UNIVERSIDAD LA SALLE

ESCUELA DE INGENIERIA

Incorporada a la **U. N. A. M.**

DESARROLLO Y CODIFICACION DEL PROGRAMA DE CONTROL DE UN EQUIPO PORTATIL DE ADQUISICION DE DATOS

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

Area: Ingeniería Electrónica

P R E S E N T A

EDGARD MAURICIO CASTILLO VELASCO

Director de Tesis: Ing. Guillermo Aranda Pérez

MEXICO, D. F. **1991**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E G E N E R A L.

4.1.2.4 Escritura en el puerto serie
4.1.2.5 Escritura en el buffer del puerto serie
4.1.2.6 Estatus del puerto serie

4.1.3 Funciones de control del generador de pulsos programable

4.1.3.1 Programación del COP

4.1.4 Funciones de control del teclado

4.1.4.1 Programación del circuito 8255
4.1.4.2 Verificación de tecleo
4.1.4.3 Decodificación

4.1.5 Funciones de Conversión

4.1.5.1 Conversión de Octeto a BCD
4.1.5.2 Conversión de BCD a Octeto
4.1.5.3 Conversión de Entero a BCD
4.1.5 4 Conversión de BCD a Entero
4.1.5.5 Redondeo de Enteros

4.1.6 Funciones Varias

4.1.6.1 Captura de un caracter
4.1.6.2 Captura de un número entero
4.1.6.3 Captura de números con punto flotante
4.1.6.4 Captura de cadena de caracteres
4.1.6.5 Intercambio de variables
4.1.6.6 Retardo (microsegundos)
4.1.6.7 Retardo (milisegundos)
4.1.6.8 Retardo (segundos)
4.1.6.9 Verificación de la bandera de comunicación del puerto serie

4.2 Presentación del Equipo, Definición de variables globales y Rutina principal

4.2.1 Presentación del Equipo Portátil de Adquisición de Datos

4.2.1.1 Despliegue del Logotipo del I.I.E.

4.2.2 Definiciones Globales del programa de control del equipo portátil de adquisición de datos

4.2.2.1 Inclusión de librerias estandar (.h)
4.2.2.2 Definición de constantes globales
4.2.2.3 Definición de los códigos de las téclas
4.2.2.4 Definición de los códigos de reconocimiento y de no_ reconocimiento en la comunicación del equipo con la computadora
4.2.2.5 Definición de Tipos simples

I N D I C E D E F I G U R A S Y T A B L A S

Tabla Página

Introducción:

Tradicionalmente, la detección de fallas en equipos rotatorios se había realizado en base a técnicas de mantenimiento preventivo; es decir, la práctica de servicio a la maquinaria en períodos previamente establecidos.

Este medio de preservar la disponibilidad de los sistemas sujetos a vibraciones, presenta una serie de desventajas debido a la necesidad de alterar o suspender su operación.

Los costos por reposición de energía y los costos propios de los mantenimientos preventivos oscilan entre el 15 y el 40% del costo total de operación.

Debido a lo anterior, surgió la necesidad de crear un programa de vigilancia sistemática de los equipos rotatorios que, a partir del estudio de los movimientos de vibración, pudiera determinar el momento mas propicio para realizar dichas actividades de mantenimiento.

Este programa recibió el nombre de "Monitoreo", y dió origen, a su vez, a otro concepto que, en algunos paises industrializados ha reportado considerables ahorros en todos los aspectos: El Mantenimiento Predictivo.

En la industria de la generación de energía eléctrica, el Mantenimiento Predictivo ha tenido gran éxito en plantas importantes a nivel mundial.

Ante tales perspectivas, el Departamento de Equipos Mecánicos del Instituto de Investigaciones Eléctricas (IIE) desarrolló un

programa de Mantenimiento Predictivo en base al monitoreo y al análisis de vibraciones de equipos rotatorios y que lleva por nombre "Sistema Portátil".

Este sistema consta de tres partes: un equipo portátil de adquisición de datos, un paquete de cómputo y una microcomputadora.

El objetivo que persigue el Instituto de Investigaciones Eléctricas, es generalizar el mantentimiento predictivo en todas las Plantas de la Comisión Federal de Electricidad, y posteriormente a la industria en general.

Sin embargo, los equipos de adquisición de datos disponibles comercialmente tienen la limitante de sus costos elevados (del orden de 10,000 a 15,000 US DLS).

Por esta razón el IIE tomó la decisión de producir un equipo que, además de satisfacer los requerimientos del programa de mantenimiento predictivo, fuera competitivo comercialmente (50% del costo), con respecto a los disponibles en el mercado.

La presente tesis está enfocada hacia el desarrollo del programa de control (software) de este equipo portátil de adquisición de datos.

CAPITULO I.- EL SISTEMA PORTATIL Y EL ANALISIS DE VIBRACIONES.

1.1 Introducción al Análisis de Vibraciones.

Con objeto de maximizar la disponibilidad y el rendimiento de los equipos mecánicos en operación, es importante conocer el comportamiento, tanto de las piezas como de los procesos que se realizan.

