detectar el sentido de avance si no se dispone de más elementos a los mostrados en la figura 1.2.

Para poder determinar el sentido de avance es necesario agregar otro elemento de lectura y algunas veces otra pista codificada junto con algunos circuitos electrónicos.

Los transductores ópticos pueden consistir de sectores opacos y transparentes, de sectores reflectores y no reflectores, o de franjas de interferencia. Cualquiera que sea el caso, en el cabezal de lectura se encuentra una fuente de luz (normalmente un LED) y un fotodetector (célula fotoeléctrica o fototransistor), cuyos inconvenientes son normalmente el polvo y las vibraciones en el sistema de enfoque [3].

Cuando el transductor emplea sectores opacos y transparentes, el emisor y el detector se sitúan uno a cada lado del elemento móvil (figura 1.3a), mientras que cuando se emplean transductores con sectores reflectores y no reflectores, el emisor y el detector se sitúan en el mismo lado (figura 1.3b).

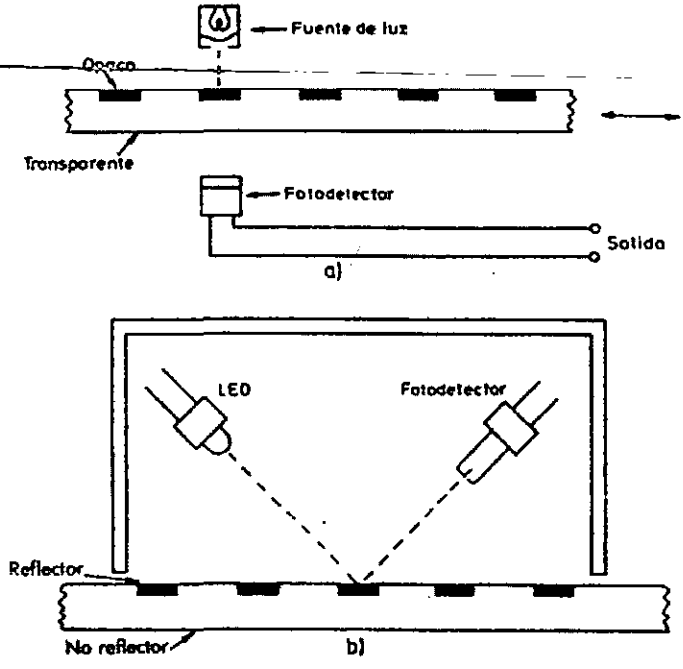


Figura 1.3. Arreglo de elementos en transductores ópticos.

Los codificadores de franjas de interferencia están basados en las figuras de Moiré, las cuales se producen mediante un movimiento lineal, empleando por ejemplo una regla fija y una móvil que tenga las franjas inclinadas con respecto a la otra (figura 1.4). Si la inclinación α es tal que $tan\ \alpha = p/d$, cuando hay un desplazamiento relativo de recorrido p (paso entre líneas), se produce un desplazamiento vertical d de una franja horizontal. Si la inclinación es n veces mayor, aparecen n franjas oscuras horizontales [9].

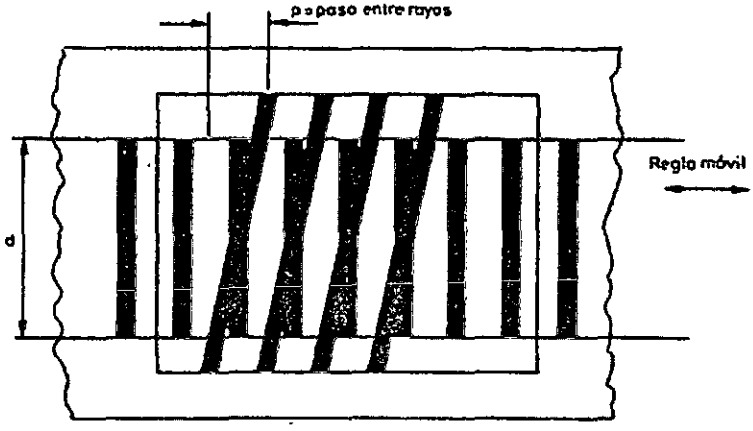


Figura 1.4. Codificadores de franjas de interferencia.

En la actualidad, los transductores que ofrecen una mayor resolución son los ópticos, ya que ésta depende del tamaño del fotodetector y es posible aumentar la resolución agregando una o más rejillas fijas entre el elemento móvil y el detector, las cuales deben tener zonas opacas y transparentes con el mismo paso (apertura) que elemento codificador (figura 1.5). Así el detector recibirá la mayor cantidad de luz cuando todas las rejillas y el elemento codificado móvil estén perfectamente alineados.

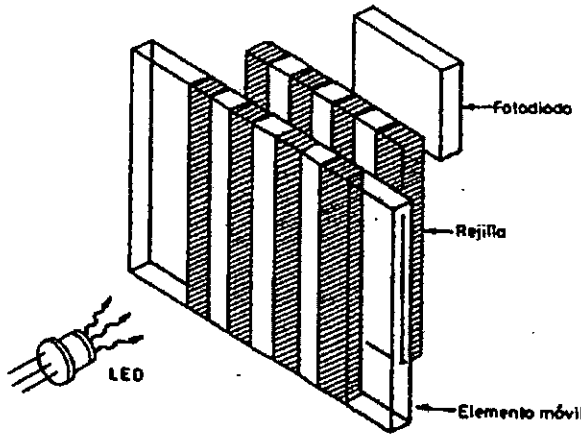


Figura 1.5. Codificador óptico.

1.3.2. Transductores inductivos

Las propiedades empleadas para la diferenciación de los sectores pueden ser ópticas, eléctricas o magnéticas, pero cualquiera que sea el método empleado, la salida básica es un tren de pulsos con un ciclo de trabajo del 50%.

Una regla o regla dentada de material ferromagnético dará un impulso de tensión cada vez que pase por enfrente de una bobina fija expuesta a un campo magnético constante (figura 1.6). La forma de señal obtenida es casi senoidal, pero esta se puede cuadrar o simplemente determinar sus pasos por cero. Evidentemente, existe una velocidad máxima y una velocidad mínima para poder aplicar este método [9].

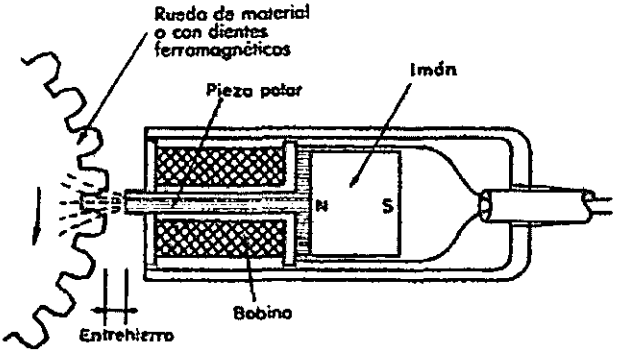


Figura 1.6. Regla dentada de material ferromagnético.

Otro método inductivo empleado es el que se aprecia en la figura 1.7, el cual está basado en un núcleo toroidal con dos bobinas. Una de éstas se emplea para la excitación, con corrientes desde 20 a 200 kHz de frecuencia, y otra para la detección. La bobina de detección tiene dos estados: '1' cuando se detecta tensión y '0' cuando se detecta una tensión de frecuencia igual a la excitación. El elemento móvil posee zonas con material magnetizado, así cuando éste queda delante de la cabeza de lectura la satura porque el flujo que emana del material se suma al creado por la señal de excitación; cuando el núcleo se encuentra saturado no se detecta tensión en la segunda bobina ($i = -d\phi/dt$, y ϕ es constante) [9]. Cuando delante de la cabeza de lectura hay una zona que no posee material magnetizado, la segunda bobina detecta una tensión inducida por la primera.

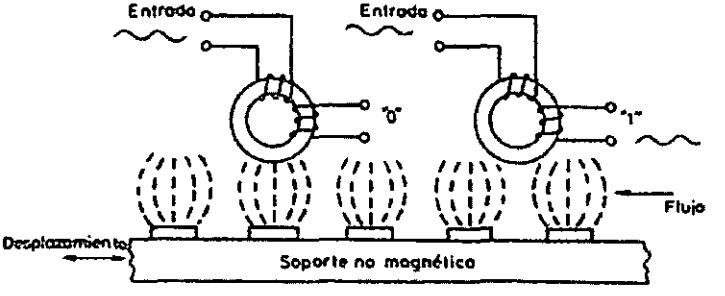


Figura 1.7. Arreglo toroidal para transductores inductivos.

En los transductores inductivos se emplea otra bobina sensora y un detector de fase para determinar el sentido del desplazamiento. Esta bobina dará una

señal desfasada 90° respecto a la anterior, y el detector de fase indicará el sentido de giro.

1.3.3. Transductores basados en interferometría láser

Un método óptico para medir dimensiones con mucha exactitud se basa en el principio de la interferencia de la luz. El instrumento basado en este principio se llama interferómetro y se usa en la calibración de los bloques calibradores y otras aplicaciones donde se requieren mediciones dimensionales absolutas con extrema precisión.

El principio de interferencia ofrece la posibilidad práctica de medir pequeños defectos de superficie y calibrar los bloques calibradores; sin embargo, se utiliza un plano óptico inclinado, como el de la figura 1.8.

La luz monocromática de la fuente se colima mediante la lente L en la placa divisora S_2, la cual es un espejo medio plateado que refleja la mitad de la luz hacia el espejo M ópticamente plano y permite la transmisión de la otra mitad hacia la pieza de trabajo W. Ambos haces se reflejan, se recombinan en la placa divisora S_2 y entonces se transmiten a la pantalla. Pueden aparecer franjas en la pantalla que resultan de las diferencias en las longitudes de las trayectorias ópticas de los dos haces.

Si el instrumento está construido de manera apropiada, estas diferencias ocurren por las variaciones dimensionales de la pieza de trabajo.

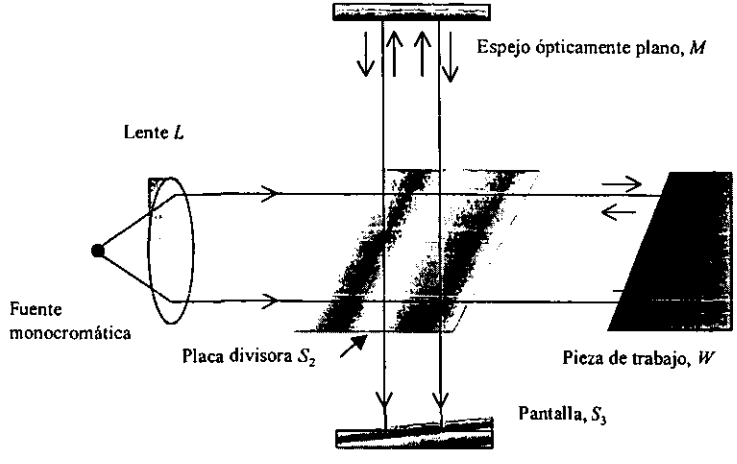


Figura 1.8. Esquema del interferómetro.

La longitud de onda de la luz de la fuente monocromática influye en el espaciamiento de las franjas. La tabla 1.1 que se muestra a continuación detalla las longitudes de onda de algunas fuentes comunes de luz y el intervalo correspondiente entre franjas de media longitud de onda.

Fuente	Longitud de onda μm	Intervalo de franja de media longitud de onda μm
Helio	0.589	0.295
Kriptón 86	0.606	0.303
Mercurio 198	0.546	0.0.273
Sodio	0.598	0.299

Tabla 1.1. Longitudes de onda.

1.4. Interfaces usuales entre transductores y PCs

Cuando es necesario registrar muchos datos, cuando esto debe hacerse durante mucho tiempo, o simplemente cuando se desea mucha fiabilidad y objetividad en las medidas, se recurre a la automatización de éstas, tanto si se realizan en el laboratorio como si se trata de controlar el proceso que genera las variables medidas. Los sistemas enfocados a la realización automática de medidas se denominan sistemas de adquisición de datos.

La situación más frecuente en el acondicionamiento de transductores consiste en convertir señales eléctricas de analógico a digital. Cuando se deben procesar varias señales, es muy común que compartan también el mismo recurso, que bien puede ser un canal de comunicación o un sistema de procesado o de control. Esta es una de las funciones de un sistema de adquisición de datos.

Una configuración simple de un sistema de adquisición de datos es el mostrado en la figura 1.9, la cual emplea lo que se denomina un sistema de multiplexado de bajo nivel. Consiste de un multiplexor analógico que asigna el convertidor analógico-digital sucesivamente a la señal de cada uno de los transductores de entrada. El multiplexor consta de una serie de

compuertas analógicas y un decodificador que va explorando o asignando tiempo de acuerdo a las ordenes del controlador al que se van a suministrar los datos de salida.

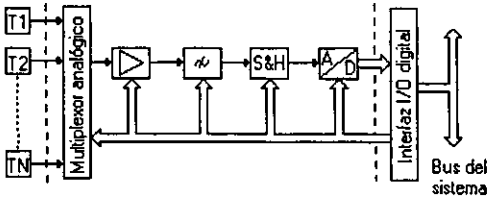


Figura 1.9. Sistema para la adquisición de datos.

El convertido A/D posee una entrada de impedancia limitada, un margen dinámico fijo, con entradas analógicas aceptadas entre 0 y 10 [V], y una velocidad máxima de conversión determinada. Esta velocidad debe permitir la exploración de cada canal con la frecuencia deseada. Estas circunstancias determinan la presencia de los tres bloques indicados entre el multiplexor y el convertidor: amplificador, filtro y circuito de muestreo y retención [9].

El amplificador es unipolar o diferencial, según sea unipolar o flotante la salida del transductor y, consecuentemente, el multiplexor sea unipolar o diferencial respectivamente.

El filtro debe impedir el paso de señales con frecuencia superior a la mitad de la frecuencia de muestreo, y en la práctica es recomendable que su frecuencia de corte sea del orden de diez veces inferior a dicha frecuencia.

El circuito de muestreo y retención es normalmente transparente para el usuario y puede considerarse como incluido en el convertidor. Este es necesario cuando la velocidad de cambio de la señal a convertir es excesiva comparada con el tiempo de conversión.

Los principales inconvenientes de esta configuración son la susceptibilidad a interferencias debido al transporte de señales de bajo nivel hasta el amplificador, alejado de los transductores. La perturbación del multiplexor al sumarse sus errores directamente a la señal de interés y la dificultad de adaptar el amplificador a todos los transductores a la vez, además de que requiere un elevado ancho de banda para tener un establecimiento rápido después de cada conmutación de canal.

Utilizando la estructura que se muestra en la figura 1.10 (sistema multiplexado de alto nivel) se superan algunos de estos inconvenientes, pero se deja de compartir el amplificador, y algunas veces el filtro. Con este esquema el amplificador puede estar muy cerca del transductor, además de mejor adaptado a las características de éste.

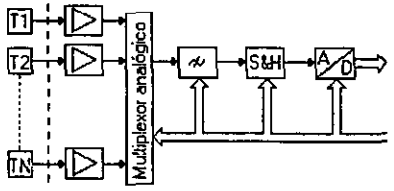


Figura 1.10. Sistema de multiplexado de alto nivel.

El principal problema se presenta cuando el número de canales es elevado, debido al multiplexor analógico, pues posee una velocidad de exploración limitada, además de que puede presentar dificultades con respecto a la compatibilidad de niveles; pero esto puede corregirse con un multiplexor digital, ya que estos no presentan ese tipo de problemas. La estructura que se emplea es entonces la mostrada en la figura 1.11, el cual es un sistema descentralizado y con multiplexor digital [9].

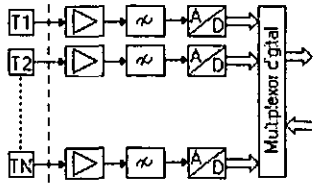


Figura 1.11. Sistema descentralizado con multiplexor digital.

El costo monetario de los convertidores A/D ha ido decreciendo, con lo que la alternativa del multiplexado digital es ya no sólo perfectamente válida, sino común. Las grandes ventajas de esta configuración son que el procesado de la señal de cada transductor puede adaptarse perfectamente a éste, además de que los convertidores y los amplificadores pueden ser de diferente resolución y exactitud.

Existe también otra alternativa cuyo empleo depende de las distancias entre los transductores y los circuitos o sistemas compartidos, ya que algunas veces el costo de los cables de conexión puede ser muy elevado [9]. Este esquema se muestra en la figura 1.12, donde todas las salidas que se conmutan secuencialmente comparten un solo canal físico.

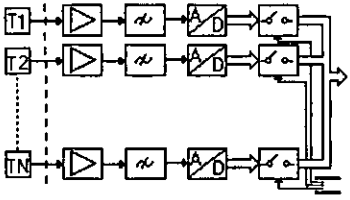


Figura 1.12. Sistema con canal físico compartido.

1.5. Máquina de Medición por Coordenadas (MMC)

Las máquinas de medición por coordenadas (MMC, figura 1.13. 1.14 y 1.15) recogen información dimensional detallada desplazando un palpador a lo largo de las superficies de la pieza de trabajo. La mayoría de las MMC adquieren los datos utilizando un palpador de gatillo que hace contacto en puntos individuales de la pieza de trabajo.

Esta técnica de medida de puntos individuales puede recolectar datos, por lo general, a velocidades máximas de 50 ó 60 puntos por minuto, velocidades considerablemente mayores que las obtenidas con instrumentos manuales de medición.

Lo que hace valiosa la metrología de coordenadas, como una herramienta de control de procesos, es que puede ser usada para medir con exactitud objetos en un amplio rango de tamaños y configuraciones geométricas, y discernir la relación entre diferentes rasgos de una pieza de trabajo. Esta flexibilidad, y la velocidad de operación de la medición por coordenadas comparada con las técnicas de superficies planas y de galgas fijas, significa que los resultados de la medición pueden ser utilizados para refinar, de una manera económica, aplicaciones de procesos de manufactura; además de analizar las tendencias del proceso.

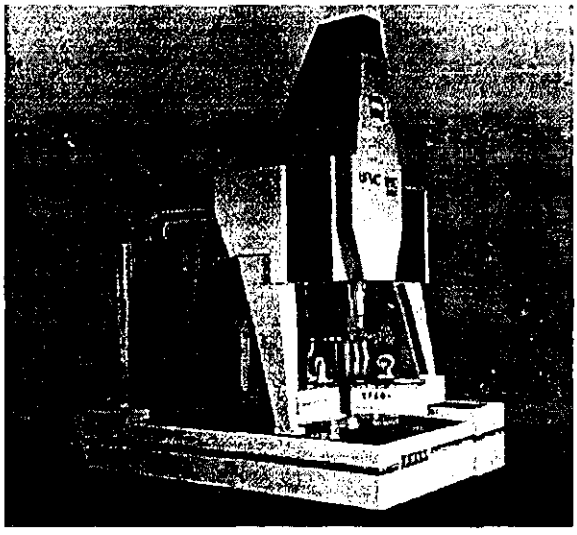


Figura 1.13. MMC 1

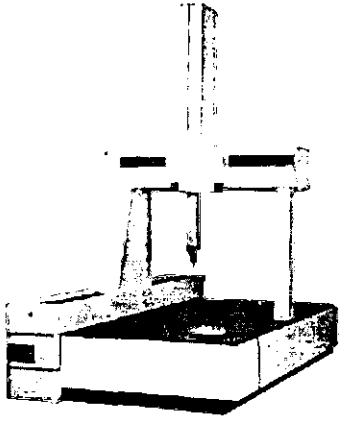


Figura 1.14. MMC 2

CAPITULO 2

Introducción a la programación

Una de nuestras preocupaciones durante la fase de planeación de esta tesis era el presentar al usuario una interfaz totalmente amigable y fácil de utilizar. Para lograr esto decidimos crear la interface bajo un ambiente visual como Windows, para lo cual empleamos Visual C++ versión 6 como herramienta de programación.

Visual C++ ofrece considerables ventajas sobre algunos otros lenguajes de programación, y mejor aún, sobre los lenguajes de programación estructurada. Es obvio que las ventajas que puede ofrecer C++ y en particular Visual C++ depende en gran medida de la aplicación que se esté desarrollando; es decir, debemos tener en cuenta la relación esfuerzo-beneficio, pues no es aconsejable crear un programa bajo Visual C++ que tan sólo multiplique dos cantidades, ya que el código necesario para la aplicación sería mucho mayor que el beneficio obtenido; a decir verdad, el beneficio sería el mismo que el obtenido con un lenguaje de programación estructurada como C; en cambio, el código y el esfuerzo empleados son mucho mayores.

Sin embargo, en programas muy grandes, las ventajas que ofrecen las clases de C++ y la programación orientada a objetos, es considerable. En esta tesis, por la naturaleza de la misma, debíamos pensar en una estructura tal que, además de ser lo suficientemente amigable con el usuario, nos permitiera –desde el punto de vista del programador–

estructurarlo de tal forma que se pudieran agregar nuevas características sin la necesidad de modificar el código, además de la necesidad de que todas estas características o "aditamentos" debían compartir los mismos recursos.

En forma más específica, para nosotros los aditamentos son las geometrías que el programa debía ser capaz de medir y los recursos que debían compartir eran los historiales de medición y el control sobre la tarjeta de interface entre los codificadores ópticos y la PC, así como el sistema de coordenadas independientemente de la rotación y traslación de ejes que se realizara.

Un segundo punto que no podíamos dejar a un lado es que nuestro trabajo estaba acotado, pero el trabajo a futuro sobre la máquina de medición por coordenadas está proyectado para ir mas allá, por lo que nuestro punto de terminación debía ser el punto de inicio de la siguiente etapa; es decir, debíamos asegurar la compatibilidad con el siguiente módulo dejando listas las "conexiones" para acoplarlo sin la necesidad de modificar el código del primer módulo.

Es evidente que la utilización de clases representaba una gran ventaja y ofrecía grandes facilidades para nuestros propósitos, además de permitirnos reutilizar código empleando funciones miembro. Debido a esto y a las ventajas que ofrece C++ en el manejo de clases, elegimos Visual C++ como herramienta de programación para el desarrollo de la aplicación.

De esta forma, estructuramos el programa creando una clase para cada tipo de geometría a medir, es decir, creamos una clase para la medición del círculo, otra clase para medir la línea, otra para el cilindro, otra clase para el cono, y así con cada una de las geometrías. Esto nos permitió agregar todas las geometrías posibles agregando sólo la respectiva clase con sus variables y funciones miembro. Cada una de estas clases esta derivada de la clase *CDialog*, la cual esta derivada a su vez de la clase CWind que encapsula la funcionalidad común a todas las ventanas.

Las clases creadas para la medición de geometrías son:

- CDialogoCilindro
- CDialogoCirculo
- CDialogoCono
- CDialogoEsfera

- CDialogoLinea
- CDialogoParche
- CDialogoPlano
- CDialogoPunto

Cada una de estas clases contiene las variables y funciones miembro que le permiten interactuar con otras aplicaciones y extraer las coordenadas espaciales de la tarjeta PC7266. Otra función miembro de cada clase creada realiza los cálculos necesarios para determinar las características propias de la geometría para la cual fue creada. Otras funciones controlan los eventos en cada uno de los objetos. Cada una de las funciones miembro de estas y otras clases se describen en el anexo A.

En este capítulo se pretende dar una visión general del ambiente de programación bajo Windows, destacando para esto sólo las características mas importantes de este lenguaje, tratando de no abundar en detalles, ya que existen marcadas diferencias entre algunas versiones. Principalmente entre la versión 5 y las versiones anteriores, pues ésta representa el salto de 16 a 32 bits. Este capítulo presenta también un esquema de la organización interna de la PC, así como las bases para la programación de los dispositivos periféricos más comunes.

2.1. Bases de programación en lenguaje C++ para Windows

Independientemente de las herramientas de desarrollo que se utilicen, la programación en Windows es diferente de la antigua programación que se utilizaba bajo MS-DOS. Un programa escrito para MS-DOS es un conjunto de instrucciones que se ejecutan de arriba abajo (denominada programación estructurada), casi en el mismo orden en que el programador lo ha diseñado; mientras que en una aplicación para Windows se presentan todas las opciones posibles a ser ejecutadas por el usuario, en una o más ventanas para que éste elija una de ellas, por lo que la secuencia en que se ejecutará cada una de las instrucciones no puede ser prevista por el programador; pensemos por ejemplo, en una aplicación que consta de dos o más botones (denominados objetos); es evidente que el programador no puede diseñar la aplicación pensando en que el usuario seguirá determinada secuencia. A este tipo de

programación se le denomina programación conducida por eventos y orientada a objetos (el evento en este ejemplo sería entonces pulsar los botones).