Para evaluar el funcionamiento de algunos equipos así como el grado de deterioro, el operador emplea instrumentación convencional e inclusive sus propios sentidos. En equipos grandes y complejos, la precisión de los diagnósticos puede repercutir en ahorros o pérdidas cuantiosas, por lo cual se hace necesario un análisis mas exhaustivo y cuidadoso del proceso.

En el caso de los equipos mecánicos rotatorios, se presentan vibraciones que repercuten en desgastes y rupturas de las piezas.

El análisis de vibraciones en este tipo de maquinaria, es por tanto una herramienta que, al mejorar el diagnóstico de fallas previene una descompostura que pudiera en determinado momento suspender las operaciones.

Las ventajas que tiene este análisis del proceso en base al estudio del comportamiento de las vibraciones principalmente benefician al proceso debido a que no es necesario suspender

la operación de la maquinaria.
Básicamente el análisis de vibraciones involucra las siguientes actividades:

a) Detección de señales de vibración: Es la detección de señales de vibración mediante transductores que transforman dichas señales en otras adecuadas para su procesamiento.
b) Acondicionamiento: Proceso de adaptación de las señales detectadas a los equipos receptores.
c) Almacenamiento: Actividad por la cual se graba la información para proporcionarla en cualquier momento en que se requiera.
d) Proceso: Operaciones tales como la descomposición espectral, filtrado síncrono, digitalización y comportamiento numérico de las señales adquiridas.
e) Presentación: Es la impresión de la información entregada por el proceso, en forma de reporte o gráfica.
f) Análisis: Estudio de los espectros, órbitas de onda, fase, vibraciones síncronas armónicas, transitorios, etc.

El análisis de vibraciones en todas sus aplicaciones está relacionado con la identificación o diagnóstico de fallas entre las que se pueden mencionar las siguientes:

a) Desbalanceo o desalineamiento, flexiones permanentes, fisuras, rozamientos y piezas sueltas en rotores.
b) Inestabilidad fluidodinámica en chumaceras y sellos.
c) Resonancia y vibraciones transmitidas entre partes o entre equipos.
d) Defectos en baleros, sellos y engranes.
e) Vibraciones inducidas por flujos.
f) Vibraciones de cimentaciones, estructuras y soportes.

1.1.1 Tipos de transductores

Los transductores de vibración se pueden clasificar en base a sus características propias; es decir, su principio de operación, o según sus requisitos de alimentación. También pueden clasificarse de acuerdo a sus capacidades para la detección de vibraciones según el parámetro a detectar, el punto de referencia con respecto al cual se miden las vibraciones o las condiciones de la detección.

Algunos transductores detectan su propia vibración con respecto a un marco inercial (fijo) de referencia; estos transductores se conocen como "sísmicos" y deben montarse de modo tal que vibren junto con la pieza cuya vibración absoluta se desea detectar. Este tipo de transductor es el único que proporciona una medida absoluta del valor de la vibración, los demás pueden detectar la vibración por contacto físico de alguna de sus partes con la pieza

observada, y se dice que se obtiene un valor relativo de la vibración.
Para el análisis de vibraciones los transductores más adecuados son los sísmicos de velocidad y aceleración y los de desplazamiento relativo que utilizan corrientes de Foucault.

1.1.1.1 Transductores Sísmicos

Estos transductores fueron creados originalmente para la detección de temblores y basan su funcionamiento en un sistema masa resorte amortiguador de un grado de libertad.
Los transductores sísmicos pueden clasificarse en dos grupos: Transductores que operan a frecuencias superiores a su frecuencia natural y los transductores que operan a frecuencias inferiores a su frecuencia natural.

a) Transductores Sísmicos de Velocidad:
Estos sensores están constituídos por cuatro elementos básicamente: Un Imán permanente, un resorte, una bobina y una carcaza (Figura 1.1).
La bobina está suspendida por un par de resortes dentro de la carcaza, la cual presenta un imán fijo en la parte inferior y un entrehierro con el fin de que el campo magnético sea cortado perpendicularmente por la bobina.
La diferencia de potencial resultante del movimiento de la bobina dentro del flujo magnético es proporcional a la

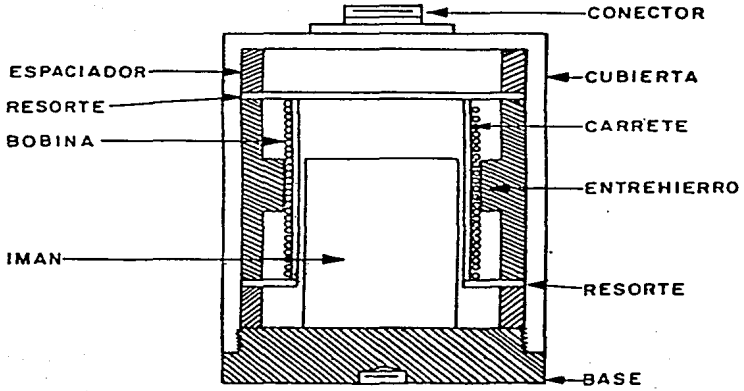

FIGURA 1.1 TRANSDUCTOR SISMICO DE VELOCIDAD

velocidad relativa que existe entre la bobina y la carcaza. Si la frecuencia natural del transductor es inferior a la frecuencia de la velocidad de vibración, la bobina permanecerá practicamente suspendida en el espacio. De ahí que el transductor genera un voltaje proporcional a la velocidad absoluta de la carcaza.

El objeto de introducir un amortiguamiento es el de limitar las amplitudes del movimiento de la masa sísmica (bobina o imán) cuando el transductor vibre a una frecuencia igual a su frecuencia natural.

La expresión matemática del voltaje generado se muestra en la ecuación 1.1

$$V = -BLv \quad \ldots\ldots\ldots\ldots\ldots \quad (1.1)$$

Donde V es el voltaje entre las terminales de la bobina, B es la densidad del flujo magnético, L es la longitud del conductor de la bobina y v la velocidad entre la bobina y el imán.

b) Transductores Sísmicos de Aceleración:

Este tipo de transductor se utiliza para medir la aceleración a la que se somete un cuerpo, de ahí que tambien reciba el nombre de acelerómetro.

El acelerómetro sísmico que comúnmente se utiliza en el análisis de vibraciones de máquinas sujetas a movimientos

rotatorios es el piezoeléctrico.

La mayoría de los acelerómetros piezoeléctricos tienen el tamaño aproximado de una batería de linterna y no requieren de una fuente externa para su funcionamiento. Sin embargo, la señal que proporcionan es muy débil, por lo cual se requiere de un preamplificador como acondicionador de la señal.

Los elementos básicos de un acelerómetro piezoeléctrico son: una masa, un cristal piezoeléctrico y una base.

La carga generada es proporcional a la fuerza aplicada al material piezoeléctrico. Algunas constantes de proporcionalidad varían con la temperatura y con la frecuencia del esfuerzo aplicado (ecuación 1.2).

$$q = A\ d\ e \quad \text{.....(1.2)}$$

Donde "q" es la carga eléctrica en Coulombs, "d" la constante piezoeléctrica en Coulombs/Newtons, "a" es el área del cristal en metros cuadrados y "e" el esfuerzo al que se somete el material en N/m.

1.1.1.2 Transductor de Corrientes de Foucault para Desplazamiento

Este tipo de transductor basa su funcionamiento en la generación de un potencial proporcional al claro existente

entre él y la superficie vibrante, es decir, no hay contacto físico.

Como las variaciones en el claro pueden ser debidas al desplazamiento de la superficie y/o al de la sonda detectora, el voltaje es proporcional al desplazamiento relativo.

Los elementos que forman el transductor de desplazamiento son:

- Sonda
- Cable de extensión
- Oscilador-demodulador

La sonda consiste en una bobina espiral cuyo plano es perpendicular a un pequeño cuerpo de acero con rosca que la contiene. En el otro extremo del cuerpo metálico se encuentra otro extremo de cable cuya longitud varía desde 100 mm hasta 2 metros y que se conecta al cable de extensión.

De la unión de la sonda con el cable de extensión, resulta un circuito resonante L-C, donde el cable constituye las capacitancias y la sonda las inductancias.

El oscilador-demodulador es el dispositivo que suministra la energía requerida para mantener las oscilaciones en el circuito resonante formado por la sonda y el cable de extensión a una frecuencia aproximada de 2.5 MHz. Dicha señal es modulada en amplitud por las variaciones de corriente cuando el claro aumenta o disminuye.

La parte demoduladora sirve para obtener la porción negativa o positiva de la envolvente pero las variaciones de la

envolvente no son proporcionales a las del claro, por tanto, se realiza un proceso de amplificación no lineal para corregir la desproporcionalidad. El diagrama de bloques de un oscilador-demodulador se muestra en la figura 1.2.

1.1.2 Técnicas de Análisis

Las técnicas de análisis son los cálculos, operaciones y desplegados que se realizan con los datos de vibración.
Las técnicas mas comúnmente utilizadas en el análisis de vibraciones son:

- Nivel global de vibración.
- Señal en el tiempo.
- Espectro lineal.
- Vectores.

1.1.2.1 Valor global, valor rms y límites de alerta y alarma de señales de vibración

El valor global de la vibración corresponde a la amplitud de la señal tomada en un intervalo de tiempo determinado. El tiempo considerado es generalmente aquel que utiliza la flecha en realizar una revolución completa.
La obtención del valor global se realiza a partir de los conceptos de valor cero a pico (amplitud máxima) y el valor rms (valor cuadrático medio en el periodo).