Por lo tanto, para programar una aplicación Windows hay que escribir código por separado para cada objeto, e incluso para cada evento del mismo objeto, quedando la aplicación dividida en pequeñas funciones.

2.1.1. Conceptos básicos

Un ***evento*** es una acción reconocida por un objeto (ventana o control). El evento puede ser causado por el usuario (por ejemplo, cuando se pulsa una tecla), o por el sistema (por ejemplo, cuando transcurrió un determinado tiempo), o indirectamente por el código (cuando el código carga, por ejemplo, una ventana). En Windows, cada ventana y cada control pueden responder a un conjunto de eventos predeterminados. Cuando ocurre uno de estos eventos, Windows envía un mensaje al objeto para identificar y ejecutar la función asociada con el objeto para ese evento [8].

Una aplicación Windows utiliza, al menos, una ventana para que el usuario pueda comunicarse con ella. La aplicación es responsable de crear la ventana y comparte la manipulación de la misma con Windows. Es decir, Windows es responsable de manipular el tamaño, la posición y los controles de la ventana y la aplicación manipula el área de trabajo del usuario.

Cuando se escribe una aplicación de C basada en MS-DOS, el mayor requisito es la función *main*. Así, cuando el usuario ejecuta la aplicación, el sistema operativo llama a esta función, después de eso, si la aplicación requiere obtener una entrada del usuario o utilizar servicios del sistema operativo, se llama a la función adecuada, como *getchar*, o algunas otras. Por otra parte, cuando Windows ejecuta un programa, llama a la función *WinMain* del programa, cuya tarea principal consiste en crear la ventana principal de la aplicación, que debe tener su propio código para procesar los mensajes que Windows le envía. La diferencia fundamental entre un programa escrito para MS-DOS y otro escrito para Windows es que el de MS-DOS llama al sistema operativo para obtener la entrada del usuario, mientras que el de Windows procesa la entrada del usuario a través de mensajes del sistema operativo.

Un ***mensaje*** es una notificación que Windows envía a una aplicación en ejecución para indicarle que ha sucedido algo; por ejemplo, una acción del usuario, como un clic sobre un botón, mover el ratón, elegir una orden de un menú, modificar el tamaño de una ventana, etc.; también puede ser una acción propia de la aplicación, como mandarse un mensaje así misma forzando su minimización, o una acción propia de Windows, como enviar un mensaje a una aplicación para que redibuje su ventana. Lógicamente la aplicación responderá al evento ocurrido con una acción específica determinada por la ejecución del código de una función. Esto quiere decir que mandar un mensaje equivale a ejecutar una función dentro de la aplicación [8].

Las ***ventanas*** son objetos con unas propiedades y un código asociado. La ventana de una aplicación generalmente contiene un título que se corresponde con el título de la aplicación, un menú, bordes de tamaño y barras de desplazamiento cuando sean necesarias. A su vez, esta ventana puede desplegar otras ventanas adicionales denominadas cuadros o cajas de diálogo, que contienen controles como botones, cajas de texto, etiquetas, etc., denominados también ventanas hijas.

Un ***handle*** es un valor entero único, utilizado por Windows para identificar un objeto. Por ejemplo, si se requiere pintar un gráfico en una ventana, necesitamos obtener un handle a un contexto de dispositivo y para ello necesitamos el handle de la ventana.

Windows permite el enlazado dinámico, lo que significa que se pueden cargar y enlazar en tiempo de ejecución bibliotecas construidas de forma especial. Múltiples aplicaciones pueden compartir bibliotecas de enlace dinámico (DLL), lo que ahorra memoria y espacio en disco.

Clases

Muchos de los elementos de diseño de C++ (y lenguajes orientados a objetos en general) tratan de emular la forma en la que nosotros interactuamos con el mundo real.

Por ejemplo, dondequiera que volteamos alrededor de nosotros, podemos ver un gran número de objetos. Organizamos todos los objetos alrededor de nosotros en nuestras mentes por "clases". Por ejemplo, tienes un libro en tus manos. Un libro es una clase general de objeto. Probablemente puedes decir, "Este objeto que estoy tomando es clasificado como un libro".

Una jerarquía de clases de objetos encierra la clase "libro", y esta se extiende en dos direcciones. Libros es un miembro de una clase más general "publicaciones". También existen tipos específicos de libros: libros de computación, libros de ficción, libros biográficos, y así. La jerarquía se extiende hacia lo general y hacia lo más específico. En este punto usted esta sosteniendo un simple y particular libro. En palabras de programación orientada a objetos, usted esta sosteniendo una "instancia" de la clase "libro".

Los libros tienen ciertos atributos que son compartidos por todos los libros: tienen pastas, varios capítulos, sin anuncios, etcétera. Estos también tienen atributos compartidos por publicaciones en general: un título, una fecha de publicación, un autor, etcétera. Tiene atributos que son compartidos por todos los objetos físicos: un lugar, tamaño, forma y peso. Esta idea de compartir los atributos es muy importante en C++. C++ modela el concepto de atributos compartidos utilizando "herencia".

Existen ciertas cosas que usted hace con y a diferentes objetos, y estas acciones cambian de objeto a objeto. Por ejemplo, puede leer un libro, y puede hojear sus páginas. Puede ver el título, buscar un capítulo específico, buscar en el índice, contar el número de páginas, etcétera. Estas acciones son en gran parte únicas para las publicaciones: usted nunca se encuentra así mismo hojeando las páginas de un martillo, por ejemplo. Sin embargo, hay acciones que son genéricas a todos los objetos físicos, tales como recogerlos. C++ toma esta realidad del mundo también en su relación, de nuevo utilizando la herencia.

La naturaleza jerárquica de las categorías de objetos, así como nuestra organización jerárquica de atributos de objetos y acciones, están todas embebidas dentro de la sintaxis y vocabulario de C++. Por ejemplo, cuando se diseña un programa usted va a descomponerlo en objetos, cada uno de los cuales tiene una "clase". Usted va a "heredar" características de una "clase base" cuando cree una "clase derivada". Esto es, va a crear clases generales de objetos y entonces hacer clases más específicas de estas, derivando lo particular de lo general. Usted va a "encapsular" los datos encontrados en un objeto con "funciones miembro", y como extienda una clase, va a sobrecargar o sobrevaluar las funciones de la clase base, que en el mundo de C++ se conoce como "overload" y "override" respectivamente. Veamos un ejemplo para describir lo anterior desde el mundo de la programación.

Un ejemplo clásico de la programación orientada a objetos es un programa de gráficos que le permita dibujar objetos –líneas, rectángulos, círculos y tales formas geométricas– en la pantalla. ¿Que es lo que todos

estos objetos tienen en común?. Todos los objetos tiene un lugar en la pantalla. Sin embargo, como diseñador del programa usted crearía una "clase base" o en otras palabras, "una clase de objetos genérica" que encierra los atributos encontrados en todos los objetos que aparecen en la pantalla. La clase base puede ser llamada "forma" para identificarla rápidamente. Usted va entonces a "derivar" diferentes objetos (círculos, cuadrados, líneas) de esta clase base, agregando nuevos atributos que son específicos a estos objetos. Un círculo específico dibujado en la pantalla utilizando la clase círculo sería entonces una "instancia" de la clase círculo, que heredó algunos de sus comportamientos de la clase *forma* más genérica.

Es posible crear esta forma de jerarquía con estructuras normales en C, pero esto no se acerca siquiera a la facilidad con la que se puede hacer en C++. C++ contiene sintaxis para manejar herencia. Por ejemplo, en C usted podría crear una estructura base que encierre la ubicación de los objetos en la pantalla y el color. Entonces, estructuras de objetos específicos pueden incluir esta estructura base y agregarla. C++ hace este proceso más fácil, por lo que va un paso adelante. En C++, las funciones pueden ser depositadas dentro de una estructura, y esta es llamada una "clase". Por tanto, la clase base puede tener "funciones miembro", como son llamadas en C++, que permiten a un objeto ser desplazado y recoloreado. Las "clases derivadas" pueden utilizar estas funciones miembro tal como son, o agregarlas en nuevas funciones miembro para incrementar la funcionalidad, o sobrevaluar las funciones miembros existentes para cambiar el comportamiento. La característica más importante para diferenciar C de C++ es esta idea de una "clase", tanto en un nivel conceptual como sintáctico. Las clases nos permiten utilizar todas las características normales de la programación orientada a objetos –encapsulamiento, herencia y polimorfismo– en los programas de C++.

Evolución de las clases

Una clase es simplemente una extensión de una estructura en C. Básicamente, una clase le permite crear una estructura, y permanentemente asociar todas las funciones relacionadas a esa estructura. Este proceso es conocido como encapsulamiento. Este es un concepto muy simple, pero esto es el corazón de la programación orientada a objetos: datos + funciones = objetos. Las clases pueden también ser construidas encima de otras clases utilizando la herencia. Bajo la herencia, una nueva clase extiende su clase base. Finalmente,

nuevas clases pueden modificar el comportamiento de sus clases base, una capacidad conocida como polimorfismo.

Una de las mejores formas para entender las clases y su importancia para un programador es comprender como y porque estas evolucionaron. Las raíces del concepto de clases radica en un tópico conocido como "abstracción de datos".

Imaginemos que está observando un cuarto lleno de alumnos, típico de un colegio, escribiendo un programa. Imagine un grupo de esos estudiantes quienes están en su primer semestre del curso de Pascal. Una vez que ellos sepan como crear asignaciones If y ciclos y arreglos, ellos están listos para escribir código, pero ellos no saben aún como organizar sus pensamientos. Si usted les pide crear un simple programa ellos crean una burbuja de código que hace el trabajo de algún modo. Éste no va a ser muy bello, pero va a trabajar.

Imagine que usted pide a esos estos estudiantes crear un programa para jugar el juego del cañón, muy famoso entre los tempranos juegos de computadora: el jugador ve un cañón y un blanco situado en un terreno que cambia de juego a juego. La meta es fijar el ángulo del cañón y la cantidad de pólvora para que la bala de cañón golpee el blanco, salvando algunas colinas u otros obstáculos en el terreno.

Asumiendo que los datos del terreno existen en un archivo de texto consistente de pares de coordenadas. Las coordenadas son los puntos finales de segmentos de línea que definen el terreno. Los estudiantes resuelven que necesitan leer éste archivo de tal forma que puedan dibujarlo, y lo necesitan también en memoria para que puedan también verificar las intersecciones en la ruta de la bala de cañón con el terreno para determinar en donde aterriza. ¿Entonces que deben hacer?. Declaran un arreglo global para encerrar las coordenadas, vaciar el archivo dentro del arreglo y entonces utilizar el arreglo en cualquier parte del programa en que este se necesite.

El problema con este proceso es que ahora el arreglo se ha incrustado a sí mismo en el código. Si un cambio se requiere alguna vez –digamos de un arreglo a una lista ligada– el programa probablemente va a ser desechado y reescrito porque este contiene demasiadas referencias específicas al arreglo, que el cambio es imposible. Desde un punto de vista producción-programación esto no es bueno porque las estructuras de datos frecuentemente cambian en un programa grande.

Un mejor camino para diseñar el programa es utilizar un "tipo de datos abstracto". En este caso, el programador primero intenta decidir como van a ser utilizados los datos. En nuestro ejemplo del terreno el programador puede pensar "necesito ser capaz de cargar los datos del terreno de donde sea que estos provengan, y dibujar el terreno en la pantalla y ver si la ruta de la bala del cañón intersecta con el terreno." Note que esto es hecho de forma abstracta (no se hace mención de un arreglo o una lista ligada en ningún lado). Entonces el programador crea funciones para implementar estas capacidades. Estas funciones pueden ser llamadas cargar_terreno, dibujar_terreno, y verificar_interseccion_terreno. Estas funciones son utilizadas a través del programa.

Las funciones actúan como una barrera. Estas esconden la actual estructura de datos del programa. Si después tiene que cambiar la estructura de datos, digamos de un arreglo a la lista ligada, la mayoría del programa permanece sin cambio, sólo las tres funciones tiene que cambiar. El programador ha tenido éxito en crear un "tipo de datos abstracto".

Muchos lenguajes formalizan este concepto. En Pascal se puede utilizar una "unidad", en C se puede utilizar una "librería". Ambos le permiten crear y compilar separadamente un archivo que contenga la estructura de datos y las funciones para accesarlo. Usted puede especificar que la estructura de datos esta "escondida", lo cual significa que el arreglo sólo puede ser accesado por las funciones en esa unidad o librería. En suma, la unidad puede ser compilada de tal forma que el código dentro pueda ser ocultado: otros programadores pueden llamar las funciones debido a la interface pública disponible, pero ellos no pueden ver o modificar el código actual.

Las unidades de Pascal y las librerías de C representan un paso en una cadena evolutiva. Estos inician el ataque al problema de la abstracción de datos pero no van lo suficientemente lejos. Funcionan, pero hay problemas:

1. No hay un camino fácil para modificar o extender el comportamiento de la librería después de que esta es compilada.

2. Estos tipos abstractos no se mezclan muy bien con el lenguaje original. Sintácticamente estos son un desorden, y no utilizan ninguno de los operadores como lo hacen los tipos "normales". Por ejemplo, si crea un nuevo tipo para el que una operación de adición es natural, no

hay forma para usted de utilizar el signo "+" para significar la operación (necesita crear una función llamada sumar).

3. Si usted esconde un arreglo en una unidad solo puede tener un solo arreglo. No puede crear múltiples instancias de los tipos de datos a menos que modifique el código.

Las clases de C++ eliminan estas deficiencias.

En respuesta a estos problemas, los lenguajes orientados a objetos tales como C++ ofrecen facilidad, varios caminos para implementar la abstracción de datos. Lo que tenemos que hacer es modificar nuestra forma de pensar de tal forma que pensemos en los problemas en una forma abstracta.

Primero que nada, debemos intentar pensar en términos de "tipos de datos". Siempre que creamos un dato, necesitamos pensar en todas las cosas que queremos hacer con ese tipo dc datos y entonces conjuntar las funciones para el tipo. Por ejemplo, digamos que queremos crear un programa que requiere un tipo de datos rectángulo conteniendo dos pares de coordenadas. Deberíamos pensar en qué necesitamos hacer con este tipo. Probablemente necesitemos realizar las siguientes acciones: fijarlos a un valor, verificar la igualdad con otro rectángulo, verificar las intersecciones con otro rectángulo, y verificar si un punto esta dentro del rectángulo. Si necesitamos un tipo de dato *terreno*, siguiendo con el mismo proceso debemos crear funciones para cargar los datos terreno, dibujar el dato, etcétera. Después ligamos estas funciones a los datos. Hacer esto por cada tipo de dato que necesitemos en un programa es la esencia de la programación orientada a objetos.

La otra técnica esencial cuando pensamos en una forma orientada a objetos envuelve un entrenamiento mental para pensar en una jerarquía "genérico a específico". Por ejemplo, cuando pensamos en el objeto terreno, probablemente notamos algunas similitudes entre éste y una lista. Después de todo, en algún lugar hay una lista de coordenadas que es cargada de un archivo. Una lista es un objeto genérico que puede ser utilizado en muchos lugares. Así que necesitamos crear una clase de lista genérica y construir el objeto terreno encima de ésta.

En resumen, podemos decir que el proceso del diseño orientado a objetos se puede descomponer en seis pasos:

1. Descubrir los objetos necesarios para el programa y agruparlos en clases.

2. Para cada clase, determinar sus características o propiedades (que formalmente debemos llamar datos miembro o variables instancia de una clase).

3. Para cada clase determinar las acciones, operaciones o responsabilidades de las mismas (denominadas funciones miembro o métodos de una clase).

4. Determinar la relación entre las clases.

5. Declarar las clases y todos los datos miembro y funciones miembro para cada clase.

6. Diseñar algoritmos para todas las funciones miembro que provengan de cada clase.

Así mismo, una clase –en su forma más general– esta formada por un constructor, un destructor y variables y funciones miembro.

2.1.2. Documentos y vistas

La ***vista***, objeto derivado de la clase CView, es una ventana que actúa como interfaz entre el usuario y los datos almacenados en el ***documento***; esto es, un objeto que indica cómo se han de visualizar los datos [6]. En muchas aplicaciones, un *documento* representa a una unidad de datos que el usuario abre con la orden *Abrir* del menú *Archivo* y guarda con la orden *Guardar* del mismo menú. El documento es un objeto de una clase derivada de CDocument, y es el encargado de almacenar los datos de la aplicación. Una vista es una ventana hija de la ventana marco. Esto es, la vista es una ventana que aparece cubriendo totalmente el área de trabajo de la ventana marco (Figura 2.1).

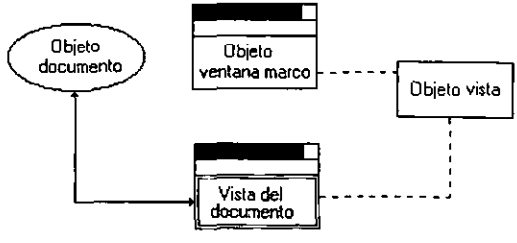


Figura 2.1. Esquema documento-vista.

Para visualizar los datos, la ventana marco, derivada de CFrameWnd, llama a las funciones miembro de la vista para que realicen la función requerida. Por esto, en una aplicación que manipule datos, además del objeto aplicación, que inicia y finaliza la aplicación, son necesarios, por lo menos, un objeto documento, un objeto ventana marco y un objeto vista. Por eso *AppWizard*, cuando genera el esqueleto de una aplicación con estas características, implementa cuatro clases: CAplicacionApp, CMainFrame, CAplicacionDoc y CaplicacionView [7].

La arquitectura *documento-vista* es la característica más importante de la biblioteca MFC (Microsoft Foundation Class) ya que proporciona un marco genérico, y no específico de la aplicación, para gestionar el almacenamiento y la visualización de los datos específicos de una aplicación. Una vista es la responsable de la visualización y modificación (pero no del almacenamiento) de los datos de un documento, y puede acceder a cualquier información que necesite de éste. Cada vista esta vinculada a un solo documento y gestiona el área de trabajo de la ventana marco en la que se visualizan los datos. Una aplicación puede tener uno o más documentos y una o más vistas por documento [6].

2.1.3. Componentes de Visual C++

Entre las características más sobresalientes de Microsoft Visual C++, podemos destacar los siguientes componentes [8]:

- Developer Studio. Este (Estudio de Desarrollador) es un entorno integrado de desarrollo para Windows que es compartido por Visual C++, Visual Java++, Visual Basic y otros. Este estudio o entorno de desarrollo es la ventana y componentes de la misma que se despliega cuando se entra a Visual C++.

- La biblioteca de clases MFC (Microsoft Foundation Class), que es la encargada de dar soporte a los objetos de Windows tales como ventanas, cajas de diálogo, controles, etc., así como objetos GDI (Graphic Device Interface) tales como lápices, pinceles, mapas de bits.

- Un asistente para el desarrollo de aplicaciones, AppWizard, que es un generador de código que crea un esqueleto de la aplicación con características, nombres de clase y archivos de código fuente.

- Una interfaz para múltiples documentos (MDI - Multiple Document Interface) que permite crear una aplicación con una ventana principal y múltiples ventanas de documento.

- Editores de recursos. Estas herramientas permiten editar la tabla de aceleradores de teclado, ventanas de diálogo, los iconos, los menús, la tabla de cadena de caracteres, las barras de herramientas o la versión de la aplicación.

- El enlazador lee los archivos OBJ y RES generados por el compilador de C/C++ y el compilador de recursos, y accede a los archivos LIB para el código de la MFC, el código de la librería de tiempo de ejecución y el código de Windows, escribiendo después el archivo ejecutable del proyecto.

- Spy++. Es una herramienta que permite visualizar las características de una ventana (dimensiones, estilo, ventana padre, entre otras) así como los mensajes que recibe esa ventana.

2.1.4. Jerarquía de ventanas

Windows mantiene una organización jerárquica de sus ventanas. Cada ventana tiene una ventana padre y cero o más ventanas hermanas. Las ventanas hijas están limitadas al área de trabajo de la ventana padre. Las ventanas propietarias y las ventanas en propiedad, como las ventanas de diálogo, pueden aparecer en cualquier lugar de la pantalla. A continuación se listan las características más importantes de la jerarquía de ventanas [7]:

- La raíz de todas las ventanas es el escritorio (desktop) creado por Windows en su arranque.

- Una ventana de nivel superior es una ventana que no tiene ventana padre o que su ventana padre es el escritorio.

- La ventana padre de una ventana hija es una ventana de nivel superior u otra ventana hija más alta en la jerarquía.

- Una vista es una ventana hija de la ventana marco que la contiene.
- Una ventana de nivel superior puede tener un propietario que sea otra ventana de nivel superior. Per ejemplo, la ventana principal de la aplicación es la propietaria de la ventana de diálogo *Acerca de ...* y la ventana padre de ambas es el escritorio.
- Una ventana propiedad de otra siempre aparece encima de su propietaria y desaparece (en el caso de que sea no modal) si su propietaria es minimizada.
- La ventana de diálogo no es una ventana hija porque no esta limitada al área de trabajo de la ventana que la visualiza; es decir, puede aparecer en cualquier parte de la pantalla, pero es propiedad de ella.

2.1.5. Archivos que componen una aplicación

Cuando se utiliza AppWizard para generar el esqueleto de una aplicación mínima, éste crea las clases mínimas para el desarrollo de la misma y crea un directorio de la aplicación en el que guarda una seric de archivos que contienen el código mínimo para la aplicación. Para explicar cada uno de estos archivos, supongamos que hemos creado mediante AppWizard una aplicación denominada *MiAplicacion*. Entonces, éste último ha creado los archivos:

- MiAplicacion.dsp — Un archivo de proyecto que permite que Developer Studio construya la aplicación.
- MiAplicacion.dsw — Un archivo de entorno de trabajo que carga el proyecto cuando se trata de editarlo.
- MiAplicacion.rc — Archivo de recursos que contiene los datos de los objetos para crear la interfaz del usuario.
- MiAplicacionView.cpp — Archivo de implementación de la clase vista que contiene las funciones miembro de la clase CmiAplicacionView. Es decir, este archivo contiene el código fuente de

la aplicación.

- MiAplicacionView.h — Archivo de cabecera de la clase de vista que contiene la declaración de la clase CmiAplicacionView.
- MiAplicacion.opt — un archivo binario que le indica a Developer Studio qué archivos están abiertos para este proyecto y cómo están organizadas las ventanas.
- MiAplicacion.ico — Icono de la aplicación, un recurso.
- resource.h — Un archivo de cabecera que contiene las definiciones de constante *#define*.

Por último, una vez que se han generado los archivos necesarios y agregado el código necesario, para poder crear el archivo ejecutable, Developer Studio ejecuta el proceso que se muestra en el diagrama de la figura 2.2:

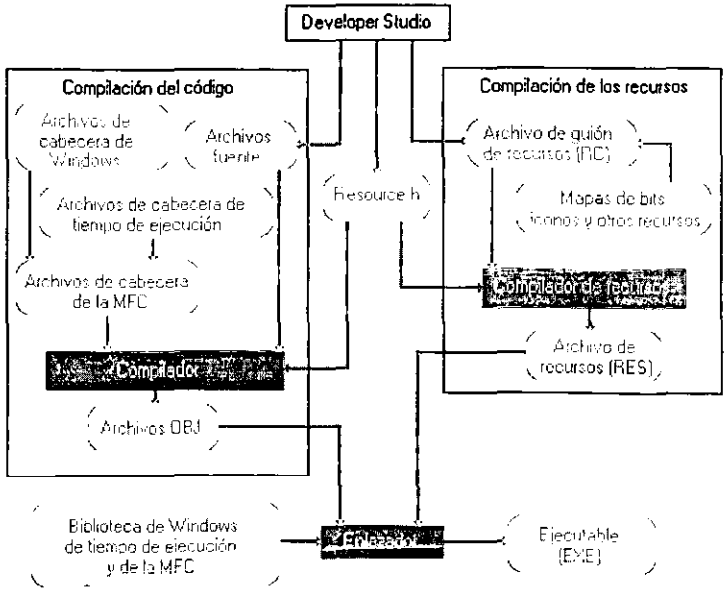


Figura 2.2. Proceso para crear el archivo ejecutable.