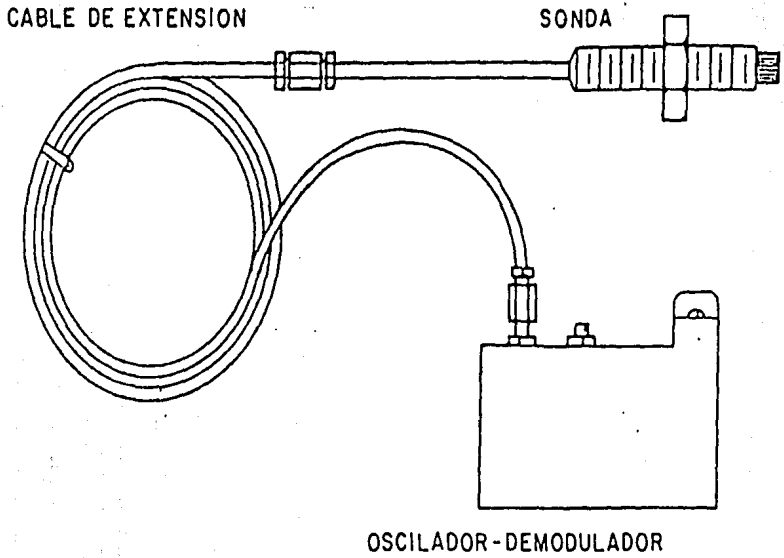

FIGURA 1.2 DIAGRAMA DE BLOQUES DE UN OSCILADOR-DEMODULADOR

La representación del valor global de una señal de vibración adquirida mediante un sensor de desplazamiento y las expresiones que relacionan dichos valores (1.3 y 1.4) se muestran a continuación:

$$Vpp = 2\ Vcp \quad \ldots\ldots\ldots\ldots \quad (1.3)$$

$$Vrms = 0.7071\ Vcp \quad \ldots\ldots\ldots\ldots \quad (1.4)$$

El uso principal de esta técnica está en la vigilancia continua de maquinaria.

El nivel global es una manera sencilla y rápida para medir los niveles de vibración, sin embargo, revela muy poco acerca de la causa de un problema de vibración.

Un nivel global máximo, es algunas veces especificado como un parámetro de conformidad de calidad para equipo nuevo y depende tanto de la operación que se realice, como de las condiciones de funcionamiento. A partir de esta idea se originaron los conceptos de niveles de alerta y alarma. Estos parámetros, indican los correspondientes niveles de alerta y alarma, tanto para los valores globales como para los valores rms, que puedan alcanzar las señales de vibración.

De la comparación de los valores global y rms de la señal de vibración, con los límites propios del equipo que se monitorea, se concluye si se han sobrepasado los niveles permisibles de operación.

1.1.2.2 Señal en el Tiempo

La señal en el tiempo es una representación de las variaciones en las amplitudes instantáneas de las señales de vibración, tomadas en un intervalo determinado de tiempo (figura 1.3).
La señal de vibración consiste en una serie de lecturas tomadas por el transductor y digitalizadas para su observación en instrumentos tales como el osciloscopio. Dado que se despliega el nivel instántaneo, las unidades son micrómetros o mils, mm/s o pul/s., dependiendo del parámetro físico que se está midiendo.
La utilidad de esta técnica de análisis de las señales de vibración, estriba principalmente en la posibilidad de almacenarse fácilmente dicha forma de onda como una serie de amplitudes. Si la señal consiste en la suma de varias señales de vibración, cada una de ellas con magnitud y frecuencia específica, se dice que se trata de una señal de vibración compleja.

1.1.2.3 Espectro lineal

El espectro lineal es un desplegado de las componentes de frecuencia presentes en una señal de vibración en unidades proporcionales a las unidades de ingeniería. Sobre la escala horizontal, el diagrama desplegará los valores de las fre-

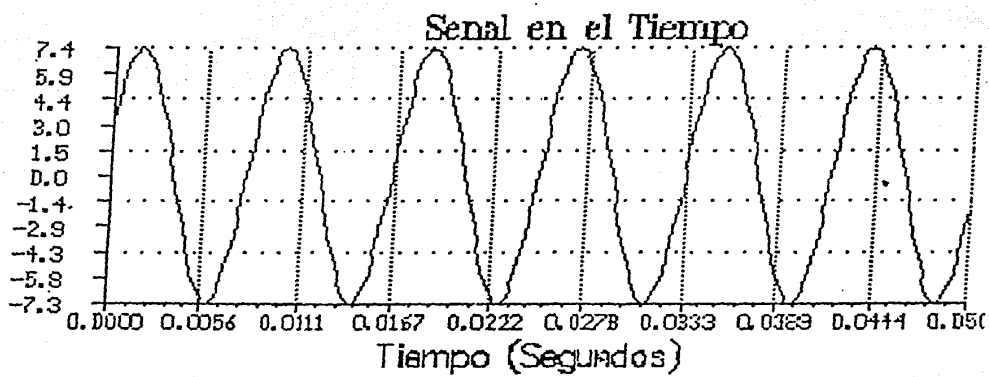

FIGURA 1.3 SEÑAL EN EL TIEMPO

cuencias en Hertz, armónicas o rpms, mientras que en la escala vertical, la amplitud en unidades de desplazamiento, velocidad o aceleración (Figura 1.4).

Esta es la técnica mas aplicada para la localización de fallas en equipos rotatorios. La amplitud de cada pico describe la severidad de la vibración, mientras que la frecuencia de la vibración sobre el eje horizontal, se usa para determinar la fuente de la vibración.