2.2. Organización de la PC

La mayoría de las computadoras que se comercializan hoy en día están basadas en la arquitectura Von Neumann, por lo que la organización de la PC que trataremos será precisamente basada en esta arquitectura.

En su forma mas simple, la arquitectura Von Neumann consiste de tres partes: una unidad central de procesamiento (CPU), una memoria y un dispositivo de conexión que puede transmitir datos entre estos dos [12], tal y como se observa en la figura 2.3.

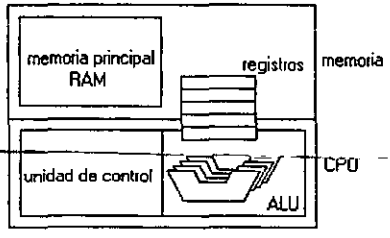


Figura 2.3. Arquitectura Von Neuman.

2.2.1. Memoria

La memoria consiste de localidades de almacenamiento en las que los datos pueden ser colocados y de donde estos mismos datos pueden ser recuperados posteriormente. La memoria puede dividirse a su vez en dos subclases: la memoria de registro y la memoria principal.

Los registros son la primer forma de memoria, aunque estos no pueden almacenar suficientes datos aún para un programa muy simple, algunos de ellos son reservados para propósitos especiales. Los registros tampoco pueden almacenar un valor por un largo periodo de tiempo, pero su principal ventaja es su velocidad, mucho más rápida que la memoria principal [12].

Para que un programa pueda ser ejecutado, éste debe estar almacenado, o cuando menos una parte de él, en la memoria principal, también llamada memoria central, interna o RAM (Random Access Memory). Las computadoras modernas tienen millones de localidades de almacenaje en memoria principal, las cuales pueden almacenar grandes cantidades de datos utilizados en programas, aunque la velocidad de acceso a estas es menor que la de los registros. Este tipo de memoria tiene las siguientes propiedades:

- Es secuencial y consiste de palabras: La memoria de la computadora de Von Neumann es lineal y aparecen como un vector de celdas, llamadas palabras, referenciadas con direcciones secuenciales 0, 1, 2,...,n-1.

- Esta no distingue entre instrucciones y datos: El modelo de Von Neumann permitía que las instrucciones fueran almacenadas en la misma memoria como datos. Las instrucciones eran indistinguibles de los datos y estas podían ser almacenadas en la misma forma que los datos. La interpretación de una palabra en memoria únicamente depende del estado de la máquina en el momento en el que la palabra es buscada en la memoria. Las computadoras con esta propiedad son llamadas computadoras de programa almacenado.

- El significado no es parte inherente de los datos, es decir, no hay nada que distinga un entero de una palabra representando una cadena de caracteres.

La función principal de la unidad de memoria consiste en gestionar los procesos que se encargan de almacenar y recuperar la información. El esquema general de una unidad de memoria es el siguiente (figura 2.4):

Registro de dirección de memoria: Antes de realizar una operación de lectura/escritura (L/E), se coloca en este registro la dirección de la celda que será utilizada.

Decodificador de dirección: Se activa cada vez que se produce una orden de L/E, conectando la celda de memoria, cuya dirección se encuentra en el registro de dirección, con el registro de datos y posibilitando la transferencia de los datos.

Registro de datos: En este registro se almacena el dato que se ha leído de memoria o que se escribirá en esta.

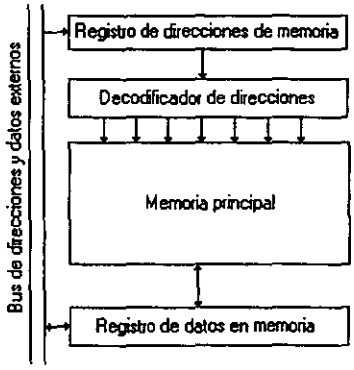


Figura 2.4. Unidad de memoria.

2.2.2. Unidad Central de Procesamiento (CPU)

La unidad central de procesamiento, también llamada CPU por sus siglas en inglés (Central Processing Unit), es el cerebro de la computadora, y es la encargada de controlar y ejecutar todas las operaciones del sistema. Para esto, extrae una a una, las instrucciones del programa ubicado en la memoria principal. La unidad central de procesamiento está compuesta de dos elementos: la unidad aritmético-lógica y la unidad de control.

Unidad Aritmético Lógica

La ALU (Arithmetic Logic Unit) está encargada de realizar las operaciones elementales de tipo aritmético y lógico, utilizando el bus de datos como medio de comunicación con otras unidades. La operación a realizar por la ALU (suma resta, multiplicación, desplazamientos, etc.) se determina por las señales enviadas de la unidad de control [13].

La unidad aritmético lógica esta a su vez formada por los siguientes componentes (figura 2.5)

- Circuito operacional: Esta formado por todos los circuitos necesarios para la realización de operaciones con los datos procedentes del registro de entrada.

- Registro de entrada: Contiene los datos u operandos que intervienen en una instrucción antes de que la operación sea realizada por el circuito operacional. También es empleado como almacenamiento de resultados intermedios o finales.

- Registro de estado: Contiene el conjunto de indicadores que permiten conocer las condiciones que se dieron en la última operación realizada y que deberán ser tomadas en cuenta para las posteriores operaciones.

- Registro acumulador: Contiene los datos que se están tratando a cada momento: Almacena los resultados de las operaciones realizadas por el circuito operacional.

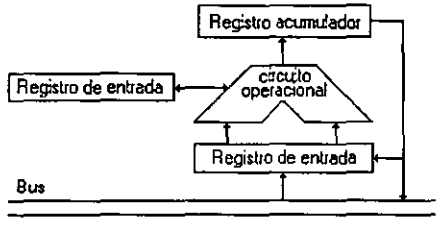


Figura 2.5. Componentes de la unidad aritmético lógica.

Unidad de Control

La unidad de control se encarga de administrar los recursos de la computadora, controlando y dirigiendo la información a las distintas unidades. De forma mas especifica, las funciones de la unidad de control son: controlar la secuencia en la que se ejecutan las instrucciones, controlar el acceso del procesador a la memoria principal, regular la sincronización de todas las operaciones que realiza la CPU, así como enviar señales de control y recibir señales de estado de las demás unidades.

La unidad de control consta a su vez de los siguientes componentes [13]:

- Contador de programa: Contiene la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar, e incrementa su valor

en uno cada vez que concluye una instrucción, a excepción de instrucciones tales como saltos o rupturas.

- Registro de instrucción: Mantiene temporalmente la instrucción que la unidad de control esta ejecutando o interpretando en ese momento.
- Decodificador: Este interpreta la instrucción, extrayendo para eso el código de operación de la instrucción en curso.
- Reloj: Proporciona los impulsos eléctricos que determinan los instantes en que han de comenzar los distintos pasos de que consta una instrucción.

2.3. Programación de periféricos para la PC

Hoy día existe una gran variedad de dispositivos periféricos, los cuales se van mejorando día con día. En el presente apartado no se pretende explicar la programación de todos y de cada uno de los mismos, sino de describir la programación de los periféricos de una forma generalizada, así como los factores o elementos que se deben tomar en cuenta para la correcta programación de los mismos, ya que el formato y las instrucciones que se utilizan dependen del lenguaje que se utilice como herramienta de programación, al igual que del sistema operativo instalado.

Los periféricos son unidades de las que dispone el procesador para comunicarse con el mundo exterior; estos son máquinas empleadas para transferir datos desde o hacia determinado medio de información, por lo que se clasifican como periféricos de entrada o de salida. El principal problema entre los dispositivos periféricos y la CPU es precisamente la conexión entre ellos, debido a que las características de los dispositivos de entrada/salida (E/S) difieren de los del procesador (velocidad, niveles electrónicos para la representación de datos, longitud de palabra, etc.). Para evitar estos problemas, se utilizan interfaces electrónicas llamadas *controladores periféricos*. Cada periférico necesita su propio controlador para poder comunicarse con la CPU. El controlador esta formado por un conjunto de circuitos de adaptación y se encarga de la transferencia de datos entre la CPU y el periférico. La transferencia de información la

realiza mediante los *puertos* de E/S, que son registros conectados directamente a uno de los buses de la computadora. Cada puerto tiene asociada una dirección, de tal forma que el procesador ve al periférico como un puerto o conjunto de puertos (ver figura 2.6).

Los objetivos de un controlador son:

- Selección o direccionamiento del periférico: La CPU coloca en el bus de direcciones la dirección del puerto con el que quiere comunicarse. Sólo un puerto puede estar conectado eléctricamente al bus de datos en un determinado instante. El controlador identifica si la dirección del bus de direcciones corresponde con su código para proceder al intercambio de información.

- Almacenamiento temporal: El controlador dispone de uno o varios puertos para almacenar temporalmente los datos a transferir.

- Sincronización: Debido a que la velocidad de la CPU es muy superior a la de un periférico, el controlador sincroniza el flujo de información a través del envío y recepción de señales de control y de estado para evitar la pérdida de datos.

- Control del periférico: La CPU se comunica con el controlador para conocer su estado o enviar ordenes.

- Conversión de datos: El controlador se encarga de adaptar las características eléctricas y lógicas de las señales empleadas por el dispositivo de E/S y el bus.

- Detección de errores: En el controlador se realizan funciones de inclusión/detección de paridad debido a que la operación de transferencia de datos es sensible a errores.

- Gestión de la transmisión de bloques de información: Puesto que existen periféricos que intercambian información mediante conjuntos de palabras y no mediante palabras aisladas, el controlador puede disponer de un contador que controle el número de palabras recibidas/enviadas.

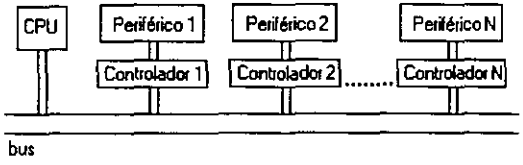


Figura 2.6. Organización de los dispositivos periféricos.

2.3.1. Organización de las E/S

Existen tres métodos para establecer y supervisar las operaciones de transferencia de datos de E/S:

Entrada/Salida programada. El inicio de la operación de E/S se realiza por iniciativa de la CPU, dependiendo del programa en ejecución. Existen en este caso dos variantes:

- ***E/S programada sin consulta de estado.*** La CPU decide el momento en el que realiza la transferencia, por lo que el dispositivo debe estar siempre listo para recibir los datos o enviarlos.

- ***E/S programada con consulta de estado (Polling).*** Antes de realizar la operación de E/S, la CPU lee el estado del controlador para comprobar si el dispositivo esta listo. En caso de no estarlo, se pueden seguir dos alternativas: se detiene el programa y se sigue consultando el estado hasta que el periférico esté listo (E/S programada con bloqueo de programa), o el programa sigue ejecutándose y periódicamente se consulta el estado del controlador para comprobar si éste esta listo (E/S con consulta periódica).

Interrupciones. Mediante este método, se permite a los periféricos actuar de forma independiente e interrumpir al procesador cuando existe algún dato o bloque de datos a transmitir desde o hacia la memoria.

Acceso directo a memoria (DMA – Direct Memory Access). En este caso los periféricos tienen acceso directo a la memoria, y una vez que el

procesador les asigna una tarea, la ejecutan de manera totalmente independiente, indicando al procesador, mediante interrupciones, el fin de la tarea asignada (este es el modo de funcionamiento de la mayoría de las unidades de disco).

2.3.2. Direcciones de E/S

Como ya se mencionó anteriormente, cada dispositivo periférico tiene asignado un rango de direcciones de entrada/salida y cada uno de estos rangos sólo puede ser utilizado por un único dispositivo; es decir, por ejemplo si el teclado tiene datos para el sistema, pone estos datos en una dirección y la CPU los recoge de ahí. Lo mismo sucede con el ratón, los puertos serie o cualquier otro dispositivo, pero en una dirección de E/S diferente.

Las computadoras personales poseen 1024 direcciones de E/S, que van desde la posición 000 hasta la 3FF, de las cuales las primeras 256 son ocupadas por componentes propios del sistema tales como el reloj, el DMA, etc. El resto de las direcciones son utilizadas para las tarjetas de expansión; estas direcciones van desde la 100 hasta la posición 3FF. Las asignaciones que tienen comúnmente las direcciones de E/S se muestran en la siguiente tabla 2.1, aunque estas pueden variar dependiendo del sistema operativo que se utilice o el tipo de procesador [5].

Para accesar a las direcciones de E/S, el procesador debe recibir del programa una instrucción IN o OUT en lenguaje máquina, ya que cada lenguaje de programación tiene su propia forma de representar esta instrucción.

Cuando la CPU recibe uno de estos comandos, activa la línea del bus de control de lectura o de escritura de E/S. Cuando el dispositivo en cuestión detecta que esta línea esta activa y que la dirección del bus de direcciones corresponde con la suya, pone/recoge los datos correspondientes en el bus de datos. La tabla 2.1 muestra los rangos de direcciones de E/S.

RANGO DE DIRECCIONES	UTILIDAD
00-0F	Controlador de DMA #1

20-21	Controlador programable de interrupciones #1
40-43	Reloj programable
60-63	Controlador de los periféricos. Sólo en los XT
60-64	Controlador de teclado. Sólo en los AT
70-71	Dirección de acceso de iniciación de RAM. Sólo en los AT
80-8F	Registro de página de DMA
A0-A1	Controlador programable de interrupciones #2
A0-AF	Registro de máscara NMI. Sólo en los XT
C0-DF	Controlador de DMA #2. Sólo en los AT
F0-FF	Coprocesador matemático. Solo en los AT
1F0-1F8	Controlador de unidades de disco duro. Sólo en los AT
200-20F	Controlador de joystick
210-217	Unidades de expansión
238-23B	Ratón
23C-23F	Ratón alternativo
258	Tarjeta de memoria expandida (LIM)
278-27F	LPT2

2B0-2DF	EGA
2E8-2EF	Puerto serie COM4
2F8-2FF	Puerto serie COM2
300-30F	Tarjeta de red de área local Ethernet
320-32F	Controlador de unidades de disco duro. Sólo en los XT
330-337	Controlador de unidades de discos Bernoulli
378-37F	Puerto paralelo LPT1
380-38F	Tarjeta SDLC
3ª0-3ªF	Tarjeta BSC
3B0-3BF	Adaptador monocromo
3BC-3BF	Puerto paralelo LPT3
3D0-3DF	CGA
3E8-3EF	Puerto serie COM3
3F0-3F7	Controlador de unidades de discos flexibles
3F8-3FF	Puerto serie COM1

Tabla 2.1. Direcciones de E/S.

2.3.3. Interrupciones

Como se mencionó anteriormente, las interrupciones son uno de los métodos para la transferencia de datos de E/S, pero estos a su vez se dividen en dos tipos: las denominadas ***interrupciones por software*** y las ***interrupciones por hardware***.

Interrupciones por software

Las interrupciones por software son rutinas del sistema operativo a las que el programador puede llamar para facilitar su tarea de programación. Estas rutinas realizan determinados procesos de control de los dispositivos conectados al procesador. El sistema operativo tiene dos tipos de interrupciones de software: las interrupciones de la BIOS (sistema básico de entrada/salida), y las interrupciones del DOS.

Las interrupciones del BIOS son rutinas utilizadas para la lectura y escritura en disco, presentación en pantalla, lectura del teclado, control de los puertos de comunicación, actualizaciones de la fecha y hora, manejo de errores, entre otras. En conjunto existen 12 interrupciones del BIOS, divididas en cinco grupos [5]:

1. **Servicios de dispositivos y periféricos**
   - INT 10H: Servicios de pantalla de video.
   - INT 13H: Servicios de disco.
   - INT 14H: Servicios de comunicación (RS232).
   - INT 15H: Servicios de casete.
   - INT 16H: Servicios de teclado.
   - INT 17H: Servicios de la impresora.
2. **Servicios de estado**
   - INT 11H: Servicios de lista de elementos del equipo.
   - INT 12H: Servicios de cálculo del tamaño de la memoria.
3. **Servicios de fecha y hora**
   - INT 1AH: Servicios de reloj.
4. **Servicios de la pantalla de impresión**
   - INT 5H: Impresión de pantalla.
5. **Servicios especiales**

- INT 18H: Activación del Basic de la ROM.
- INT 19H: Activación de la rutina de arranque del ordenador.

Por su parte, DOS ofrece dos tipos de servicios de acceso a las interrupciones. Por un lado, permite el acceso directo por medio de una instrucción INT. Y por otro, accesando a las funciones que se invocan a través de la interrupción 21H, especificando previamente en el registro AH el número de función que se invoca [5].

Los servicios de interrupción de DOS son los siguientes:

- **INT 20H:** Termina la ejecución de un programa
- **INT 21H:** Llamada a función. Se debe colocar en el registro AH el número de la función, entre las cuales, las más importantes son:

  0 Terminación de un programa y devolución del control al DOS

  1 Entrada de carácter con eco

  2 Salida a la pantalla

  3 Entrada por el puerto serie

  4 Salida por el puerto serie

  5 Salida a la impresora

  6 E/S directa a pantalla

  7 Entrada directa de carácter sin eco

  8 Entrada de carácter sin eco

  9 Visualización de una cadena de caracteres

  10 Entrada desde el teclado

  11 Comprobación del estado de entrada

  12 Borra registro de entrada y procede a una introducción de datos

  13 Inicializar la unidad de disco

  15 Abrir un archivo

16 Cerrar un archivo

17 Buscar el primer archivo

18 Buscar el próximo archivo

19 Borrar un archivo

20 Leer registro de archivo secuencial

21 Escribir registro de archivo secuencial

22 Crear archivo

23 Renombrar archivo

33 Leer archivo de acceso aleatorio

34 Escribir archivo de acceso aleatorio

35 Cálculo del tamaño del archivo

42 Obtener la fecha

43 Fijar la fecha

44 Obtener la hora

45 Fijar la hora

- **INT 22H:** Dirección de terminación. Guarda la dirección donde se transfiere el control cuando termina la ejecución del programa.
- **INT 23H:** Dirección de break. Dirección de la rutina que se ejecuta cuando se pulsa Ctrl-Break.
- **INT 24H:** Dirección del manejador de errores críticos. Esta rutina se ejecuta siempre que se produce un error crítico.
- **INT 25H:** Lectura directa de sectores del disco.
- **INT 27H:** Termina un programa y devuelve el control a DOS sin borrar el programa de la memoria. Esta interrupción es utilizada por los programas residentes y los virus informáticos.

Interrupciones por hardware

Existen dos tipos de interrupciones por hardware: las interrupciones internas y las interrupciones externas [5]. Las ***interrupciones de hardware internas*** (también llamadas interrupciones lógicas) son llamadas por el propio procesador cuando se produce una operación incorrecta, por ejemplo, la división por cero.

Las ***interrupciones de hardware externas*** son provocadas por los diferentes dispositivos periféricos conectados al procesador. Cada periférico tiene acceso a una línea de petición de interrupción diferente. A estas líneas de interrupción se les conoce como IRQ (Interrupt Request Line), que van desde IRQ0 a IRQ15 en los procesadores 286 y superiores.

La CPU no censa continuamente las direcciones de E/S de cada dispositivo para ver si hay nuevos datos, sino que cada vez que alguno de estos dispositivos tiene datos que intercambiar con la CPU hace sonar un timbre para llamar la atención del mismo, este timbre son las interrupciones IRQ. Cada vez que un dispositivo activa una de estas interrupciones, la CPU deja lo que esta haciendo y ejecuta la rutina correspondiente a la interrupción activada.

Las direcciones de las rutinas que se deben ejecutar para cada una de las interrupciones, están almacenadas en la tabla de vectores de interrupción. Esta tabla es una serie de direcciones almacenadas en memoria RAM a partir de la dirección 08h. Las interrupciones que se utilizan normalmente en las computadoras personales se muestran en la tabla 2.2.

PROCESADOR	INTERRUPCIÓN	DISPOSITIVO
PC-XT	0	Reloj
Y	1	Teclado
PC-AT	2	Disponible en los XT. Utilizado en los AT como pasarela para las IRQ 8-15
	3	COM2
	4	COM1

	5	Disco duro en los XT
	6	Disco flexible
	7	LPT1
PC-AT	8	Reloj
	9	Red de área local
	10	Disponible
	11	Disponible
	12	Disponible
	13	Coprocesador
	14	Disco duro
	15	Disponible

Tabla 2.2. Interrupciones de hardware.

Las líneas físicas IRQ procedentes de cada periférico no van directamente al chip de la CPU, sino que existe un chip intermediario que se encarga de controlar las interrupciones. Este chip, llamado PIC (Programmable Interrupt Controller) o controlador de interrupciones programable, es el 8259A. Cada uno de estos chips puede controlar hasta ocho interrupciones, por los que las PC de tipo AT o posteriores disponen de dos de estos chips. El PIC asigna prioridades a las interrupciones, previniendo el problema de que dos o mas interrupciones ocurrieran al mismo tiempo. El PIC asigna la prioridad más alta a la IRQ 0, y la prioridad mas baja a la IRQ 7.

2.3.4. Puerto paralelo

El puerto paralelo se utiliza generalmente en la PC para conectar la impresora; sin embargo, esta interfaz es muy flexible y permite ser utilizada para distintas aplicaciones como por ejemplo, la conexión de pequeños robots o de dos procesadores de forma directa.

El puerto paralelo está diseñado para transmitir y recibir datos de 8 en 8 bits, por lo que presenta una mayor velocidad de transferencia con respecto al puerto serie. Otra de las diferencias con respecto a este último, es que las señales eléctricas utilizadas para representar los estados lógicos 0 y 1 son de 0 y +5 volts respectivamente, en lugar de los valores negativos (-3 y –5 [V]) y los positivos (+3 y +5 [V]) que utiliza el puerto serie. La interfaz del puerto paralelo consta de 25 pins, cuyo funcionamiento se describe en la tabla 2.3.

CONECTOR PC	SEÑAL	TIPO SEÑAL	CONECTOR IMPRESORA	DESCRIPCIÓN
1	/STR	Control	1	Activación de transferencia de datos
2	D0	Dato	2	Bit de dato 0
3	D1	Dato	3	Bit de dato 1
4	D2	Dato	4	Bit de dato 2
5	D3	Dato	5	Bit de dato 3
6	D4	Dato	6	Bit de dato 4
7	D5	Dato	7	Bit de dato 5
8	D6	Dato	8	Bit de dato 6
9	D7	Dato	9	Bit de dato 7
10	/ACK	Estado	10	Transferencia de datos correcta
11	BSY	Estado	11	Impresora ocupada
12	PAP	Estado	12	Sin papel
13	OFON	Estado	13	Indicador de impresora en línea
14	/ALF	Control	14	Alimentación de línea automática

15	/FEH	Estado	32	Error
16	/INI	Control	31	Inicialización de impresora
17	/DSL	Control	36	Selección de impresora
18-25	Tierra		19-30, 33	Tierra 0 V
-	0 V		16	-
-	Chasis		17	Tierra protegida
-	+5 V		18	-
-	-		34, 35	Sin usar

Tabla 2.3. Pins del puesto paralelo.

El puerto paralelo dispone de tres registros a través de los cuales intercambia los datos y señales de control. Estos registros son el registro de datos, registro de estado y registro de control. El registro de datos contiene la dirección que será transmitida a la interfaz o que ha sido recibida de la misma. El registro de estado ofrece información del estado del dispositivo conectado, como por ejemplo si éste se encuentra conectado, o si recibe los datos correctamente, entre otros. El registro de control controla el comportamiento del dispositivo, así como la generación de interrupciones de hardware. El registro de datos y el registro de control son registros bidireccionales, mientras que el registro de estado es un registro de solo lectura. La tabla 2.4 muestra los bits de los registros del puerto paralelo, mismos que pueden observarse de forma gráfica en la figura 2.7.

REGISTRO	OFFSET		7	6	5	4	3	2	1	0
Registro de datos	00h	Señal	D7	D6	D5	D4	D3	D2	D1	D0
		Pin	9	8	7	6	5	4	3	2
Registro de estado	01h	Señal	/BSY	/ACK	PAP	OFON	/FEH	-	-	-
		Pin	11	10	12	13	15	-	-	-
Registro de control	02h	Señal	-	-	-	IRQ	DSL	/INI	ALF	STR
		Pin	-	-	-	-	17	16	14	1

Tabla 2.4. Registros del puerto paralelo.

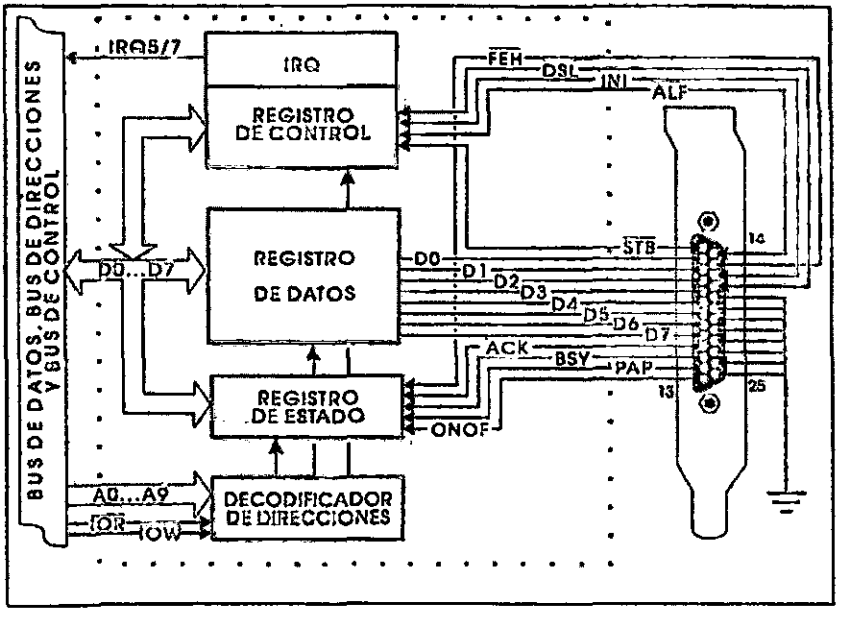


Figura 2.7. Registros del puerto paralelo.

Impresora

En el caso de la impresora, la señal /BSY indica si la impresora está ocupada o no; es decir, si el buffer de la impresora esta lleno o no. El puerto no transfiere un nuevo carácter a menos que el pin /BSY esté activado. La línea /ACK se utiliza por la impresora para aceptar o

rechazar el carácter que recibe. Si la señal /ACK esta activa, indica que la impresora esta todavía ocupada con la recepción.

Cuando la impresora no tiene papel, pone el bit PAP a 1, al mismo tiempo que desactiva la señal OFON y activa /FEH para indicar que hay un error. El bit OFON indica si la impresora está o no en línea. El significado de los bits del registro de estado en la impresora se muestra en la tabla 2.5

/BSY	Ocupada	1 = Impresora no ocupada 0 = Impresora ocupada, fuera de línea o error
/ACK	Aceptar	1 = Transferencia de datos en progreso 0 =Transferencia de datos completa
PAP	Papel	1 = No hay papel 0 = Papel disponible
OFON	En línea	1 = Impresora en línea 0 = Impresora fuera de línea
/FEH	Error	1 = No hay error 0 = Error

Tabla 2.5. Bits de registro de la impresora.

El registro de control vigila el comportamiento de la impresora y la generación de interrupciones de hardware. Cuando el bit IRQ está activo, la interfaz produce una interrupción vía IRQ5 (LPT1) o IRQ7 (LPT2) cada vez que hay una transición de estado 0 a 1 de la señal /ACK. A pesar de esto, el BIOS de la PC normalmente no utiliza este procedimiento para controlar la transferencia de datos a la impresora, sino que simplemente comprueba el estado del bit /ACK.

El bit /INI activa una inicialización de la impresora y el bit ALF es activado cuando se desea que la impresora realice una alimentación de líneas de forma automática después de imprimir cada línea. Por último, el bit STR (Strobe) genera un pulso de activación para que la impresora lea las señales del registro de datos. STR sincroniza la transferencia de datos entre el procesador y la impresora. Esto significa que para transferir datos a la impresora se deben de colocar estos en el registro de datos y desactivar y activar la señal STR. La tabla 2.6 muestra el significado de los bits del registro de control.

IRQ	Petición de interrupción hardware	1 = Habilitado 0 = Inhabilitado
DSL	Selección de impresora	1 = Impresora seleccionada 0 = Impresora no seleccionada
/INI	Inicialización de impresora	1 = Operación normal 0 = Inicializar
ALF	Alimentación de línea automática	1 = Alimentación de línea por impresora 0 = Alimentación de línea por procesador
STR	Activación de transferencia	1 = Transferir datos a impresora 0 = No transferir

Tabla 2.6. Bits del registro de control.

CAPITULO 3

Automatización del proceso de adquisición de lecturas

Recientemente en el Centro de Instrumentos UNAM, particularmente en la Sección de Metrología, se ha desarrollado y construido un prototipo mecánico de una Máquina de Medición por Coordenadas, MMC. El uso extendido de tales instrumentos de medición dimensional ha cobrado un gran auge como parte de los procesos industriales de control de calidad. El amplio mercado para las MMC ha motivado el desarrollo de una versión económica que ofrezca ventajas adicionales sobre los productos disponibles en el mercado. Nuestra idea no es competir en complejidad con las MMC's comerciales, sino rescatar las necesidades industriales en un diseño realista de MMC para los consumidores nacionales. En éste sentido, estamos en la posición de ofrecer un producto competitivo en exactitud y software específico que no abrume al operador con sistemas altamente complicados.

En éste sentido, nuestra labor electrónica y de computación consiste en automatizar el proceso de medición de la infraestructura existente, que hasta el momento era completamente manual y carente de interconexión con sistemas de cómputo. Es indudable que las necesidades de procesamiento en las MMC's actuales requieren una interconexión con computadoras; ya que los procesos de medición, cada vez más complejos, así lo exigen. Dotar a una MMC con interconexión a la computadora permite procesar geometrías que son requeridas en los procesos de inspección industrial. Un software específico para el procesamiento de

geometrías habilita al operador para medir piezas industriales genéricas y determinar la situación espacial de la pieza bajo inspección o compararla contra mediciones patrón o calibres patrón.

En el presente capítulo examinaremos el esquema propuesto para la automatización del proceso de adquisición de lecturas mediante una PC. Tal tarea es vital para el posterior procesamiento de lecturas y la evaluación de geometrías típicas. También examinaremos las características del hardware utilizado que nos permitirá la interconexión electrónica. Finalmente, analizaremos la implementación utilizada en esta tesis para la medición de las geometrías típicas.

3.1. Esquema propuesto

Planteamos el problema de automatizar la captura y administración de lecturas mediante la MMC construida en el CI UNAM; cuya operación, hasta ese momento, era completamente manual; es decir, la única forma de registrar las mediciones hechas a algún objeto era mediante la observación y un registro posterior de los valores que arrojaban los codificadores ópticos, cuya tarea, en conjunto, consiste de posicionar en el espacio el punto palpado del objeto. Una vez registradas todas las mediciones, se procedía a hacer todas las operaciones necesarias para determinar las características dimensionales y geométricas del objeto en estudio. Por ejemplo, para el caso de una esfera, estas características son el diámetro y centro geométrico de la misma. Tal vez el realizar estas operaciones no presente mayor problema, pero cuando las mediciones se realicen a base de puntos en el espacio o cuando la cantidad de objetos en estudio sea una cantidad considerable, la ejecución manual de tantas operaciones se convierte en una tarea abrumadora que consume un periodo de tiempo considerable. Con la automatización, pretendíamos obtener, entre otros, los siguientes beneficios:

- Capturar mediante computadora posiciones x, y y z.
- Creación de un software específico para la medición de geometrías típicas.
- Creación de software específico para transformación de coordenadas

Esto nos traería, indudablemente, un ahorro considerable de tiempo y de trabajo; evitaría también errores en los cálculos debidos al factor humano al determinar las característica propias de los objetos en estudio.

El esquema planteado se puede apreciar en la figura 3.1.

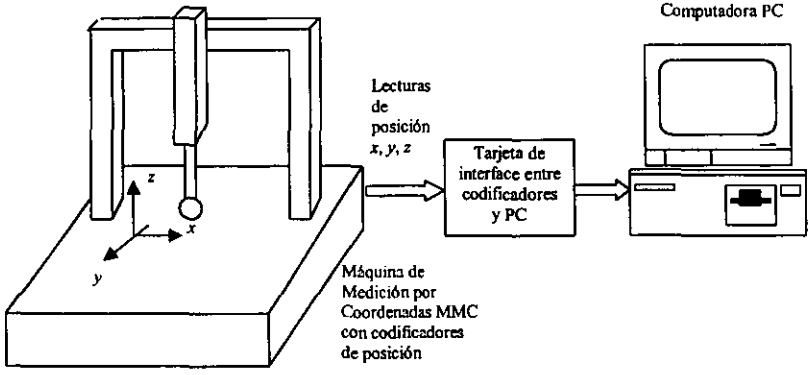


Figura 3.1. Automatización de una MMC.

En el esquema de la figura 3.1, una computadora captura las lecturas de posición x, y, z provenientes de tres transductores ópticos de posición solidarios a la estructura de la MMC. Para lograr esto, se utiliza una tarjeta electrónica como interfaz entre los transductores y la PC. El software específico captura lecturas espaciales y las ajusta a las geometrías típicas. Para observar a detalle las conexiones realizadas entre cada elemento, refiérase al anexo B.

MMC

Una máquina de medición por coordenadas (MMC) "es un equipo de medida, de carácter fijo, concebido para realizar mediciones a partir de al menos tres desplazamientos lineales o angulares generados por la máquina. Al menos uno de los tres desplazamientos debe ser lineal" [2].

La MMC construida en el Centro de Instrumentos y parte fundamental de este proyecto, es una máquina de medir en tres ejes lineales x, y y z, la cual constituye un diseño realista y simplificado para las necesidades de la industria nacional. Algunas de sus características principales son las siguientes:

- Compite en exactitud con equipos similares.

- Equipo diseñado y construido en su totalidad por el CI UNAM.
- Medición en tres ejes.
- Volumen de medición de 0.5 m^3.
- Resolución mínima de 2 μm.

La MMC del CI UNAM se muestra en la figura 3.2.

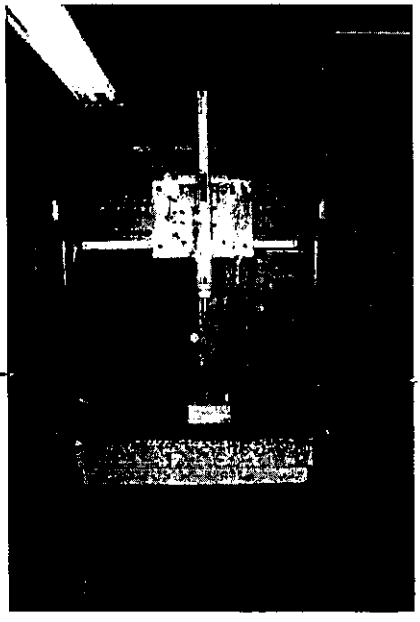


Figura 3.2. MMC CI UNAM.

3.2 Tarjeta de interface entre codificadores incrementales y PC

Como se mencionó anteriormente, en la interface entre los codificadores ópticos y la PC se utilizó equipo comercial. Éste consiste de una tarjeta PC7266, que constituye el medio electrónico a través del cual la PC se conecta a los codificadores ópticos en la MMC. El desplazamiento de los ejes en esta máquina de medición es registrado por los codificadores

ópticos. Entonces la tarjeta PC7266 se encarga de contar las transiciones eléctricas ocurridas en los codificadores ópticos, por lo que la lectura de los contadores no se interrumpe, ya que éste conteo es independiente de la velocidad del procesador de la PC. De esta forma la PC puede encuestar el contenido de los contadores en la PC7266 para procesar tal cuenta e interpretarla como un desplazamiento lineal en la MMC.

La figura 3.3 muestra la tarjeta PC7266.

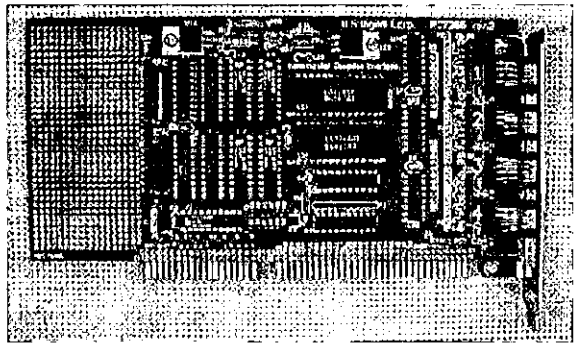


Figura 3.3. Tarjeta US-Digital PC7266.

La tarjeta PC7266 se conecta en una cavidad ISA estándar de 16 bits de computadoras IBM PC y compatibles. Esta incluye dos contadores duales de 24 bits LS7266R1. La versión particular de la tarjeta acepta entradas compatibles con TTL. Cuatro PAL's reciben las entradas de "índice" y "acarreo" y controlan las señales de "carga de contador", "inicio de contador" e "interrupción". Cada uno de los cuatro canales de conexión a transductores ópticos puede programarse independientemente en uno de ocho modos diferentes. Siete "jumpers" son usados para seleccionar la dirección de interrupción IRQ. Esta tarjeta adicionalmente proporciona polarización 5.5 [V] CD a los codificadores externos. La tarjeta incluye capacidades de programación bajo Windows 98 en Visual C++ V6.0.

La figura 3.4 ilustra la implementación en la PC7266 utilizando un solo circuito integrado LS7266R1.

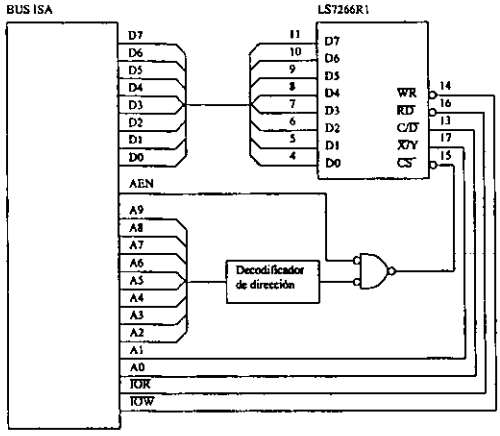


Figura 3.4. Implementación de la PC7266 usando el LS7266.

3.2.1. Características

La tarjeta PC7266 tiene las siguientes características [14]:

- 4 canales de entrada.
- Contadores de 24 bits con capacidad de pre-carga.
- Salidas retenidas de los contadores.
- Resolución multiplicada X1, X2 y X4.
- Interface TTL
- Generador de interrupciones y entradas con índice programables.
- Puerto de entrada/salida de 8 bits.
- Software para desarrollar aplicaciones.
- Area de desarrollo de prototipos.

Los conectores en la tarjeta son modulares tipo telefónico de 8 pines, como los mostrados en la figura 3.5.

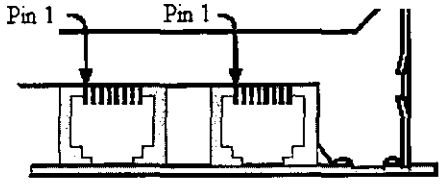


Figura 3.5. Conectores para los codificadores ópticos.

La función en cada pin del conector se describe en la tabla 3.1.

Pin	Nombre	Descripción
1	GND	Tierra común de polarización y datos
2	PWR	Salida de polarización a los codificadores
3	I-	Entrada en cuadratura, I+ invertida
4	I+	Entrada en cuadratura
5	A-	Entrada en cuadratura, A+ invertida
6	A+	Entrada en cuadratura
7	B-	Entrada en cuadratura, B+ invertida
8	B+	Entrada en cuadratura

Tabla 3.1. Función de los pines

La selección de la dirección base que utilizará la tarjeta dentro de la PC se realiza con la ayuda del conector "dip switch" en la PC7266, mostrado en la figura 3.6, y de acuerdo a la tabla 3.2.

HEX	A9	A8	A7	A6	A5	A4
220	1	0	0	0	1	0
240	1	0	0	1	0	0

250	1	0	0	1	0	1
260	1	0	0	1	1	0
300	1	1	0	0	0	0
310	1	1	0	0	0	1
330	1	1	0	0	1	1
340	1	1	0	1	0	0
350	1	1	0	1	0	1
360	1	1	0	1	1	0

Tabla 3.2. Dirección base.

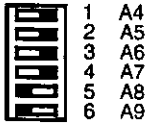


Figura 3.6. Dip switch, 0=bajo, 1=alto.

3.2.2. Mapa de puertos

El mapa de puertos en la tabla 3.3 muestra las funciones que desempeña la PC7266, y que son de utilidad en el momento de implementar nuestra aplicación [14].

Dir. (+base)	Leer / Escribir	Descripción
0	Leer/Escribir	Dato hacia/desde LS7266R1 Codificador #1
1	Leer	Bandera de estado desde LS7266R1 Codificador #1
1	Escribir	Control al LS7266R1 Codificador #1

2	Leer/Escribir	Dato hacia/desde LS7266R1 Codificador #2
3	Leer	Bandera de estado desde LS7266R1 Codificador #2
3	Escribir	Control al LS7266R1 Codificador #2
4	Leer/Escribir	Dato hacia/desde LS7266R1 Codificador #3
5	Leer	Bandera de estado desde LS7266R1 Codificador #3
5	Escribir	Control al LS7266R1 Codificador #3
6	Leer/Escribir	Dato hacia/desde LS7266R1 Codificador #4
7	Leer	Bandera de estado desde LS7266R1 Codificador #4
7	Escribir	Control al LS7266R1 Codificador #4
8	Leer	Estado desde todas las PAL's (cuatro primeros bits)
8	Escribir	Modo buffer para las PAL's 1 y 2
9	Escribir	Modo buffer para las PAL's 3 y 4
A	Leer/Escribir	Hacia/desde el puerto entrada/salida

Tabla 3.3. Mapa de puertos.

3.2.3. Programación

La figura 3.7 es un diagrama de flujo que ilustra los pasos a seguir para operar la tarjeta PC7266.

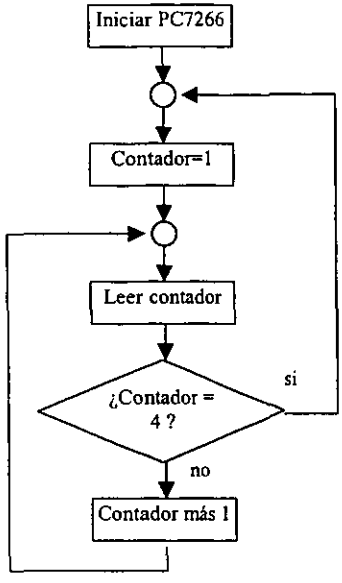


Figura 3.7. Diagrama básico de flujo

3.3 Codificadores incrementales ópticos lineales

La MMC cuenta con tres codificadores ópticos fijados a su estructura de tal forma que se hacen corresponder con los ejes coordenados *x, y, z*. Estos codificadores, fabricados por la marca comercial ACU-RITE, tienen una longitud de aproximadamente 0.5m y una resolución de 2μm. Los codificadores envían señales digitales en cuadratura cuando se produce un desplazamiento; estas señales son utilizadas para codificar la posición y actualizar los registros de posiciones espaciales. Las figuras 3.8 y 3.9 muestran el diagrama y una fotografía respectivamente de los codificadores empleados.

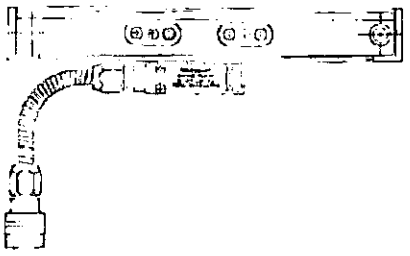


Figura 3.8. Diagrama de los codificadores solidarios a la MMC.

Figura 3.9. Codificadores ACU-RITE utilizados.

Los codificadores ópticos utilizados están basados en la tecnología de escalas de cristal. Esta tecnología es el estándar en la industria para la medición por posiciones, ya que el cristal es un material altamente estable que resiste cambios de tamaño, forma o densidad debidas a variaciones de temperatura o humedad. Esta estabilidad permite la generación de numerosos patrones de línea ultrafina, proveyendo a las escalas de cristal de estos codificadores, una excepcional exactitud por debajo de ±1.5μm (0.00006"). Las escalas ACU-RITE están disponibles en longitudes de desplazamiento desde 1" hasta 120" y con resoluciones de 0.5μm (0.00002") a 10 μm(0.0005").

Una escala de precisión de cristal como la empleada por estos codificadores consta de una escala de cristal de gran calidad óptica y una cabeza fotoelectrónica lectora. La cabeza lectora se desplaza a lo largo de la escala de cristal iluminando una línea patrón de cromo. Este patrón es reconocido por los fotodetectores, los cuales generan a su vez una señal eléctrica (figura 3.10) [1].

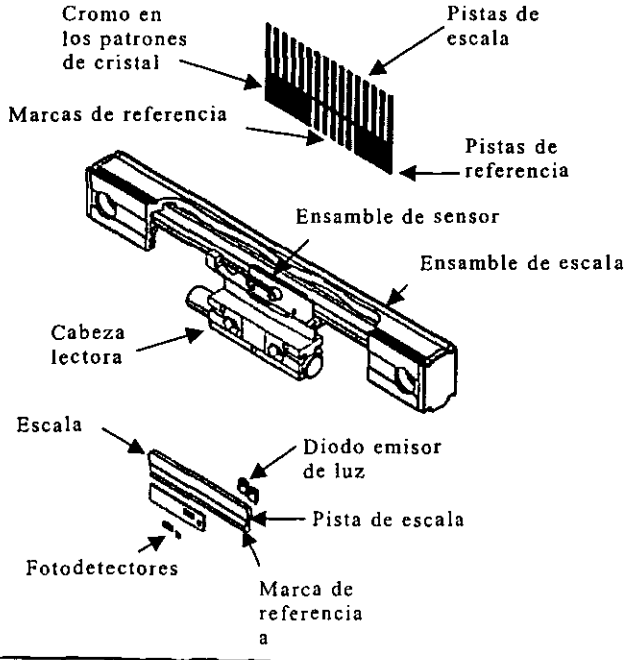


Figura 3.10. Componentes del codificador óptico.

Una escala de precisión de cristal ACU-RITE de 1μm de precisión tiene 1270 líneas de cromo por pulgada, cada una de las cuales es detectada por la cabeza fotoeléctrica lectora cuando estas son desplazadas a través del cristal. Esto proporciona un intervalo de medición fundamental 10 veces más fino que la tecnología de codificadores alternativa más cercana, tal como se muestra en la figura 3.11 [1].

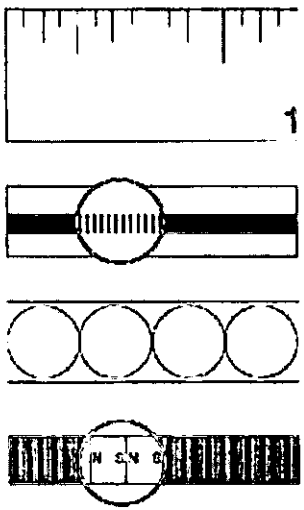


Figura 3.11. Precisión de la escala de cristal.

3.4 Ejemplo de implementación

Para realizar esta tesis, utilizamos una computadora personal con un procesador Pentium MMX a 200MHz, 32Mb de memoria RAM y Windows 98 como sistema operativo. No obstante, el sistema puede ser transportable a diferentes arquitecturas, pero siempre bajo plataformas de 32 bits.

Las tareas a realizar entonces consistieron en lo siguiente:

- Interconectar electrónicamente la MMC con la PC.
- Programar la interface con el usuario para administrar lecturas.
- Programar algoritmos específicos para medición de geometrías típicas y transformación de coordenadas.
- Realizar la programación anterior en un ambiente visual para Windows 98.