La siguiente tabla muestra algunos de los problemas típicos de vibraciones y las frecuencias a las que se presentan:

Problema	Frecuencia.
1) Desbalanceo	1 x rpm
2) Desalineamiento	2 x rpm
3) Inestabilidad en Chumaceras.	0.4 a 0.5 x rpm
4) Problemas en impulsores engranes y álabes.	n x rpm
5) Aflojamiento de partes	1, 2, 3, x rpm

Notación: rpm = frecuencia en rpm de giro del rotor.
n = Número de impulsores, álabes o dientes de engranes

1.2 Características del Sistema PORTATIL

El Sistema PORTATIL es un programa de mantenimiento predictivo de equipos rotatorios desarrollado por el departamento de Equipos Mecánicos del Instituto de Investigaciones Eléc-

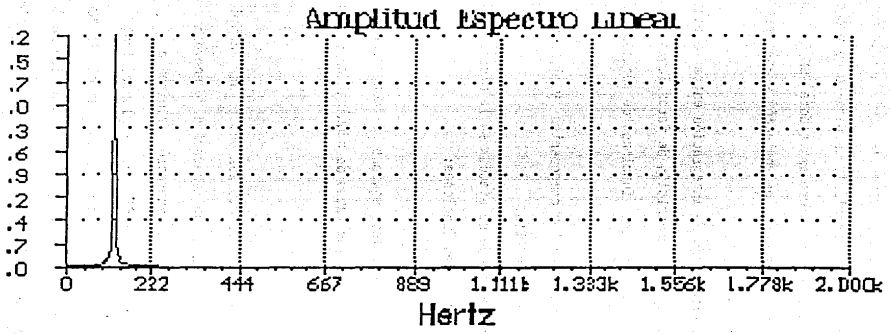

FIGURA 1.4 ESPECTRO LINEAL

tricas, y consiste de un equipo comercial de adquisición de información (mediciones de vibración), un paquete de programas de cómputo (PORTATIL), una microcomputadora IBM AT para el almacenamiento, procesamiento, presentación y análisis de datos y una impresora.

PORTATIL tiene un alto grado de integración entre la microcomputadora y el equipo de adquisición de datos logrando así simplificar y automatizar las funciones de adquisición y análisis.

En la figura 1.5 se muestran los componentes y el flujo de información del Sistema PORTATIL.

El conjunto de programas que se ejecuta en la microcomputadora, realiza las siguientes funciones:

1) Define los datos de los equipos a monitorear, así como los datos de las rutas de adquisición y puntos de monitoreo.

   La información está estructurada en: planta (debido a que se trata de plantas de generación de energía eléctrica), unidades de generación, equipos a monitorear y puntos de medición. La planta, las unidades y los equipos son definidos durante la "Configuración de planta". Los puntos de medición se definen durante la "Configuración de rutas de adquisición".

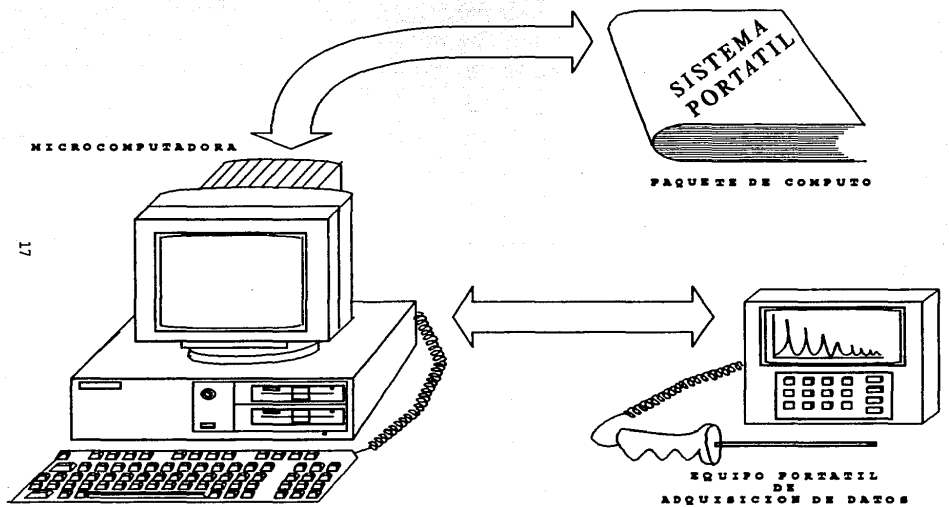

FIGURA 1.5 COMPONENTES Y FLUJO DE INFORMACION DEL SISTEMA PORTATIL.

Los puntos de medición son generalmente de vibración; sin embargo, en algunos casos se requieren otras variables como la temperatura u otras que pueden ser capturadas manualmente a partir de indicadores escalares, mismas que se censarán con el Equipo de Adquisición de Datos.

Las rutas de adquisición se configuran en base a los equipos que serán monitoreados y los puntos de medición de los mismos.

Para cada ruta se define una periodicidad de recorrido en semanas.