Para interconectar la máquina de medición por coordenadas con la PC, utilizamos –como ya se mencionó– la tarjeta PC7266, que fue instalada en la PC ocupando la dirección 300h. Esta tarjeta provee de una alimentación de 5 [V] de CD a los codificadores ópticos, que a su vez transmiten los pulsos eléctricos a los contadores de la tarjeta en cada transición, equivalente a un desplazamiento de 2 [µm] del palpador.

Aunque la tarjeta PC7266 cuenta con cuatro canales de entrada, utilizamos sólo tres de ellos, que se hicieron corresponder con los tres ejes coordenados *x, y, z*, para los cuales esta diseñada la MMC. De esta forma, el desplazamiento en cada uno de los ejes es atendido de forma particular por un canal, asociado a su vez a un contador en la tarjeta. Esto nos permite manejar de forma independiente cada eje coordenado.

Una vez que conectamos la PC con la MMC, debíamos programar la interfaz para poder ver, administrar e interpretar las lecturas que registraban los contadores de la tarjeta. Así mismo, esta interfaz debía ser capaz de controlar las funciones de la propia tarjeta.

Creamos entonces las funciones que debían controlar e inicializar la tarjeta para su correcta operación. Una vez que la tarjeta estuvo lista para operar, para obtener las lecturas de cada eje coordenado, creamos una rutina que consulta los contadores de la tarjeta PC7266 cada ochenta milisegundos. Las lecturas recabadas son actualizadas en los paneles de la barra de estado en que se despliega cada eje; de esta forma, es posible observar la posición en la que se encuentra el palpador mientras éste se desplaza. Para mas detalles acerca de las funciones y clases de la aplicación, refiérase al anexo A.

Algoritmos específicos para la medición de geometrías típicas

Ya que habíamos logrado administrar las lecturas proporcionadas por los contadores de la tarjeta, nuestro siguiente objetivo era que en base a estas lecturas, poder determinar las características de interés en las mediciones de geometrías típicas, tales como el centro y diámetro en el caso de una esfera, el vector normal a un plano en estudio, etcétera.

Las geometrías típicas que comúnmente se miden a través de una máquina de medición por coordenadas y que fueron implementadas en el

software desarrollado en esta tesis son el punto, la línea, el parche triangular, el circulo, el plano, la esfera, el cilindro y el cono.

Línea

Para determinar una línea, son necesarios sólo dos puntos en el espacio. Con los dos puntos palpados, se determina la ecuación vectorial de la recta en el espacio

$$P = Po + t\overline{U}$$

a partir de la cual se obtiene el punto de intersección de ésta con el plano XY y el ángulo mínimo que forma con los planos XZ y YZ.

Para determinar los ángulos que forma la recta con el plano XZ, se obtiene la proyección de la recta sobre este plano y se determina el ángulo que forma la proyección de la recta con el eje Z, tal y como se muestra en la figura 3.12. De igual forma, para el plano YZ se obtiene la proyección de la primera sobre éste último y se determina el ángulo que forma la recta con el eje Z.

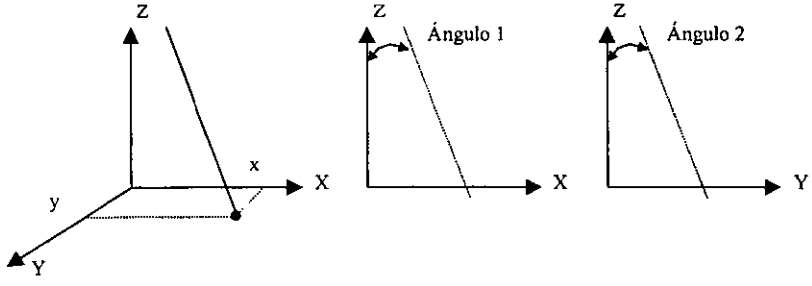


Figura 3.12. Línea.

Plano

Un plano puede ser definido por tres puntos no colineales y puede ser representado por su ecuación cartesiana

$$Ax + By + Cz + D = 0$$

Partiendo de estas definiciones, para determinar las características de un plano, se deben palpar tres puntos no colineales sobre el mismo. Con los

puntos palpados, se calcula el vector normal al plano para determinar las componentes *A*, *B*, *C* y *D* de la ecuación cartesiana del plano.

Con la ecuación cartesiana del plano se determina el eje coordenado con el cual el vector normal forma el mínimo ángulo. A este eje se le denomina *eje de referencia*. Una vez determinado el eje de referencia, se encuentra el punto en el que dicho eje corta al plano en cuestión. Se calculan también los ángulos que forma la proyección del vector normal sobre los planos cartesianos formados por el eje de referencia y los dos ejes coordenados restantes. Para los cálculos refiérase a la figura 3.13.

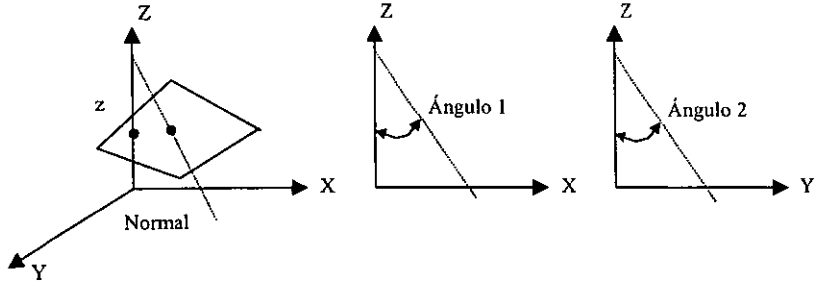


Figura 3.13. Plano.

Es de gran importancia mencionar que en el desarrollo de las operaciones, el algoritmo toma en cuenta el radio de la esfera del palpador, restándolo algebraicamente a los puntos palpados para que las características del palpador no influyan en los resultados finales.

Círculo

Para determinar las características de un círculo en el espacio, es necesario palpar cuatro puntos sobre la circunferencia del mismo. Con los cuatro puntos medidos, y utilizando la ecuación cartesiana de la esfera

$$x^2 + y^2 + z^2 + Gx + Hy + Iz + K = 0$$

se resuelve el sistema de ecuaciones generado, para determinar el diámetro y el centro del círculo. La utilización de la ecuación cartesiana de la esfera para determinar las características del círculo, reside en que una esfera puede formarse al hacer girar un círculo en el espacio sobre su propio eje de simetría, por lo tanto, las características a evaluar aquí,

tales como el diámetro y el centro del círculo, no cambian cuando los cálculos se realizan sobre éste como si fuese una esfera (figura 3.14).

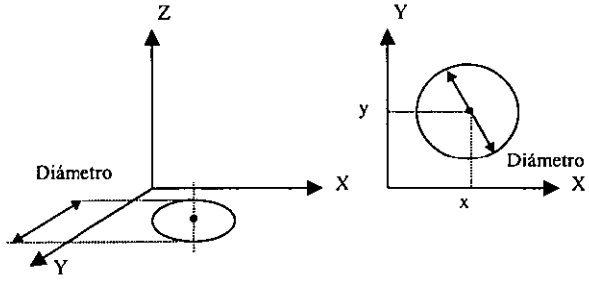


Figura 3.14. Círculo.

Cilindro

En el caso del cilindro circular, las características a determinar son:

- Diámetro.
- Eje de referencia: Es el eje coordenado con el cual el eje del cilindro forma el menor ángulo. Ver figura 3.15.
- Angulos que forma el eje de referencia con la proyección del eje del cilindro sobre los planos formados por el eje de referencia y cada uno de los dos ejes coordenados restantes.

Para determinar éstas características, es necesario que el usuario palpe cuatro puntos del cilindro sobre un plano perpendicular al eje del cilindro. Con éstos puntos, se genera el sistema de ecuaciones a partir del cual se determinan las características antes mencionadas.

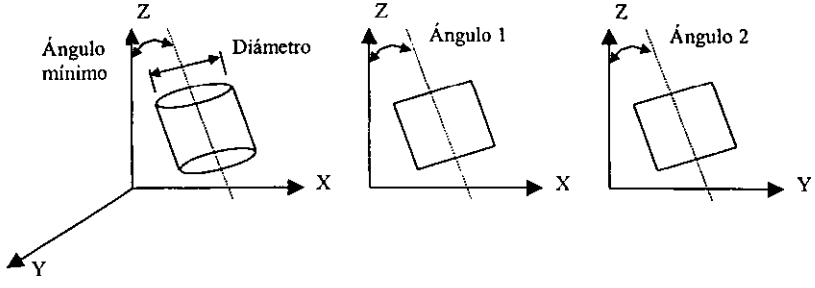


Figura 3.15. Cilindro.

En la figura 3.15, el eje *Z* es el eje coordenado con el cual el eje del cilindro forma el mínimo ángulo, por lo que el eje *Z* se convierte en el eje de referencia. Los ángulos 1 y 2 son entonces los ángulos que forma la proyección del eje del cilindro sobre los planos *XZ* y *YZ* respectivamente con el eje *Z*.

Cono

En el caso del cono, se requiere que cuatro puntos coplanares sean palpados en una sección perpendicular al eje del cono, y otros cuatro puntos coplanares sean palpados en otra sección perpendicular (diferente a la primera) al eje del cono circular. Con los ocho puntos palpados en total, se determina el ángulo de abertura del cono circular, así como el eje de referencia, elegido bajo el mismo criterio que en el cilindro. Se determinan también los ángulos que dicho eje de referencia forma con las proyecciones del eje del cono sobre los planos coordenados formados por el eje de referencia y los ejes coordenados restantes.

Para los cálculos refiérase a la figura 3.16

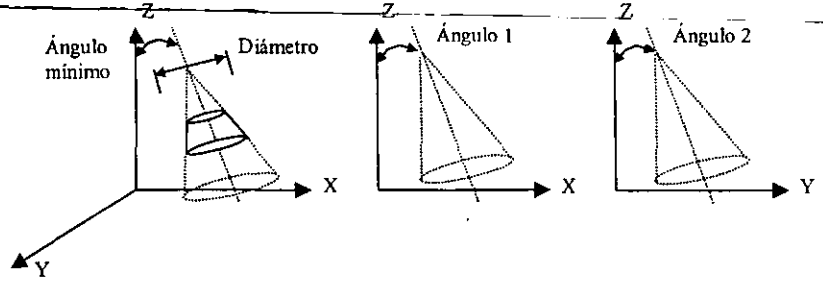


Figura 3.16. Cono.

Esfera

La esfera es una de las formas geométricas más utilizadas en el mundo de la metrología, por lo que es imprescindible que una MMC tenga la capacidad de determinar las características de éstas. Las características más importantes a determinar en una esfera son su diámetro y el centro geométrico de la misma. Para determinarlas, el algoritmo emplea obviamente, la ecuación cartesiana de la esfera

$$x^2 + y^2 + z^2 + Gx + Hy + Iz + K = 0$$

Partiendo de que los valores de *x, y, z* son conocidos; es decir, son determinados por el punto palpado por el usuario, podemos observar que tenemos una ecuación con cuatro incógnitas *G, H, I* y *K*, por lo tanto, para formar un sistema de ecuaciones lineales con solución única, es necesario palpar cuatro puntos, obteniendo con esto un sistema de ecuaciones lineales de 4x4. A partir de este sistema de ecuaciones puede obtenerse el valor de las cuatro incógnitas y por consiguiente, las características del objeto en cuestión (figura 3.17).

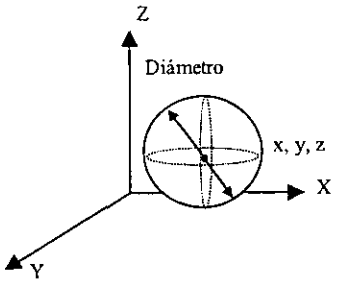


Figura 3.17. Esfera.

Parche triangular

Dentro del campo de la metrología, es también común que el cuerpo en estudio no se corresponda con alguna de las formas geométricas como las mencionadas anteriormente. Para representar estas formas, se recurre a la utilización de *parches triangulares*, que en conjunto y dependiendo de la "resolución" de los mismos, son capaces de representar a la superficie en cuestión.

El algoritmo para determinar un parche triangular es semejante al utilizado para determinar las características de un plano, pero en esta ocasión, la característica relevante es el vector normal al triángulo medido, cuyo sentido puede ser manipulado por el metrologista.

CAPITULO 4

Ambiente de operación para la máquina de medición por coordenadas

Con todos los componentes del sistema acoplados (computadora, tarjeta de interface, codificadores y MMC), logramos automatizar la adquisición de lecturas en su forma más básica. Seguimos implementando el software para agregar algunas herramientas que son necesarias y de gran utilidad en las máquinas de medición por coordenadas. Así se creó el software de *medición por coordenadas* ***MMCCI***, el cual contiene todo el ambiente de operación para la MMC, la adquisición y la interpretación de lecturas. Este software de operación esta compuesto por varios archivos que se mencionan a continuación y que una vez compilados producen el programa aplicación ***MMCCI.EXE***, el cual es el programa más importante para el usuario final.

La aplicación MMCCI.EXE fue programada utilizando Visual C++ Versión 6 como herramienta de desarrollo, por lo que es una aplicación para plataformas de 32 bits; es decir, requiere de Microsoft Windows versión 95, 98 o NT versión 4 para poder correr. También es utilizada la tarjeta PC7266 como interface entre la máquina de medición por coordenadas (MMC) y la PC, la cual cuenta con cuatro contadores independientes, de los cuales son ocupados tres de estos para recibir las

lecturas provenientes de las tres regletas (codificadores ópticos) que representan los tres ejes coordenados en la MMC.

En este capítulo pretendemos dar una descripción muy general del programa, así como proporcionar una guía de operación para el usuario.

4.1 Descripción general del programa

La Microsoft Foundation Class Library & Active Template Library (MFC&T) es realmente la base estándar de programación para aplicaciones Windows. El programa MMCCI.EXE utiliza algunas características de la MFC&T como las siguientes [6]:

- Soporte para documento único.
- Clase de listas, arrays y cadenas.
- Clase relativa al tiempo.
- Clase de objetos visuales como barras y cajas de diálogo.
- Sistema de ayuda.
- Acceso a controles ActiveX.
- Intercambio de datos mediante el portapapeles.

El resto de la aplicación consistió en proporcionar soporte para el control de la tarjeta PC7266 mediante la librería dinámica 32pc7266.dll.

Los archivos necesarios para compilar la aplicación en Visual C++ son los siguientes:

- DialogoCilindro.cpp. Implementa la clase CDialogoCilindro para el procesamiento de la geometría cilindro.
- DialogoCirculo.cpp. Implementa la clase CDialogoCirculo para el procesamiento de la geometría circulo.
- DialogoCono.cpp. Implementa la clase CDialogoCono para el procesamiento de la geometría cono.

- DialogoEjesAuto.cpp. Implementa la clase CDialogoEjesAuto para el procesamiento de la definición de ejes automática.
- DialogoEsfera.cpp. Implementa la clase CDialogoEsfera para el procesamiento de la geometría esfera.
- DialogoLinea.cpp. Implementa la clase CDialogoLinea para el procesamiento de la geometría línea.
- DialogoParche.cpp. Implementa la clase CDialogoParche para el procesamiento de la geometría parche triangular.
- DialogoPlano.cpp. Implementa la clase CDialogoPlano para el procesamiento de la geometría plano.
- DialogoPunto.cpp. Implementa la clase CDialogoPunto para el procesamiento de la geometría punto.
- MainFrm.cpp. Implementa la clase CMainFrame para el registro de las barras de estado y herramientas.
- MmcCI.cpp. Implementa la clase CMmcCIApp para la implementación del comportamiento de la aplicación.
- MmcCIDoc.cpp. Implementa la clase CMmcCIDoc para la serialización de archivos de lecturas (lectura escritura).
- MmcCIView.cpp. Implementa las clases CMmcCIView, CDialogoCargarValor, CDialogoEjes, CdialogoPalpador y CdialogoMatriz para las opciones de medición.
- Msflexgridpto.cpp. Implementa la clase CMSFlexGridPto para el manejo de eventos en el control rejilla ActiveX.
- StdAfx.cpp. Incluye librerías y archivos estándar precompilados.
- DialogoCilindro.h. Incluye prototipos y definiciones para DialogoCilindro.cpp.
- DialogoCirculo.h. Incluye prototipos y definiciones para DialogoCirculo.cpp.
- DialogoCono.h. Incluye prototipos y definiciones para DialogoCono.cpp.

- DialogoEjesAuto.h. Incluye prototipos y definiciones para DialogoEjesAuto.cpp.
- DialogoEsfera.h. Incluye prototipos y definiciones para DialogoEsfera.cpp.
- DialogoLinea.h. Incluye prototipos y definiciones para DialogoLinea.cpp.
- DialogoParche.h. Incluye prototipos y definiciones para DialogoParche.cpp.
- DialogoPlano.h. Incluye prototipos y definiciones para DialogoPlano.cpp.
- DialogoPunto.h. Incluye prototipos y definiciones para DialogoPunto.cpp.
- MainFrm.h. Incluye prototipos y definiciones para MainFrm.cpp.
- MmcCI.h. Incluye prototipos y definiciones para MmcCI.cpp.
- MmcCIDoc.h. Incluye prototipos y definiciones para MmcCIDoc.cpp.
- MmcCIView.h. Incluye prototipos y definiciones para MmcCIView.cpp.
- Msflexgridpto.h. Incluye prototipos y definiciones para Msflexgridpto.cpp.
- Resource.h. Incluye definiciones de constantes.
- StdAfx.h. Incluye prototipos y definiciones para StdAfx.cpp.
- MmcCI.ico. Icono de aplicación.
- MmcCIDoc.ico. Icono de documento.
- AlinearXY.bmp. Bitmap para la caja de diálogo "Definición automática de ejes".
- AlinearZ.bmp. Bitmap para la caja de diálogo "Definición automática de ejes".

- CilindroExt.bmp. Bitmap para la caja de diálogo "Cilindro" (externo).
- CilindroInt.bmp. Bitmap para la caja de diálogo "Cilindro" (interno).
- CirculoExt.bmp. Bitmap para la caja de diálogo "Circulo" (externo).
- CirculoInt.bmp. Bitmap para la caja de diálogo "Circulo" (interno).
- ConoExt.bmp. Bitmap para la caja de diálogo "Cono" (externo).
- ConoInt.bmp. Bitmap para la caja de diálogo "Cono" (interno).
- EsferaExt.bmp. Bitmap para la caja de diálogo "Esfera" (externa).
- EsferaInt.bmp. Bitmap para la caja de diálogo "Esfera" (interna).
- Linea.bmp. Bitmap para la caja de diálogo "Línea".
- Parche.bmp. Bitmap para la caja de diálogo "Parche triangular".
- Plano.bmp. Bitmap para la caja de diálogo "Plano".
- Punto.bmp. Bitmap para la caja de diálogo "Punto".
- Toolbar.bmp. Bitmap para la barra de herramientas.
- MmcCI.rc. Archivo de recursos.
- MmcCI.dws. Archivo de proyecto.
- MmcCI.dsp. Archivo para generar el programa ejecutable.
- MmcCI.clw. Archivo de referencia para las clases.

4.2 Software de operación

El programa "Medición por coordenadas MMCCI.EXE" está diseñado bajo un ambiente amigable para el usuario; Incluye menúes de "persiana", soporte para comunicaciones con la tarjeta de interface entre

codificadores ópticos y PC PC7266, medición de geometrías típicas, definición de ejes, y un área de despliegue de historial de mediciones, figura 4.1.

El programa MMCCI es la interfaz con el usuario para:

- Realizar mediciones de geometrías típicas mediante la MMC.
- Salvar historial de mediciones.
- Recuperar historial de mediciones.
- Definición de ejes (transformación de coordenadas).
- Transporte de mediciones a programas de terceros.

Descripción del programa MMCCI.EXE

El programa principal cuenta con el siguiente menú de opciones:

❖	**Archivo**	Opciones de archivo.
❖	**Editar**	Edición de datos.
❖	**Ver**	Ver herramientas de programa.
❖	**Geometrías**	Medición de geometrías típicas.
❖	**MMC**	Definición de sistemas de coordenadas.
❖	**Calibrar**	Calibración de datos.
❖	**Ayuda**	Opciones de ayuda.

La ventana principal cuenta con una área para el despliegue del historial de mediciones en donde se registran las operaciones realizadas en el programa, figura 4.1. El formato del historial puede salvarse para recuperarse posteriormente. Esencialmente el formato esta construido sobre la base de "primitivas geométricas", en donde se lista el nombre de la primitiva y los parámetros que la definen.

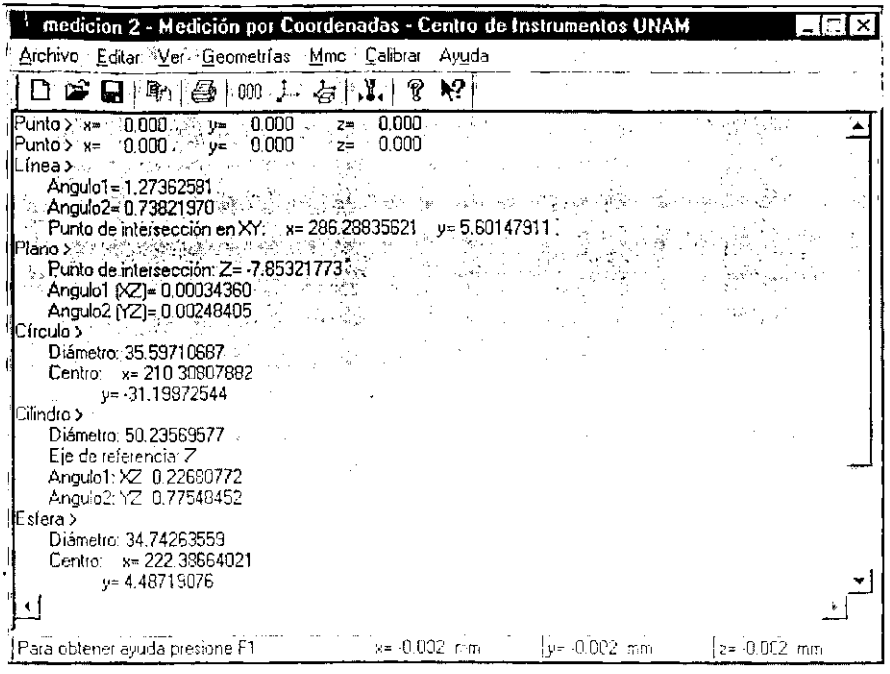


Figura 4.1. "MMCCI". Presentación del historial.

4.2.1. Barra de estado

En la parte baja del programa podemos visualizar una barra de estado con cuatro divisiones mostrando un breve texto de ayuda y las tres coordenadas espaciales con lecturas en tiempo real, figura 4.2.

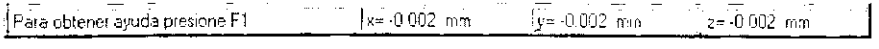


Figura 4.2. Barra de estado.

4.2.2. Barra de herramientas

El programa incluye una barra de herramientas para acceso a las funciones más comunes del programa, figura 4.3.

- Archivo ➢ **Nuevo**
- Archivo ➢ **Abrir**
- Archivo ➢ **Guardar**
- Editar ➢ **Copiar**
- MMC ➢ **Reset xyz**
- MMC ➢ **Definir ejes ➢ Manual**
- MMC ➢ **Definir ejes ➢ Automático**
- Calibrar ➢ **Palpador**
- Ayuda ➢ **Acerca de MMCCI**

Figura 4.3. Barra de herramientas.

4.2.3. Menú *Archivo*

Las opciones del menú *Archivo* permiten la administración estándar de archivos así como las capacidades de impresión, figura 4.4.

- **Nuevo**

Crea un nuevo historial.

- **Abrir**

Abre un historial ya existente.

- **Guardar**

Guarda el historial activo.

- **Guardar como**

Guarda el historial activo con un nuevo nombre.

❖ **Imprimir**

Imprime el historial activo.

❖ **Configurar impresora**

Cambia la impresora y las opciones de impresión.

❖ **Salir**

Sale de la aplicación; pregunta por salvar historial.

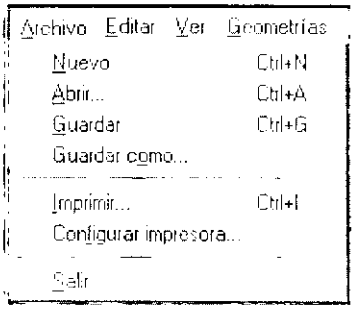


Figura 4.4. Menú *Archivo*.

Otra capacidad de administración de historiales es la asociación de documentos con el *Administrador de Archivos*. De esta manera, es posible buscar un archivo de mediciones *.mmc mediante el *Administrador de Windows* y ejecutarlo desde el mismo administrador. Los archivos *.mmc se indicarán con este símbolo de documento, figura 4.5.

Figura 4.5. Icono de documento *.mmc.

Pulsar dos veces sobre el archivo *.mmc, automáticamente arranca el programa MMCCI. De otra forma, marcar y arrastrar el archivo *.mmc a la aplicación MMCCI ya abierta.

4.2.4. Menú *Editar*

La única opción del menú *Editar* permite copiar al portapapeles la selección actual del historial de mediciones, figura 4.6.

Figura 4.6. Menú *Editar*.

De esta forma resulta sencillo transferir las lecturas a aplicaciones de terceros, como por ejemplo Word, Excel o programas de diseño gráfico.

4.2.5. Menú *Ver*

Las opciones del menú *Ver* permiten ocultar o mostrar los elementos de la ventana principal del programa MMCCI, figura 4.7.

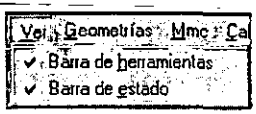


Figura 4.7. Menú *Ver*.

❖ **Barra de herramientas**

Muestra u oculta la barra de herramientas.

❖ **Barra de estado**

Muestra u oculta la barra de estado.

La Barra de herramientas muestra botones para el acceso simplificado a funciones comunes descritas en 4.2.2.

La barra de estado cuenta con cuatro divisiones mostrando la información descrita en 4.2.1.

4.2.6. Menú *Geometrías*

Las opciones del menú *Geometrías* permiten medir geometrías típicas, como lo muestra la figura 4.8.

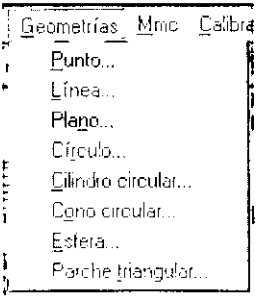


Figura 4.8. Menú *Geometrías*.

❖ Punto

Despliega la caja de diálogo "Punto", figura 4.9, en donde se pueden capturar posiciones espaciales al seleccionar *Agregar*. Al presionar *Aceptar*, el historial despliega la primitiva *Punto* y las coordenadas *x*, *y* y *z*.

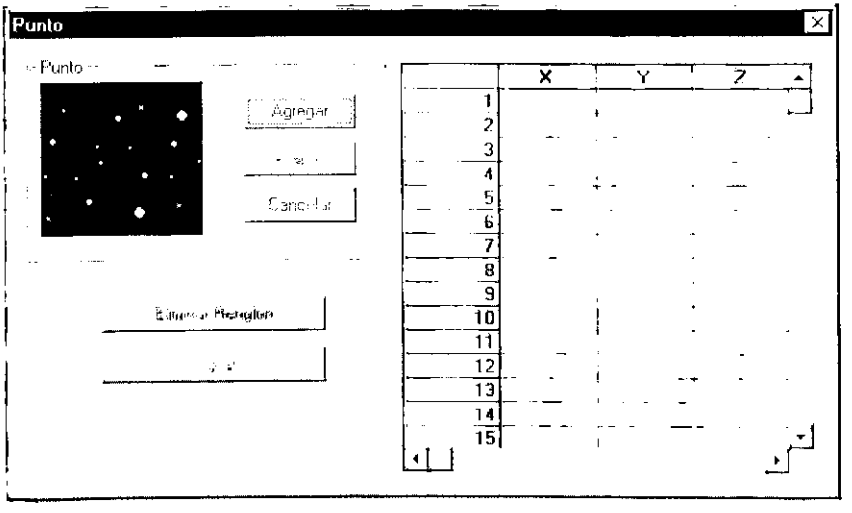


Figura 4.9. Caja de diálogo "Punto".

❖ Linea

Despliega la caja de diálogo "Línea", figura 4.10, en donde se ingresan las dos posiciones espaciales que definen la recta en el espacio, al seleccionar *Agregar*. Al presionar *Aceptar*, el historial despliega la primitiva *Línea*, el ángulo de su proyección con el plano *XZ*, el ángulo de su proyección con el plano *YZ* y el punto de intersección en el plano *XY*.

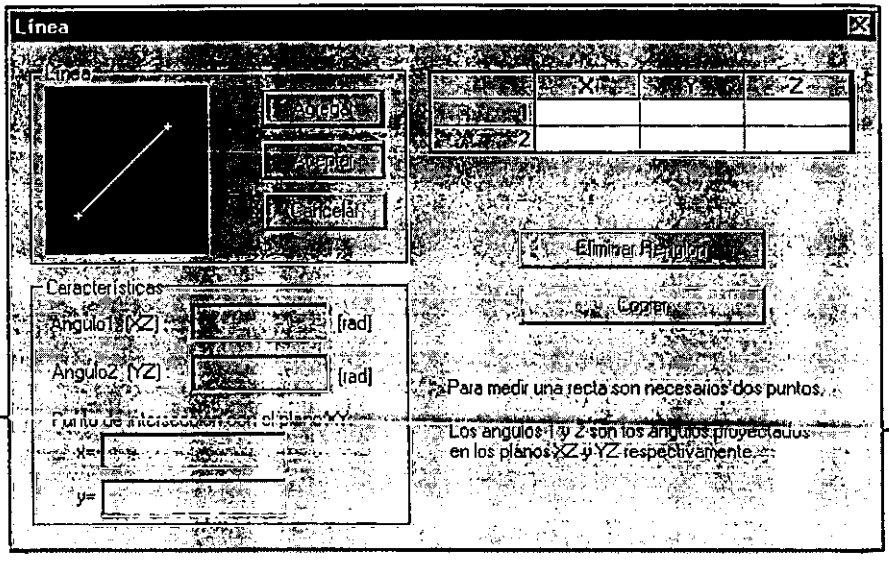


Figura 4.10. Caja de diálogo "Línea".

❖ Plano

Despliega la caja de diálogo "Plano", figura 4.11, en donde se ingresan las tres posiciones espaciales que definen el plano en el espacio, al seleccionar *Agregar*. El sentido de palpación define el sentido del vector normal al plano, según la regla de la mano derecha. Al presionar *Aceptar*, el historial despliega la primitiva *Plano*, la coordenada de intersección con el eje *Z*, el ángulo de su proyección con el plano *XZ* y el ángulo de su proyección con el plano *YZ*.

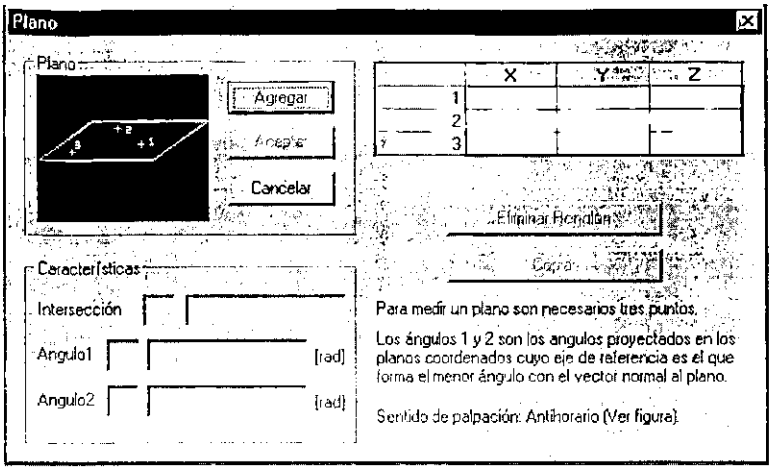


Figura 4.11. Caja de diálogo "Plano".

❖ Circulo

Despliega la caja de diálogo "Círculo", figura 4.12, en donde se ingresan las tres posiciones espaciales que definen el círculo en el plano, al seleccionar *Agregar*. La caja de diálogo cuenta con la opción de medir círculos internos (huecos) o círculos externos (sólidos). Al presionar *Aceptar*, el historial despliega la primitiva *Circulo*, el diámetro, y las coordenadas de su centro.

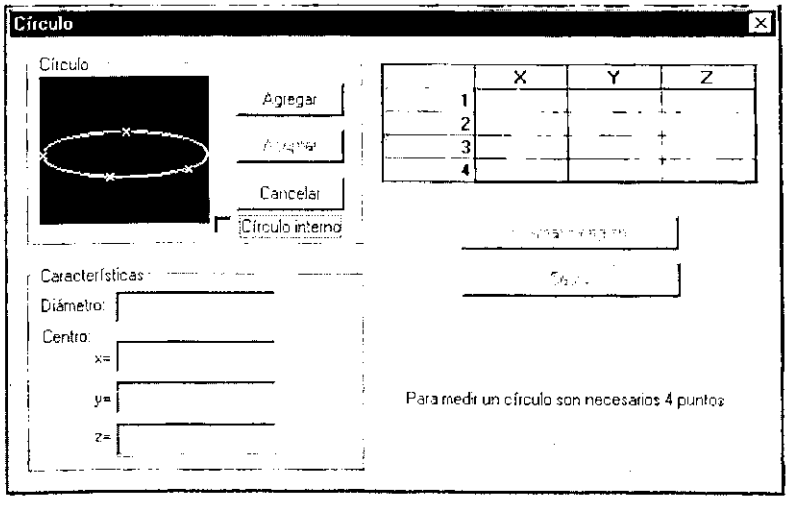


Figura 4.12. Caja de diálogo "Circulo".

❖ Cilindro circular

Despliega la caja de diálogo "Cilindro", figura 4.13, en donde se ingresan las cuatro posiciones espaciales que definen el cilindro, al seleccionar *Agregar*. La caja de diálogo cuenta con la opción de medir cilindros internos (huecos) o cilindros externos (sólidos). Al presionar *Aceptar*, el historial despliega la primitiva *Cilindro*, el diámetro, el ángulo de su proyección con el plano *XZ* y el ángulo de su proyección con el plano *YZ*.

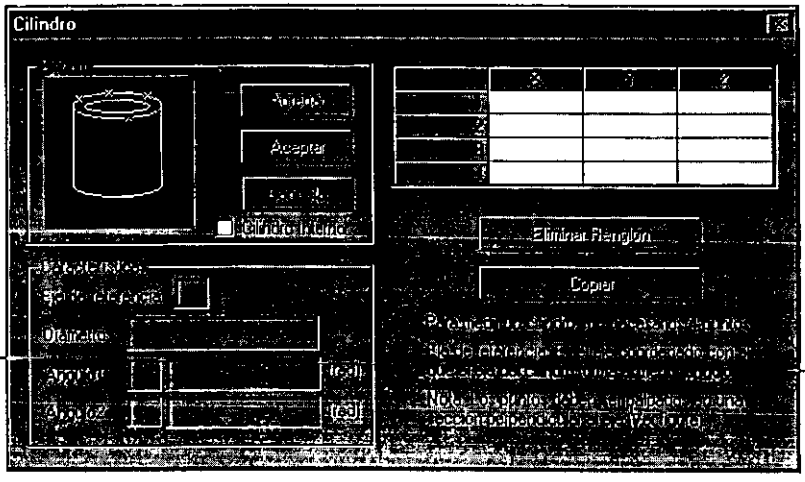


Figura 4.13. Caja de diálogo "Cilindro".

❖ Cono circular

Despliega la caja de diálogo "Cono", figura 4.14, en donde se ingresan los ocho puntos espaciales para definir al cono; cada uno de estos es incluido al hacer click sobre el botón *Agregar*. La caja de diálogo tiene la opción para medir conos internos (huecos) y externos (sólidos). Una vez palpados los ocho puntos, al presionar el botón *Aceptar*, el historial despliega la primitiva *Cono*, el eje de referencia, el ángulo de abertura y los ángulos de proyección con los planos coordenados formados por el eje de referencia y los dos ejes restantes.

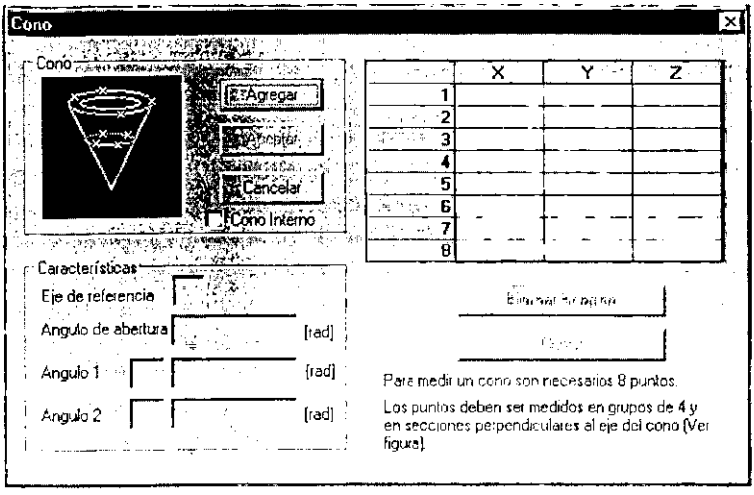


Figura 4.14. Caja de diálogo "Cono".

❖ Esfera

Despliega la caja de diálogo "Esfera", figura 4.15, en donde se ingresan las cuatro posiciones espaciales que definen la esfera en el espacio, al seleccionar *Agregar*. La caja de diálogo cuenta con la opción de medir esferas internas (huecas) o esferas externas (sólidas). Al presionar *Aceptar*, el historial despliega la primitiva *Esfera*, el diámetro y la posición espacial de su centro.

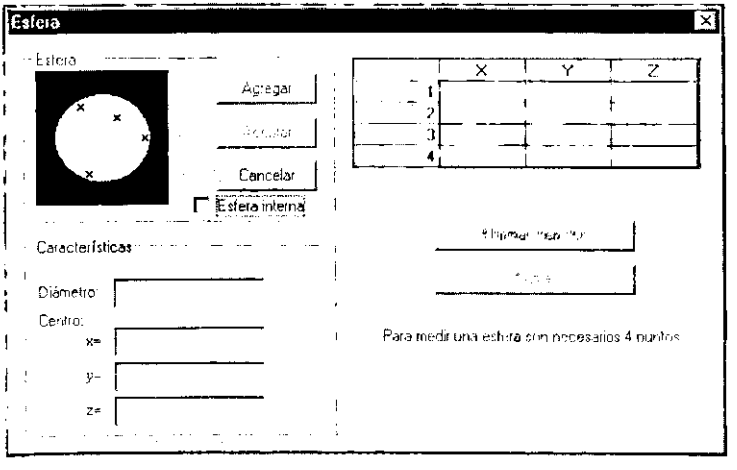


Figura 4.15. Caja de diálogo "Esfera".

❖ **Parche triangular**

Despliega la caja de diálogo "Parche Triangular", figura 4.16, en donde se ingresan los tres vértices espaciales que definen el parche triangular en el espacio al seleccionar *Agregar*. El sentido de palpación define el sentido del vector normal al plano, según la regla de la mano derecha. Al presionar *Aceptar*, el historial despliega la primitiva *Parche triangular*, las tres coordenadas espaciales de los vértices y los tres vectores normales a cada vértice.

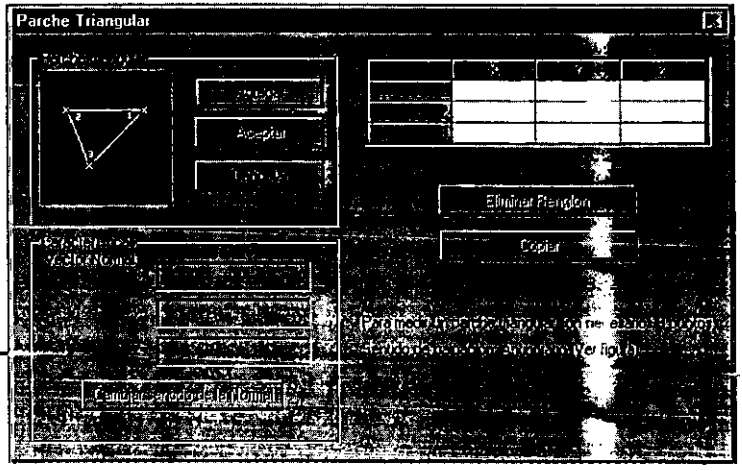


Figura 4.16. Caja de diálogo "Parche Triangular".

4.2.7. Menú *MMC*

Las opciones del menú *MMC* permiten definir el sistema de coordenadas a utilizar, como lo muestra la figura 4.17.

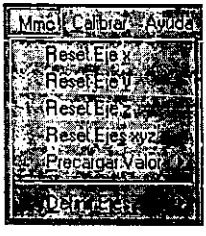


Figura 4.17. Menú *MMC*.

❖ **Reset Eje x**

Establece a cero la lectura del eje *x*.

❖ **Reset Eje y**

Establece a cero la lectura del eje *y*.

❖ **Reset Eje z**

Establece a cero la lectura del eje *z*.

❖ **Reset Ejes xyz**

Establece a cero la lectura de los ejes *xyz* simultáneamente.

❖ **Precargar Valor/Eje x**

Despliega la caja de diálogo "Valor a cargar", figura 4.18, en donde se introduce el valor de sobrecarga a la lectura del eje *x*.

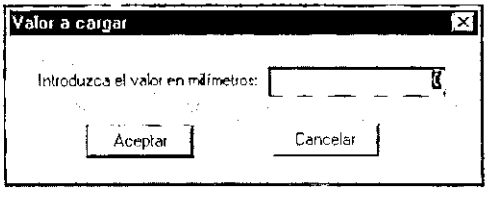


Figura 4.18. Caja de diálogo "Valor a cargar".

❖ **Precargar Valor/Eje y**

Despliega la caja de diálogo de la figura 4.18, en donde se introduce el valor de sobrecarga a la lectura del eje *y*.

❖ **Precargar Valor/Eje z**

Despliega la caja de diálogo de la figura 4.18, en donde se introduce el valor de sobrecarga a la lectura del eje *z*.

❖ **Definir Ejes/Manual**

Despliega la caja de diálogo "Definición manual de ejes", figura 4.19, en donde se introducen las traslaciones y rotaciones en los ejes coordenados.

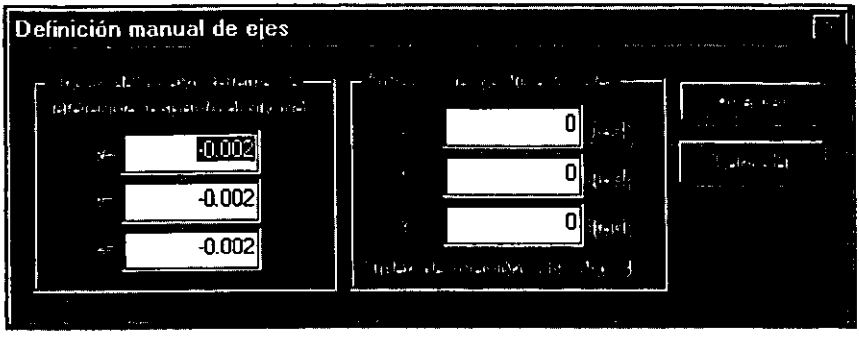


Figura 4.19. Caja de diálogo "Definición manual de ejes".

❖ **Definir Ejes/Automático**

Despliega la caja de diálogo "Definición automática de ejes", figura 4.20, en donde se define el nuevo sistema de coordenadas mediante los tres pasos siguientes.

1. Origen del nuevo sistema de referencia mediante la intersección de tres planos. Se miden tres planos que, mediante el punto de su intersección, definen el nuevo origen.

2. Alineación con respecto al eje *Z*. Se mide un plano para que su normal defina la orientación del nuevo sistema con respecto al eje *Z*.

3. Alineación con respecto a los ejes *XY*. Se mide un plano que define la orientación del nuevo sistema con respecto al plano *XY*.

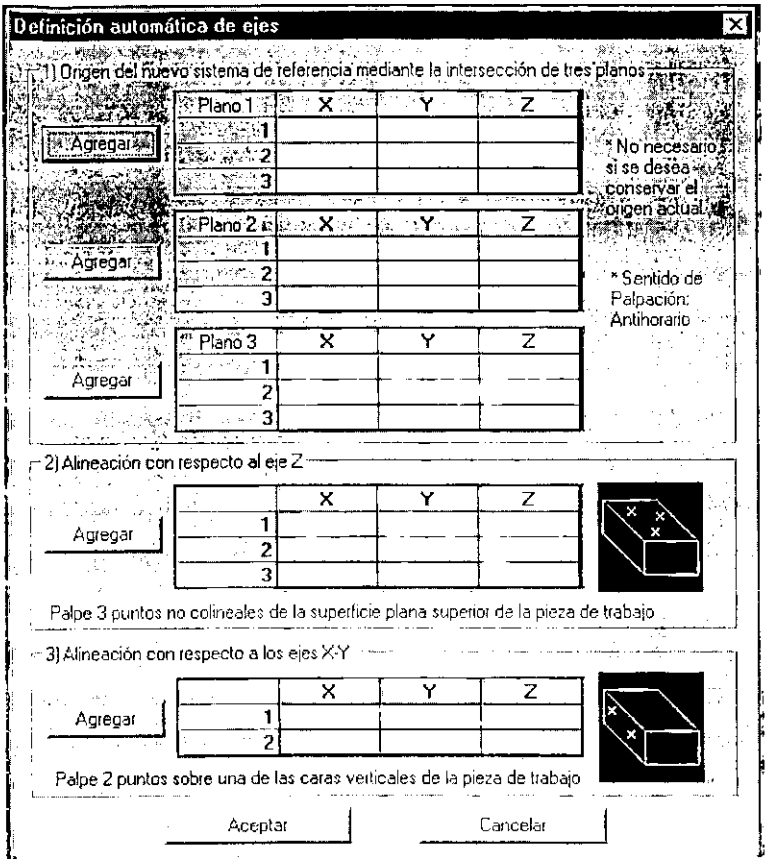


Figura 4.20. Caja de diálogo "Definición automática de ejes".

❖ **Definir Ejes/Regresar**

Deshace la definición de ejes previa y restablece la anterior.

4.2.8. Menú *Calibrar*

Las opciones del menú *Calibrar* permiten establecer las opciones de calibración, como lo muestra la figura 4.21.

Figura 4.21. Menú *Calibrar*.

❖ **Palpador**

Despliega la caja de diálogo "Calibración del palpador", figura 4.22, en donde el usuario introduce el diámetro de la esfera de contacto del palpador.

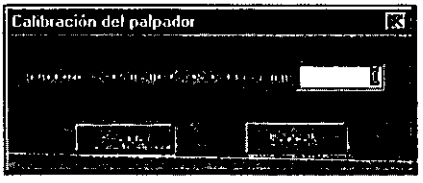


Figura 4.22. Caja de diálogo "Calibración del palpador".

❖ **Matriz**

Despliega la caja de diálogo "Matriz de calibración", figura 4.23, en donde el usuario altera la matriz de filtrado a través de la cual pasan las lecturas de los codificadores ópticos, es decir

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [T] \begin{bmatrix} lx \\ ly \\ lz \end{bmatrix}$$

en donde $[x, y, z]^T$ son las lecturas de posición desplegadas en la barra de estado, $[T]$ es la matriz de calibración y $[lx, ly, lz]^T$ son las lecturas de los codificadores de posición. Observe que el valor por omisión $[T]$=*Matriz identidad*, no afecta las lecturas de los codificadores.

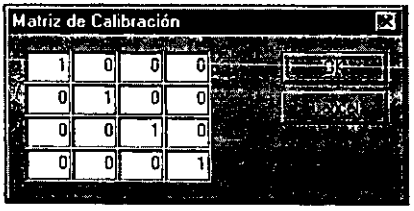


Figura 4.23. Caja de diálogo "Matriz de Calibración".

4.2.9. Menú *Ayuda*

Despliega el sistema de ayuda, similar al texto de éste capítulo, en el formato estándar de Windows.