2) Programa al equipo de adquisición de datos para la ruta específica y recupera los datos del mismo.

Una interfaz serie RS-232-C es utilizada para la transferencia de información entre el equipo de adquisición de datos y la computadora.

3) Crea y mantiene una base de datos con las mediciones obtenidas por el equipo de adquisición de datos.

4) Compara los valores de la vibración para detección de alertas y alarmas y de cambios significativos en los valores entre 2 recorridos sucesivos.

5) Genera reportes y desplegados gráficos para la evaluación del estado de funcionamiento de los equipos.

1.2.1 Análisis de vibraciones con el sistema PORTATIL

El proceso de análisis de vibraciones que se lleva a cabo mediante el sistema PORTATIL está basado fundamentalmente en la detección de cambios significativos en el comportamiento característico de los equipos rotatorios. La manera como el sistema PORTATIL realiza dicha detección, es por medio de la información que proporciona al usuario en forma de diagramas de evolución, espectros en cascada, diagramas de tendencia y una serie de reportes que a continuación se detallan:

1.2.1.1 Reportes que entrega el sistema portátil

a) Reporte de configuración de planta:
Comprende datos o parámetros definidos para la plantas, unidades de generación de energía eléctrica, y equipos.

b) Reporte de configuración de rutas:
Comprende los datos generales de la ruta, los equipos a monitorear y los parámetros de los puntos de medición.

c) Lista de Despacho:
Comprende, la definición de los parámetros de adquisición para todos los puntos de una ruta con datos tales como localización del punto de medición, sensor a utilizar, el canal de medición, etc.

d) Reporte de valores globales:
Contiene los valores globales de todos los puntos de una ruta, obtenidos en un recorrido. Figura 1.6.

e) Reporte de Alertas y Alarmas:
Contiene un listado de todos los puntos que han rebasado los niveles de alerta y alarma previamente definidos durante la configuración. Esta opción despliega un mensaje en la pantalla en los casos en los que ciertos puntos y por tanto equipos que puedan tener problemas. El uso adecuado de este reporte permite reducir el tiempo de análisis de la información de un recorrido (figura 1.7).

f) Reporte de cambios significativos:
Contiene un listado de todos los puntos en los cuales se detectó un cambio de recorrido a recorrido, por encima de un valor definido por el usuario.

1.2.1.2 Salidas gráficas del sistema portátil

Las salidas gráficas que el sistema provee son:
a) Diagrama de Barras.
Presenta un diagrama de barras de los puntos de vibración medidos en un equipo en una determinada dirección. Permite visualizar rápidamente en que parte de un equipo se tienen las vibraciones mas significativas.

Equipo : TURB-1

Punto/ Canal	Nombre	Descripcion	TM	ES	Hora	Valor	Alarma Baja	Alarma Alta	Unidades
10-1	VIAX1	VIB AXIAL L-L SENAL	V	VD	16:36	9.88	0.00	10.00	puig/s pk
10-1	VIAX1	VIB AXIAL L-L SENAL	V	VR	16:36	0.08	0.00	2.00	puig/s pk
10-1	*VIAX1*	*VIB AXIAL L-L SENAL*	*V*	*PS*	*16:36*	*10.33*	*0.00*	*0.50*	*puig/s pk*
10-1	VIAX1	VIB AXIAL L-L SENAL	V	DC	16:36	-0.00	-10.00	0.00	Volts CD
20-1	VIRAD1	VIB RAD LL 0-90 SEN	V	VD	16:37	9.88	0.00	10.00	puig/s pk
20-1	VIRAD1	VIB RAD LL 0-90 SEN	V	VR	16:37	0.10	0.00	3.00	puig/s pk
20-1	*VIRAD1*	*VIB RAD LL 0-90 SEN*	*V*	*PS*	*16:37*	*10.31*	*0.00*	*0.50*	*puig/s pk*
20-1	VIRAD1	VIB RAD LL 0-90 SEN	V	DC	16:37	-0.00	-10.00	0.00	Volts CD
20-2	VIRAD2	VIB RAD LL 0-90 SEN	V	VD	16:37	9.80	0.00	10.00	puig/s pk
20-2	VIRAD2	VIB RAD LL 0-90 SEN	V	VR	16:37	0.11	0.00	3.00	puig/s pk
20-2	*VIRAD2*	*VIB RAD LL 0-90 SEN*	*V*	*PS*	*16:37*	*10.18*	*0.00*	*0.50*	*puig/s pk*
20-2	VIRAD2	VIB RAD LL 0-90 SEN	V	DC	16:37	-0.03	-10.00	0.00	Volts CD
30-1	VIAX2	VIB AXIL LM V GLOBAL	V	VD	16:37	9.85	0.00	10.00	puig/s pk
40-1	VIRAD3	VIB RAD LM 0-G V SEN	V	VD	16:37	5.00	0.00	10.00	mils pp
40-1	VIRAD3	VIB RAD LM 0-G V SEN	V	VR	16:37	0.07	0.00	3.00	mils pp
40-1	VIRAD3	VIB RAD LM 0-G V SEN	V	PS	16:37	5.22	0.00	0.50	mils pp
40-1	VIRAD3	VIB RAD LM 0-G V SEN	V	DC	16:37	-0.01	-10.00	0.00	Volts CD