CAPITULO 5

Resultados y conclusiones

En el presente capítulo hacemos una descripción de los resultados obtenidos con el desarrollo de ésta tesis, así como una evaluación de nuestro sistema de medición en tres ejes y el trabajo a futuro sobre el mismo; ya que desde el inicio del proyecto se estableció que el sistema debía contemplar posibles expansiones y/o mejoras, tales como la automatización del movimiento sobre los ejes coordenados x, y y z, y la graficación de algunas geometrías, entre otros.

5.1 Resultados

Finalmente, con el trabajo realizado durante todo el proyecto, hemos construido un sistema de medición en tres dimensiones totalmente desarrollado en el Centro de Instrumentos de la UNAM. Este sistema consta de una máquina de medición por coordenadas (MMC) construida anteriormente en la misma dependencia, una computadora personal y una tarjeta de interface entre codificadores ópticos y la PC. El sistema en conjunto es capaz de medir en tres ejes sobre volúmenes de 0.5 m^3 con resolución mínima de 2 μm y con las ventajas tradicionales de medir que ofrecen los equipos de MMC comerciales.

El sistema implementado se muestra en las figuras 5.1 y 5.2.

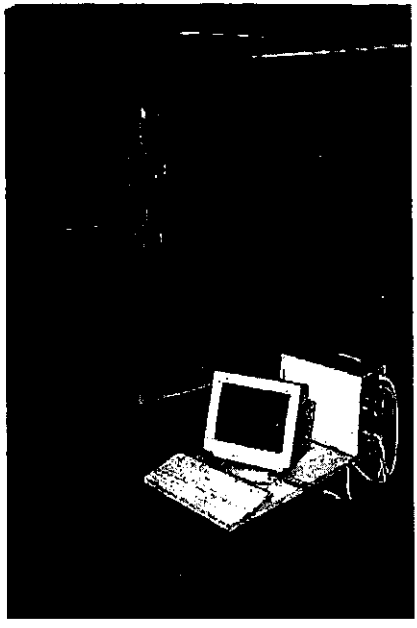


Figura 5.1. Sistema de medición.

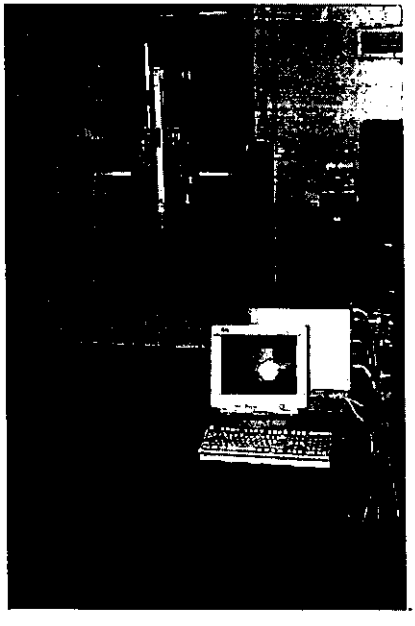


Figura 5.2. Sistema de medición.

Las figuras 5.1 y 5.2 muestran una PC conectada a los tres codificadores ópticos en la MMC, fijados de tal forma que se hacen coincidir con un sistema de ejes cartesianos x, y, z. El sistema completo es capaz de medir dimensionalmente en tres ejes con la asistencia del software de operación.

Un pequeño ejemplo de la funcionalidad del sistema es el siguiente listado, que resulta de copiar el historial de una de las mediciones realizadas al portapapeles, usando el programa MMCCI.EXE y posteriormente pegarlo en éste documento Word. No obstante, el programa guarda los historiales de medición en archivos propios (*.mmc).

```
Punto > x=    0.000      y=    0.000      z=    0.000
Punto > x=    0.000      y=    0.000      z=    0.000
Línea >
     Angulo1= 1.27362581
     Angulo2= 0.73821970
     Punto de intersección en XY:   x= 286.28835621   y= 5.60147911
Plano >
     Punto de intersección: Z= -7.85321773
     Angulo1 (XZ)= 0.00034360
     Angulo2 (YZ)= 0.00248405
Círculo >
     Diámetro: 35.59710687
     Centro:   x= 210.30807882
               y= -31.19872544
Cilindro >
     Diámetro: 50.23569577
     Eje de referencia: Z
     Angulo1: XZ  0.22680772
     Angulo2: YZ  0.77548452
Esfera >
     Diámetro: 34.74263559
     Centro:   x= 222.38664021
               y= 4.48719076
               z= 34.83576211
Parche triangular >
     P1  (  200.186,  17.870,  35.250)       N  (  -0.30890000,  -1.06587200,
157.86567600)
     P2  (  212.208,  6.388,  35.196)        N  (  -0.30890000,  -1.06587200,
157.86567600)
     P3  (  214.136,  17.678,  35.276)       N  (  -0.30890000,  -1.06587200,
157.86567600)
Punto > x=  -20.334      y=   -0.048      z=    0.000
```

5.2 Conclusiones

Cuando planteamos el software de operación tomamos en consideración que el usuario es un técnico en el área de metrología dimensional y ha tomado algún tipo de entrenamiento en el uso de MMC's. No obstante de la complejidad de los sistemas MMC's, el enfoque amigable y basado en Windows del software de operación, reduce la curva de aprendizaje del mismo. De esta manera, estamos seguros que los operadores sin capacitación en la MMC del Centro de Instrumentos UNAM serán capaces de familiarizarse por sí mismos con el software de operación mediante la ayuda en línea.

Por otra parte, el software desarrollado resulta ser una interface amigable con el usuario. Adicionalmente, el software contempla intercambio de datos mediante el "portapapeles" de manera que estos puedan ser procesados por programas de terceros (Auto Cad, Excel, Matlab) o programas de propósito específico.

Otro punto importante es la capacidad del software para la medición de geometrías típicas, a partir de la interpretación de las lecturas provenientes de los codificadores ópticos. El software desarrollado cuenta también con una matriz de calibración para la corrección de errores en las mediciones, debido a imperfecciones en la máquina de medición por coordenadas.

5.3 Evaluación

Las MMC's han revolucionado el mundo de la metrología dimensional y se han vuelto parte integral de los sistemas de calidad industriales, resultando ello en bajos costos de inspección y un incremento en la productividad. Desde que las MMC's estuvieron disponibles hace 30 años, el sector productivo y los usuarios han expresado el deseo de evaluar comprensivamente su desempeño. Estas máquinas deben verificarse durante su instalación y periódicamente durante su operación. Por lo tanto el desarrollo de técnicas eficientes y exactas se ha vuelto recientemente una prioridad. No obstante, tales técnicas evalúan en conjunto la operación de una MMC considerando como un sistema único tanto software como hardware. En nuestro caso, deseamos evaluar

aisladamente el software de operación, no obstante que algunas de sus bondades como presentación y sencillez de operación han sido evaluadas subjetivamente. En cuanto a una evaluación cualitativa más rigurosa, una primera evaluación del desempeño particular del software es contrastar aisladamente los resultados numéricos que proporciona el programa contra cálculos numéricos o contra objetos patrones.

En una de las primeras evaluaciones, tomamos las mediciones de una esfera patrón, comparando el valor obtenido contra el valor nominal. Este proceso se muestra en la figura 5.3.

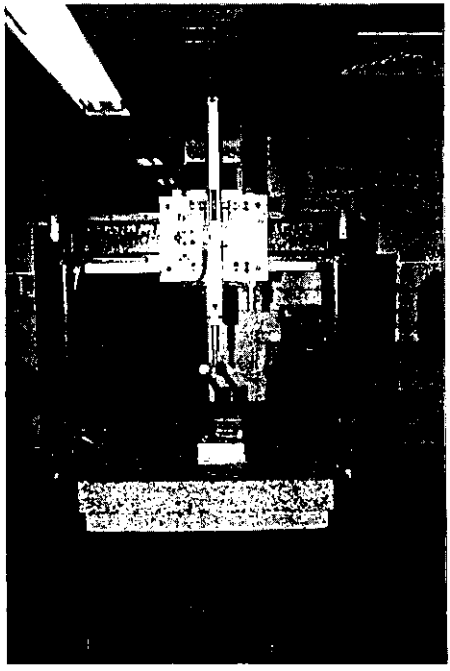


Figura 5.3. Evaluación del sistema de medición.

La tabla 1 muestra el valor obtenido por nuestro sistema de medición y el valor nominal de la esfera.

Valor nominal (mm)	Valor medido (mm)	Error (mm)
30	30.012	0.012

Tabla 5.1. Medición de una esfera patrón.

Sin duda alguna el resultado de la medición no sólo depende de la precisión numérica con que se realizan los cálculos en el interior del software. Otro tipo de errores son acarreados debido a la imperfección en la construcción de la MMC y al registro de la calibración en la esfera patrón. En este sentido, nuestro propósito aquí es proporcionar una primera aproximación a la evaluación del software.

Tratando de realizar una evaluación de la interface programada, en forma aislada de la máquina de medición por coordenadas, podríamos evaluar el software en los siguientes rubros: exactitud en los cálculos, eficiencia en la medición de geometrías, aprovechamiento de los recursos de la PC, facilidades para la reutilización de código, intercambio de datos con otras aplicaciones, así como las ventajas y desventajas de esta interface frente a otras.

Exactitud en los cálculos

La exactitud con que se realizan los cálculos es muy alta, ya que los tipos de datos empleados en los cálculos son flotantes de doble precisión, lo cual nos permite manejar inclusive una resolución mayor a la de los codificadores ópticos. Así mismo. A pesar de esto, no debemos pasar por alto que a una resolución de 2μm la fuerza aplicada por el palpador sobre el punto medido implica una deformación del cuerpo en medición, lo cual causa errores en la medición, los cuales hasta el momento son imposibles de eliminar.

Eficiencia en la medición de geometrías

En la medición de geometrías la aplicación presenta una eficiencia aceptable, ya que con pasos muy simples realizados por el operador, el software determina las características del cuerpo geométrico medido, aunque al igual que la mayoría del software de operación de las máquinas de medición por coordenadas comerciales, existen ciertas restricciones al momento de palpar los puntos sobre el cuerpo geométrico. Por ejemplo, al medir un cilindro circular, las dos secciones circulares palpadas deben ser transversales al eje del cilindro, y en el caso del parche triangular, el sentido de palpación debe ser horario, lo cual nos permite conocer el sentido del vector normal al triángulo.

Aprovechamiento de los recursos de la PC

El aprovechamiento de los recursos de la PC es óptimo, ya que al utilizar la tarjeta PC7266 como interface entre los codificadores y la PC, nos permite liberar al procesador de la tarea que significaría monitorear directamente los codificadores ópticos, lo cual implicaría –en caso de ser posible la implementación–, tener el procesador completamente dedicado a esta tarea. En cambio, con el método empleado podemos tener otros procesos corriendo en la misma PC sin afectar las mediciones y sin perder cuentas.

En segundo plano, gracias a la abstracción de datos de C++, el número de líneas escritas en el código de programación se reduce considerablemente, ocupando el menor espacio posible. A decir verdad, el tamaño en disco de la aplicación (archivo ejecutable) es de 475183 bytes.

En tercer plano, debido a que los historiales de medición son escritos en formato texto, el espacio que ocupan en disco es muy pequeño, en comparación con algunos otros formatos como por ejemplo documentos.

Facilidades para la reutilización de código

Gracias a las características propias de Visual C++, y en particular a la abstracción de datos, es posible agregar nuevas características, módulos o geometrías al programa, sin la necesidad de modificar el código ya existente, o si es el caso, con una mínima modificación, con lo cual es posible realizar el trabajo proyectado a futuro sobre la máquina de medición por coordenadas.

Intercambio de datos con otras aplicaciones

El programa presenta la opción de intercambio de datos con otras aplicaciones mediante el portapapeles de Windows, lo que permite exportar los datos a otras aplicaciones para la realizar otro tipo de cálculos o introducir los datos en programas de terceros. Además, gracias al formato texto de los archivos de medición generados por el programa, éstos presentan una gran flexibilidad para que puedan ser utilizados por otros programas, así mismo, el formato de presentación de datos en estos archivos puede ser modificado con gran facilidad modificando sólo una mínima parte de código.

Ventajas y desventajas de la interface frente a otras

Las principales ventajas de esta interface frente a las interfaces de las MMC comerciales, es la facilidad de operación que ésta presenta para el operador en la medición de las geometrías. Otra característica importante de resaltar, es el empleo de la matriz de corrección para contrarrestar los errores ocasionados por el deterioro en la máquina de medición por coordenadas. Y finalmente, la baja inversión que se requiere para obtener la interface desarrollada.

Por otra parte, una de las principales desventajas de esta interface, puede ser tal vez la inexistencia de un equipo de trabajo dedicado exclusivamente al mejoramiento y consecutivo desarrollo de la interface, como lo hacen las empresas del ramo de la medición por coordenadas.

En general, podemos calificar la interface como un programa que compite en exactitud y eficiencia con un equipo comercial, superando incluso a algunos de estos en sencillez de operación.

Posteriormente, se pueden realizar evaluaciones en el ámbito de software de mayor complejidad, tratando de evaluar aisladamente los cálculos numéricos.

5.4 Trabajo a futuro

Finalmente, aunque la MMC y su software han demostrado su exactitud y utilidad, existe trabajo a futuro enfocado a evaluar su desempeño periódico y automatizar el movimiento en los tres ejes. Para la evaluación del sistema es necesario hacer una calibración a la MMC y determinar los componentes de la matriz de calibración del programa ***MMCCI***, para posteriormente someter al sistema en conjunto a nuevas evaluaciones mediante la medición de objetos patrón más avanzados, figura 5.4.

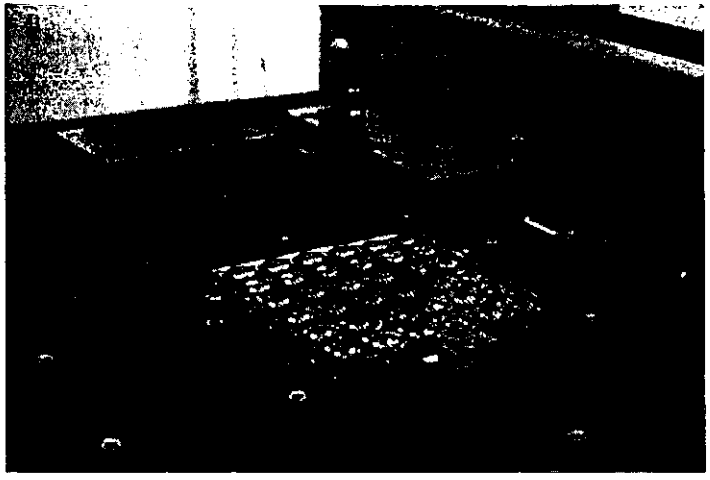


Figura 5.4. Objetos patrón para la calibración de MMC.

Las limitaciones actuales nos impiden desarrollar algoritmos específicos para un palpado automático. En un futuro y con la automatización en los tres ejes de la MMC, pensamos desarrollar algoritmos de palpado mediante contacto poco susceptibles a las deformaciones propias del contacto mecánico. Esta idea aún se encuentra en su fase de planeación, pero podemos adelantar que algunas de las opciones analizadas son la utilización de pequeñas cámaras de video, o el empleo de un haz luminoso muy fino. Ver figura 5.5 y 5.6.

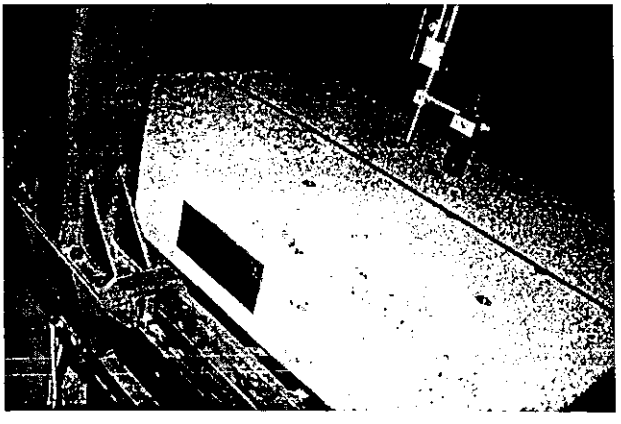


Figura 5.5. Método de palpado poco susceptible al contacto mecánico.

Figura 5.6. Algoritmos de palpado poco susceptibles al contacto mecánico

Existe también la posibilidad de agregar algoritmos para la medición de geometrías más complicadas, como las mostradas en las figuras 5.7 y 5.8. Tales formas no se adaptan a las geometrías básicas o son composiciones de varias de éstas; lo cual dificulta su modelado matemático y en consecuencia, la determinación de algunas de sus características más importantes.

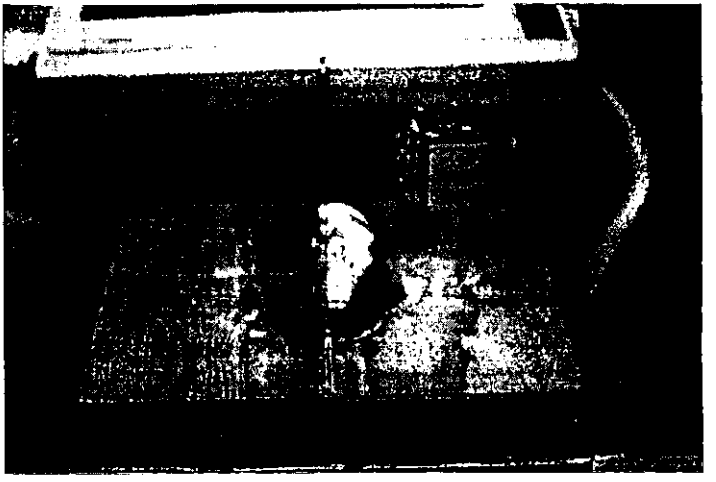


Figura 5.7. Medición de geometrías complicadas.

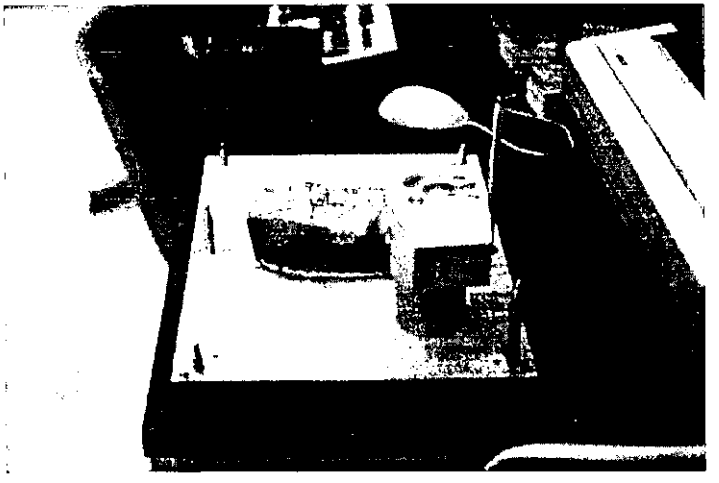


Figura 5.8. Medición de geometrías complicadas.

En este sentido, el programa ***MMCCI*** desarrollado tiene la opción de medir parches triangulares; mediante los cuales y con trabajo posterior a esta tesis, será posible definir las características de estas geometrías en su totalidad. De este modo, la precisión de la forma estará determinada por la resolución de cada uno de los parches medidos. Aunado a esto, está contemplada la posibilidad de exportar a programas de terceros, el conjunto de parches triangulares que forman la geometría medida para su representación gráfica. Ver figura 5.9.

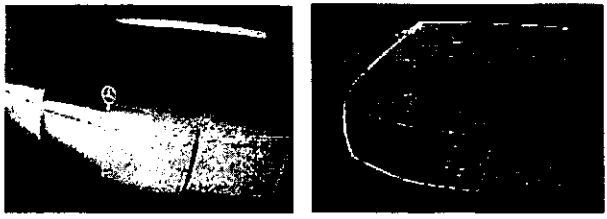


Figura 5.9. Representación gráfica de geometrías complicadas.

Como puede apreciarse, aún existe el suficiente trabajo a futuro para desarrollar otra tesis, pero lo más importante es que el primer paso está dado; y nuestro proyecto, desarrollado bajo el nombre de ésta tesis ***Programación de la Interface entre una Máquina de Medición por Coordenadas y una PC*** está trabajando, cumpliendo con las expectativas que en ésta se habían planteado.

ANEXO A

Programación Visual C++

Debido a que el código fuente es muy extenso, en este apartado solamente nos concretaremos a describir las clases, ya que los archivos que conforman la aplicación han sido descritos en el capítulo 4.

Descripción de clases

La aplicación ***MmcCI*** consta de 19 clases, mostradas en la figura A.1. Cada una de estas clases contienen varias funciones miembro, entre las que se encuentran el constructor y destructor de la clase, funciones para el manejo de eventos y otras mas para cálculos internos.

Figura A.1. Clases de la aplicación MmcCl.

A continuación se describen cada una de las clases mostradas en la figura A.1 con las funciones miembro que las componen.

CAboutDlg

Esta clase describe al cuadro de diálogo Acerca de..., que aparece al hacer clic sobre la opción Acerca de... del menú ayuda. La figura A.2 muestra las funciones miembro que componen esta clase, descritas a continuación.

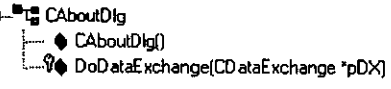


Figura A.2. Funciones miembro de la clase CAboutDlg.

CAboutDlg(); Constructor estándar de la caja de diálogo Acerca de MMCCI.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

CDialogoCargarValor

Esta clase define el cuadro de diálogo que es utilizado para precargar una valor a los contadores de la tarjeta PC7266.Esta clase contiene las funciones miembro que se muestran en la figura A.3.

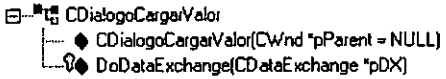


Figura A.3. Funciones miembro de la clase CDialogoCargarValor.

CDialogoCargarValor(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Valor a cargar.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones). Esta caja de diálogo se utiliza para cargar un valor a los ejes coordenados. Limita el rango numérico de entrada.

CDialogoCilindro

Esta clase fue creada para la medición de la geometría cilindro y contiene las funciones miembro necesarias para realizar y procesar las mediciones sobre el cuerpo geométrico. Las funciones miembro de esta clase se muestran en la figura A.4.

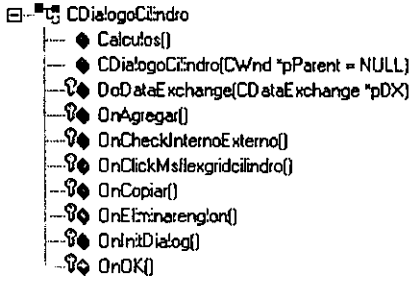


Figura A.4. Funciones miembro de la clase CDialogoCilindro.

CDialogoCilindro(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Cilindro.

void Calculos(); Ajusta los puntos palpados a la ecuación del cilindro. Para los cálculos refiérase a la figura 3.15.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo, agrega datos de la PC7266 a la rejilla.

afx_msg void OnClickMsflexgridcilindro(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Cilindro.

afx_msg void OnCheckInternoExterno(); Maneja el evento pulsar sobre la caja de verificación Cilindro interno. Actualiza el mapa de bits en la caja de diálogo que muestra la geometría interna o externa.

CDialogoCirculo

Esta clase es utilizada para realizar las mediciones y operaciones necesarias en la medición de círculos. En la figura A.5 se muestran las funciones miembro de la clase CDialogoCirculo.

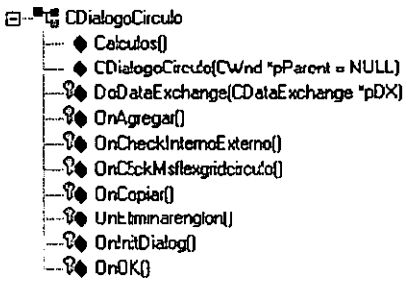


Figura A.5. Funciones miembro de la clase CDialogoCirculo.

CDialogoCirculo(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Circulo.

void Calculos(); Ajusta los puntos palpados a la ecuación del círculo. Para los cálculos refiérase a la figura 3.14.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo, agrega datos de la PC7266 a la rejilla.

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnClickMsflexgridcirculo(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Círculo.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

afx_msg void OnCheckInternoExterno(); Maneja el evento pulsar sobre la caja de verificación Círculo interno. Actualiza el mapa de bits en la caja de diálogo que muestra la geometría interna o externa.

CDialogoCono

Los datos y las funciones necesarias para obtener las características y realizar la medición del cono circular están contenidas en esta clase. La clase CDialogoCono contiene las funciones miembro que se muestran en la figura A.6.

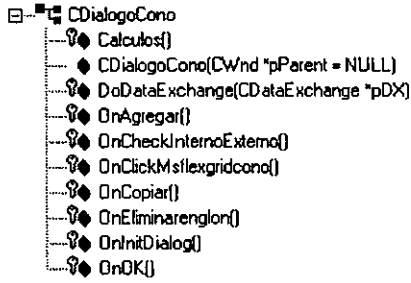


Figura A.6. Funciones miembro de la clase CDialogoCono.

CDialogoCono(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Cono.

void Calculos(); Ajusta los puntos palpados a la ecuación del cono. Para los cálculos refiérase a la figura 3.16.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo, agrega datos de la PC7266 a la rejilla.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Cono.

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnClickMsflexgridcono(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

afx_msg void OnCheckInternoExterno(); Maneja el evento pulsar sobre la caja de verificación Cono interno. Actualiza el mapa de bits en la caja de diálogo que muestra la geometría interna o externa.

CDialogoEjes

Esta clase agrupa los datos y controla el comportamiento del cuadro de dialogo que se emplea para introducir los datos de rotación y traslación de ejes, indicando de forma específica los ángulos de rotación y el nuevo origen el sistema de coordenadas. Esta clase contiene las funciones miembro que muestra la figura A.7.

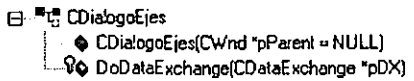


Figura A.7. Funciones miembro de la clase CDialogoEjes.

CDialogoEjes(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Definición manual de ejes.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones). Esta caja de diálogo recibe el ángulo en radianes que deberá rotarse cada eje coordenado. Limita el rango numérico de entrada.

CDialogoEjesAuto

Esta clase se encarga del cuadro de diálogo utilizado para realizar la rotación y traslación automática de los ejes coordenados. Las funciones miembro de esta clase se muestran en la figura A.8.

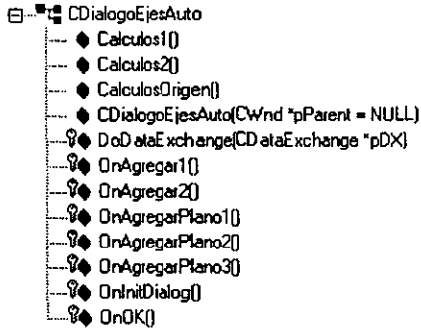


Figura A.8. Funciones miembro de la clase CDialogoEjesAuto.

CDialogoEjesAuto(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Definición manual de ejes.

void Calculos1(); Calcula la alineación en el espacio: rotación con respecto al los ejes X y Y.

void Calculos2(); Calcula la alineación con respecto al plano normal.

void CalculosOrigen(); Calcula el nuevo origen del sistema de coordenadas sobre la base de la intersección de los tres planos coordenados.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnAgregar1(); Agrega los valores de los puntos palpados sobre el plano superior del objeto en medición para determinar el eje Z.

afx_msg void OnAgregar2(); Agrega los valores de los puntos palpados sobre el plano lateral del objeto en medición para determinar los ejes X y Y.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la transformación de coordenadas.

afx_msg void OnAgregarPlano1(); Agrega el plano 1 para calcular el origen.

afx_msg void OnAgregarPlano2(); Agrega el plano 2 para calcular el origen.

afx_msg void OnAgregarPlano3(); Agrega el plano 3 para calcular el origen.

CDialogoEsfera

Esta clase fue creada para medir la geometría esfera y para realizar todos los cálculos necesarios para extraer sus características. Esta clase contiene las funciones miembro que se muestran en la figura A.9.

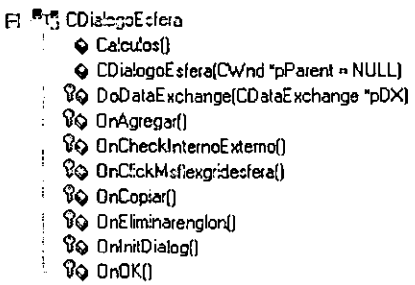


Figura A.9. Funciones miembro de la clase CDialogoEsfera.

CDialogoEsfera(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Esfera.

void Calculos(); Ajusta los puntos palpados a la ecuación de la esfera. Para los cálculos refiérase a la figura 3.17.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo, agrega datos de la PC7266 a la rejilla.

afx_msg void OnClickMsflexgridesfera(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Esfera.

afx_msg void OnCheckInternoExterno(); Maneja el evento pulsar sobre la caja de verificación Esfera interna. Actualiza el mapa de bits en la caja de diálogo que muestra la geometría interna o externa.

CDialogoLinea

Esta clase contiene las variables y funciones miembro necesarios para medir y determinar las características de una línea en el espacio. La clase CDialogoLinea contiene las funciones miembro que se muestran en la figura A.10.

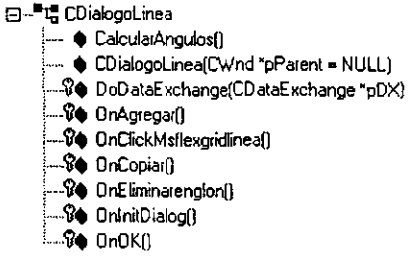


Figura A.10. Funciones miembro de la clase CDialogoLinea.

CDialogoLinea(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Línea.

void CalcularAngulos(); Ajusta los puntos palpados a la ecuación de la recta. Para los cálculos refiérase a la figura 3.12.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo, agrega datos de la PC7266 a la rejilla.

afx_msg void OnClickMsflexgridlinea(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Línea.

CDialogoMatriz

Esta clase se encarga de controlar el cuadro de diálogo utilizado para introducir la matriz de corrección para la máquina de medición por coordenadas, el cual aparece al hacer clic sobre item *matriz* del menú Calibrar. En la figura A.11 se muestran las funciones miembro de la clase CDialogoMatriz.

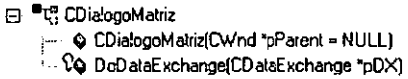


Figura A.11. Funciones miembro de la clase CDialogoMatriz.

CDialogoMatriz(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Matriz de calibración. Inicia valores por omisión de la matriz.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

Esta caja de diálogo recibe los valores de la matriz de calibración. Si estos valores no son proporcionados inicia valores por omisión de la matriz.

CDialogoPalpador

Esta clase define al cuadro de diálogo que nos permite introducir el diámetro del palpador, mismo que será considerado en los cálculos de las geometrías. Esta clase contiene las funciones miembro que se muestran en la figura A.12.

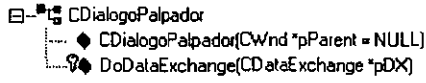


Figura A.12. Funciones miembro de la clase CDialogoPalpador.

CDialogoPalpador(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Calibración del palpador. Inicia a cero el diámetro del palpador.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones). Recibe el valor del diámetro del palpador, limitando el rango del mismo.

CDialogoParche

El cuadro de diálogo parche es definido por esta clase, y contiene las funciones miembro necesarias para los cálculos y medición de esta geometría. La figura A.13 muestra las funciones miembro de la clase CDialogoParche.

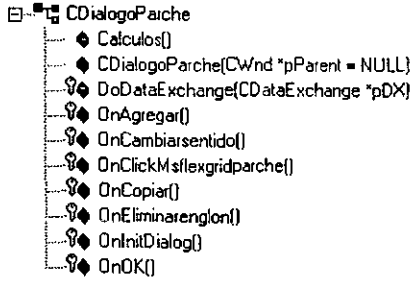


Figura A.13. Funciones miembro de la clase CDialogoParche.

CDialogoParche(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Parche triangular.

void Calculos(); Ajusta los puntos palpados a la ecuación del plano y calcula la normal.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo. Agrega datos de la PC7266 a la rejilla.

afx_msg void OnClickMsflexgridparche(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnCambiarsentido(); Cambia el sentido del vector normal al plano.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Parche triangular.

CDialogoPlano

Esta clase fue implementada para definir el cuadro de diálogo que nos permite realizar las mediciones de un plano en el espacio y determinar sus características. La clase CDialogoPlano contiene las funciones miembro que se muestran en la figura A.14.

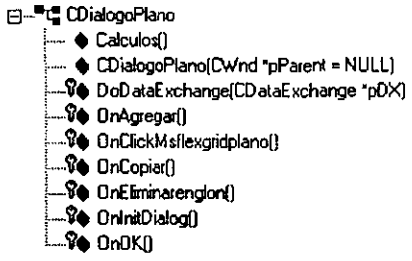


Figura A.14. Funciones miembro de la clase CDialogoPlano.

CDialogoPlano(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Plano.

void Calculos(); Ajusta los puntos palpados a la ecuación del plano. Para los cálculos refiérase a la figura 3.13.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo. Agrega datos de la PC7266 a la rejilla.

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

afx_msg void OnClickMsflexgridplano(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Plano.

CDialogoPunto

Esta clase define al cuadro de diálogo punto, utilizado para determinar la posición de un punto dentro del volumen de medición. En la figura A.15 se muestran las funciones miembro de esta clase.

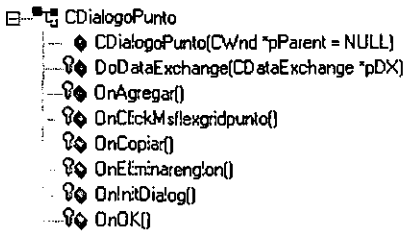


Figura A.15. Funciones miembro de la clase CDialogoPunto.

CDialogoPunto(CWnd* pParent = NULL); Constructor estándar de la caja de diálogo Punto.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

afx_msg void OnAgregar(); Maneja el evento pulsar sobre el botón Agregar de la caja de diálogo. Agrega datos de la PC7266 a la rejilla.

afx_msg void OnClickMsflexgridpunto(); Maneja eventos sobre la rejilla de la caja de diálogo.

afx_msg void OnCopiar(); Maneja el evento pulsar sobre el botón Copiar de la caja de diálogo. Copia la selección al portapapeles.

afx_msg void OnEliminarenglon(); Maneja el evento pulsar sobre el botón Eliminar renglón de la caja de diálogo. Elimina el renglón seleccionado.

virtual void OnOK(); Maneja el evento pulsar sobre el botón Aceptar y procesa la primitiva Punto.

virtual BOOL OnInitDialog(); Inicia variables y parámetros de la rejilla de la caja de diálogo.

CMainFrame

Esta clase es creada por appwizard y define el comportamiento de la ventana principal de la aplicación. Las funciones miembro de esta clase se muestran en la figura A.16.

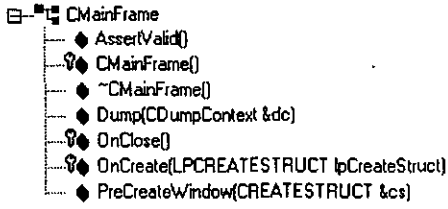


Figura A.16. Funciones miembro de la clase CMainFrame.

CMainFrame(); Constructor de la ventana principal. Carece de código. Insertado por el ambiente de programación.

virtual BOOL PreCreateWindow(CREATESTRUCT& cs); Impide que la ventana se pueda maximizar.

virtual ~CMainFrame(); Destructor. Carece de código. Insertado por el entorno de programación.

virtual void AssertValid() const; Insertada por el ambiente de programación con fines de depuración.

virtual void Dump(CDumpContext& dc) const; Insertada por el ambiente de programación con fines de depuración.

afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct); Inicia la barra de estado y la barra de herramientas.

afx_msg void OnClose(); Termina la aplicación y pregunta por salvar el documento.

CMmcCIApp

Esta clase contiene las funciones miembro que se muestran en la figura A.17.

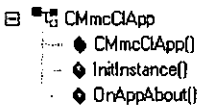


Figura A.17. Funciones miembro de la clase CMmcCIApp.

CMmcCIApp(); Constructor. Carece de código. Insertado por el ambiente de programación.

virtual BOOL InitInstance(); Inicia la instancia: Habilita controles, establece el perfil de la aplicación, establece la aplicación de documento único, habilita el ejecutar/abrir con DDE, despliega la ventana.

afx_msg void OnAppAbout(); Maneja el evento Acerca de MMCCI, del menú de opciones.

CMmcCIDoc

Esta clase, creada por appwizard, define el comportamiento del documento de la aplicación, permitiendo crear y abrir nuevos documentos. La figura A.18 muestra las funciones miembro de la clase CMmcCIDoc.

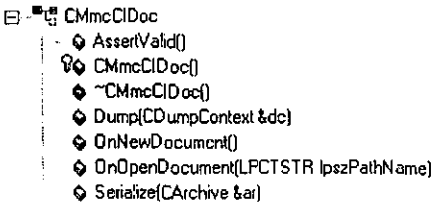


Figura A.18. Funciones miembro de la clase CMmcCIDoc.

CMmcCIDoc(); Constructor. Carece de código. Insertado por el ambiente de programación.

virtual BOOL OnNewDocument(); Establece el soporte para manejo de documentos.

virtual void Serialize(CArchive& ar); Soporte para serialización de datos. No utilizado.

virtual BOOL OnOpenDocument(LPCTSTR lpszPathName); Maneja un apuntador a la vista para procesar el mensaje Abrir del menú Archivo.

virtual ~CMmcCIDoc(); Destructor. Carece de código. Insertado por el ambiente de programación.

virtual void AssertValid() const; Insertada por el ambiente de programación con fines de depuración.

virtual void Dump(CDumpContext& dc) const; Insertada por el ambiente de programación con fines de depuración.

CMmcCIView

Esta clase define la vista del documento, y contiene todas las funciones miembro que son compartidas por las clases definidas para la medición de las geometrías. Esta clase también es la encargada de visualizar la posición espacial del palpador dentro del volumen de medición.

La clase CMmcCIView contiene una gran cantidad de funciones miembro, ya que esta clase es la encargada de manejar la vista de la aplicación. Las funciones miembro que la componen se muestran en la figura A.19.

CMmcCIView(); Constructor: Inicia la dirección de la PC7266, Inicia el texto de la historia.

void MostrarHistoria(); Muestra el texto de la historia en la ventana principal.

int ValidarResolucion(double); Subrutina para determinar si el valor a cargar tiene la resolución correcta.

void GuardarAlCerrar(); Llama al procedimiento OnArchivoGuardar.

void GuardarMatriz(); Guarda los valores de la matriz de calibración en el archivo Mmmcci.dat

int ImprimirEncabezado(CDC *pDC, CPrintInfo *pInfo); Establece el encabezado para la opción Imprimir.

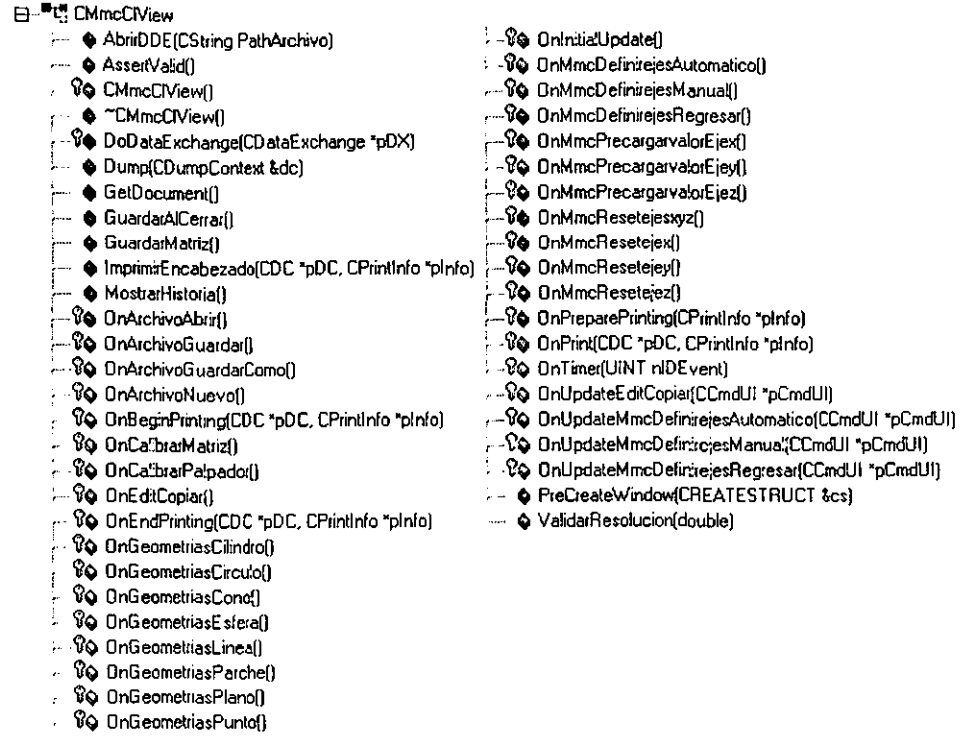


Figura A.19. Funciones miembro de la clase CMmcCIView.

void AbrirDDE(CString PathArchivo); Abre el archivo de la historia y establece un apuntador al texto de la historia.

virtual BOOL PreCreateWindow(CREATESTRUCT& cs); Establece el estilo de la ventana.

virtual void DoDataExchange(CDataExchange* pDX); Soporte DDX/DDV (Intercambio de datos con otras aplicaciones).

virtual void OnInitialUpdate(); Llamada inicialmente por el constructor: Carga la DLL 32PC7266 en memoria y obtiene la dirección de sus funciones, obtiene los valores de la matriz de calibración, establece el temporizador para el despliegue de lecturas.

virtual BOOL OnPreparePrinting(CPrintInfo* pInfo); Prepara la impresión del historial.

virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo); Inicia la impresión del historial.

virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo); Carece de código. Insertada por el ambiente de programación.

virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo); Maneja la opción Imprimir del menú Archivo.

virtual ~CMmcCIView(); Destructor. Libera la memoria de la DLL 32PC7266 y el temporizador.

virtual void AssertValid() const; Insertada por el ambiente de programación con fines de depuración.

virtual void Dump(CDumpContext& dc) const; Insertada por el ambiente de programación con fines de depuración.

afx_msg void OnGeometriasPunto(); Maneja la opción Punto del menú Geometrías.

afx_msg void OnMmcResetejex(); Maneja la opción Reset eje X del menú MMC.

afx_msg void OnMmcResetejey(); Maneja la opción Reset eje Y del menú MMC.

afx_msg void OnMmcResetejez(); Maneja la opción Reset eje Z del menú MMC.

afx_msg void OnMmcResetejesxyz(); Maneja la opción Reset ejes XYZ del menú MMC.

afx_msg void OnMmcPrecargarvalorEjex(); Maneja la opción Precargar valor/eje X del menú MMC.

afx_msg void OnMmcPrecargarvalorEjey(); Maneja la opción Precargar valor/eje Y del menú MMC.

afx_msg void OnMmcPrecargarvalorEjez(); Maneja la opción Precargar valor/eje Z del menú MMC.

afx_msg void OnTimer(UINT nIDEvent); Evento del temporizador: Obtiene datos de la PC7266, aplica la definición de ejes y la matriz de calibración a los datos de la PC7266, actualiza la barra de estado.

afx_msg void OnArchivoNuevo(); Maneja la opción Nuevo del menú Archivo.

afx_msg void OnGeometriasLinea(); Maneja la opción Línea del menú Geometrías.

afx_msg void OnGeometriasPlano(); Maneja la opción Plano del menú Geometrías.

afx_msg void OnGeometriasCirculo(); Maneja la opción Círculo del menú Geometrías.

afx_msg void OnGeometriasCilindro(); Maneja la opción Cilindro del menú Geometrías.

afx_msg void OnGeometriasEsfera(); Maneja la opción Esfera del menú Geometrías.

afx_msg void OnGeometriasParche(); Maneja la opción Parche triangular del menú Geometrías.

afx_msg void OnMmcDefinirejesManual(); Maneja la opción Definir ejes/Manual del menú MMC.

afx_msg void OnMmcDefinirejesAutomatico(); Maneja la opción Definir ejes/Automático del menú MMC.

afx_msg void OnMmcDefinirejesRegresar(); Maneja la opción Definir ejes/Regresar del menú MMC.

afx_msg void OnUpdateMmcDefinirejesRegresar(CCmdUI* pCmdUI); Habilita o deshabilita la opción Definir ejes/Regresar del menú MMC.

afx_msg void OnUpdateMmcDefinirejesManual(CCmdUI* pCmdUI); Habilita o deshabilita la opción Definir ejes/Manual del menú MMC.

afx_msg void OnUpdateMmcDefinirejesAutomatico(CCmdUI* pCmdUI); Habilita o deshabilita la opción Definir ejes/Automático del menú MMC.

afx_msg void OnArchivoGuardar(); Maneja la opción Guardar del menú Archivo.

afx_msg void OnArchivoGuardarComo(); Maneja la opción Guardar como del menú Archivo.

afx_msg void OnUpdateEditCopiar(CCmdUI* pCmdUI); Habilita o deshabilita la opción Copiar del menú Editar.

afx_msg void OnEditCopiar(); Maneja la opción Copiar del menú Editar. Copia la selección del historial en el portapapeles.

afx_msg void OnCalibrarPalpador(); Maneja la opción Palpador del menú Calibrar.

afx_msg void OnCalibrarMatriz(); Maneja la opción Matriz del menú Calibrar.

afx_msg void OnArchivoAbrir(); Maneja la opción Abrir del menú Archivo.

afx_msg void OnGeometriasCono(); Maneja la opción Cono del menú Geometrías.

ANEXO B

Diagramas de conexiones

El diagrama de los conectores empleados entre el codificador óptico y la tarjeta PC7266 se muestra en la figura B.1.

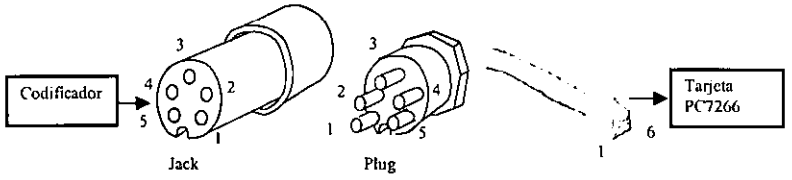


Figura B.1. Diagrama de conectores empleados.

La tabla B.1. muestra las conexiones físicas realizadas entre los conectores de la figura B.1.

Codificador	Rojo	Verde	Azul	Blanco	Negro	Estaño
Jack	5	4	3	2	1	1
Plug	5	4	3	2	1	-
Conector telefónico	4	5	-	3	6	-

Tabla B.1. Conexiones.

Bibliografía

[1] Acu-Rite, *Mini-scale and mate system encoders*, Acu-Rite, 1994.

[2] AENOR, *Metrología dimensional*, tomo IV, editorial AENOR (Asociación Española de Normalización), 1998.

[3] Alegre B. José María, *Transductores y medidores electrónicos*, editorial Marcombo, España 1983, segunda edición.

[4] Bentley B. Jhon P., *Sistemas de medición* , Compañía Editorial Continental , primera edición en español, México 1993.

[5] Carballar José A., *El Libro de las comunicaciones del PC*, editorial RA-MA, México 1997.

[6] Ceballos Francisco Javier, *Visual C++ Aplicaciones para Win32*, editorial RA-MA, 1998.

[7] Kate Gregory, Ussing Visual C++ 6, Ed. QUE, 1999.

[8] Kruglinski David J., *Programación avanzada con Microsoft Visual C++*, editorial Mc-Graw Hill.

[9] Pallás Areny Ramón. *Transductores y acondicionadores de señal*, editorial Marcombo, España 1989.

[10] Sánchez Angel Mª, Carro Javier, *Elementos de metrología*, editorial Universidad politécnica de Madrid.

[11] Sandoval Rodríguez Juan Antonio, *Metrología Dimensional*, editorial Aconcagua, México 1995, primera edición.

[12] Scragg Greg W., *Computer organization*, editorial McGraw-Hill, USA, 1992.

[13] Ureña Luis A., otros, *Fundamentos de informática*, editorial RA-MA, México 1998.

[14] US-Digital, *PC7266, PC to Incremental Encoder Interface Card*, Technical Data Rev. 02.29.97, September 1997.

[15] Metrology basics, url: *<www_uktm.external.hp.com/mikehut/basics.html>*