Equipo : BAAN-1

Punto/ Canal	Nombre	Descripcion	TM	ES	Hora	Valor	Alarma Baja	Alarma Alta	Unidades
50-1	VIBAX1	VIB AXIAL L-L VG	V	VD	16:37	9.88	0.00	10.00	puig/s pk
60-1	VIRAD1	VIB RAD LL 0-90G SEN	V	VD	16:38	9.87	0.00	10.00	puig/s pk
60-1	VIRAD1	VIB RAD LL 0-90G SEN	V	VR	16:38	0.03	0.00	3.00	puig/s pk
60-1	*VIRAD1*	*VIB RAD LL 0-90G SEN*	*V*	*PS*	*16:38*	*10.27*	*0.00*	*0.50*	*puig/s pk*
60-1	VIRAD1	VIB RAD LL 0-90G SEN	V	DC	16:38	-0.01	0.00	-10.00	Volts CD
60-2	VIRAD2	VIB RAD LL 0-90G SEN	V	VD	16:38	9.53	0.00	10.00	puig/s pk
60-2	VIRAD2	VIB RAD LL 0-90G SEN	V	VR	16:38	0.17	0.00	3.00	puig/s pk
60-2	VIRAD2	VIB RAD LL 0-90G SEN	V	PS	16:38	9.88	0.00	0.50	puig/s pk
60-2	VIRAD2	VIB RAD LL 0-90G SEN	V	DC	16:38	-0.03	-10.00	0.00	Volts CD
70-1	VIBAX2	VIB AXIAL LM V GLOB	V	VD	16:38	9.86	0.00	10.00	puig/s pk
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	VD	16:39	4.99	0.00	10.00	mils pp
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	VR	16:39	0.07	0.00	3.00	mils pp
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	PS	16:39	5.19	0.00	0.50	mils pp
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	DC	16:39	-0.01	0.00	-10.00	Volts CD

FIGURA 1.6 REPORTE DE VALORES GLOBALES

Listado de Alarmas :

Planta : IIE Descripcion : INS. DE INV. ELEC.

Recorrido del : 12-NOV-90

Siglas :
TM : Tipo de Medicion
 V : Vibracion P : Proceso S : Cond. Maquina
ES : Parametro Estatico
 VD : Vibracion Directa VR : Vibracion del Rotor
 PS : Prime Spike HF : Vibracion de Alta Frecuencia
 DC : DC Gap

Italicas : Mediciones en que ha ocurrido un Sobre-Rango

Equipo : TURB-1

Punto/ Canal	Nombre	Descripcion	TM	ES	Hora	Valor	Alarma Baja	Alarma Alta	Unidades
10-1	*VIAX1*	*VIB AXIAL L-L SENAL*	*V*	*PS*	*16:36*	*10.33*	*0.00*	*0.50*	*pulg/s pk*
20-1	*VIRAD1*	*VIB RAD LL 0-90 SEN*	*V*	*PS*	*16:37*	*10.31*	*0.00*	*0.50*	*pulg/s pk*
20-2	*VIRAD2*	*VIB RAD LL 0-90 SEN*	*V*	*PS*	*16:37*	*10.18*	*0.00*	*0.50*	*pulg/s pk*
40-1	VIRAD3	VIB RAD LM 0-G V SEN	V	PS	16:37	5.22	0.00	0.50	mils pp

Equipo : BAAN-1

Punto/ Canal	Nombre	Descripcion	TM	ES	Hora	Valor	Alarma Baja	Alarma Alta	Unidades
60-1	*VIRAD1*	*VIB RAD LL 0-90G SEN*	*V*	*PS*	*16:38*	*10.27*	*0.00*	*0.50*	*pulg/s pk*
60-1	VIRAD1	VIB RAD LL 0-90G SEN	V	DC	16:38	-0.01	0.00	-10.00	Volts DC
60-2	VIRAD2	VIB RAD LL 0-90G SEN	V	PS	16:38	9.88	0.00	0.50	pulg/s pk
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	PS	16:39	5.19	0.00	0.50	mils pp
80-1	VIRAD3	VIB RAD LM 0-G SENAL	V	DC	16:39	-0.01	0.00	-10.00	Volts DC

FIGURA 1.7 REPORTE DE ALERTAS Y ALARMAS

b) Gráficas de Evolución

Presentan la evolución de los valores globales de un punto de medición a lo largo del tiempo, es decir, a lo largo de los diferentes recorridos.

Permite evaluar rápidamente si la vibración está creciendo (indicativa de que un problema puede estar aumentando en su severidad) o ha permanecido constante (figura 1.8).

c) Gráfica de Tendencia

La tendencia es una predicción que se hace del valor futuro que tomará el valor global de vibración, en base a una interpolación polinomial de los valores obtenidos durante los recorridos (figura 1.9).

d) Espectro Lineal y Señal en el Tiempo

Presenta la señal en el tiempo capturada y el espectro lineal de la misma, de un punto de medición. Este es el formato básico del PORTATIL, dada la correlación que existe entre las causas de vibración y frecuencias específicas (figura 1.10).

e) Orbitas.

Presenta la órbita o movimiento que efectúa el centro de un rotor dentro de una chumacera. Para obtenerla es necesario utilizar sensores de desplazamiento observando directamente la flecha y en configuración ortogonal (figura 1.11).

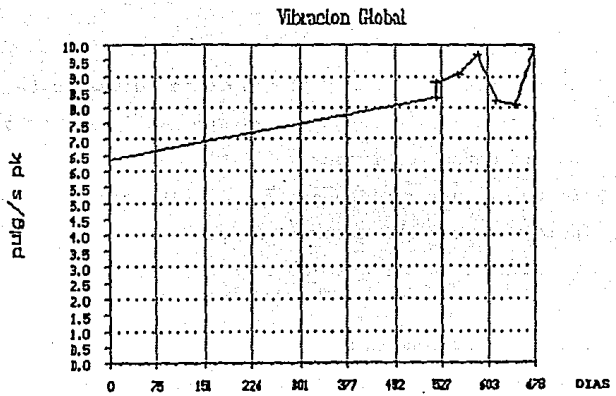

FIGURA 1.8 EVOLUCION

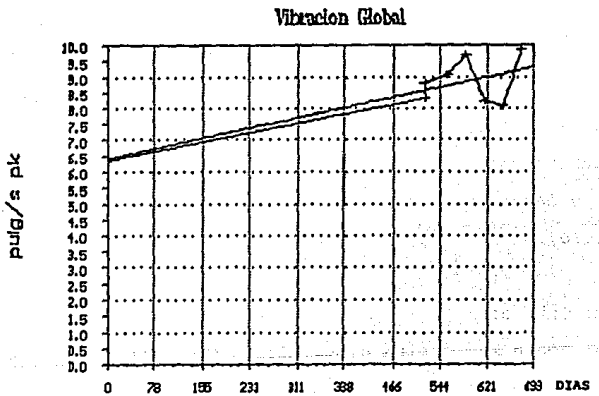

FIGURA 1.9 TENDENCIAS

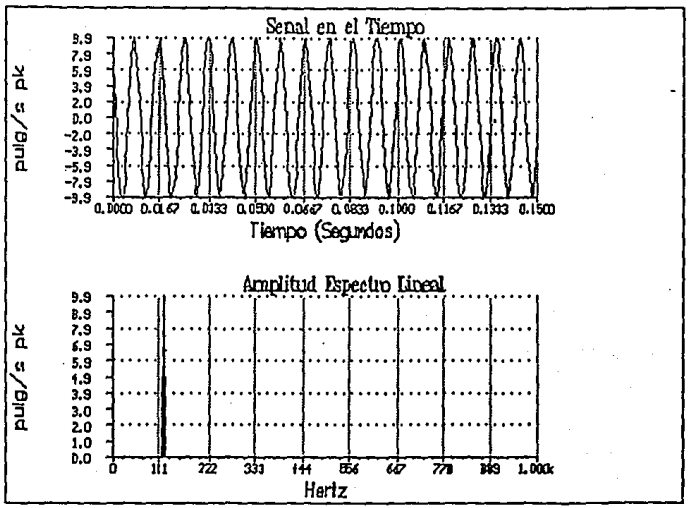

FIGURA 1.10 ESPECTRO LINEAL Y SENAL EN EL TIEMPO

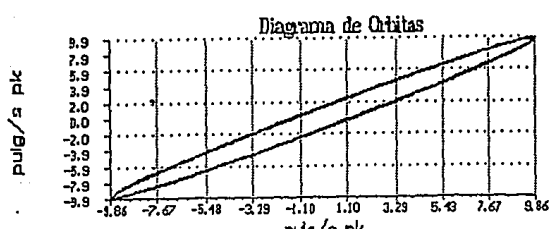

FIGURA 1.11 ORBITAS

f) Espectros en Cascada

Desplegado típico en el análisis de vibraciones que, permite observar la variación de las componentes espectrales a lo largo del tiempo (figura 1.12).

g) Evolución de Componentes.

Desplegado que contiene la evolución en el tiempo de una componente de vibración de un punto.

h) Evolución de un Equipo

Desplegado que contiene la evolución de todos los puntos, tanto de vibración como de captura manual, definidos para un equipo. Permite visualizar rápidamente como han cambiado las variables de un equipo.

i) Espectros en un Equipo

Desplegado que contiene los espectros de todos los puntos de un equipo, que hayan sido configurados como señal y adquiridos durante un recorrido. Permite visualizar las componentes de vibración en los diferentes puntos de un equipo, dando una idea primaria acerca de donde puede provenir un problema: de la unidad motriz, acoplamientos o de la unidad movida.

j) Diferencia de Espectros

Desplegado que contiene la diferencia entre dos espectros de un punto de medición de vibraciones. Los espectros general

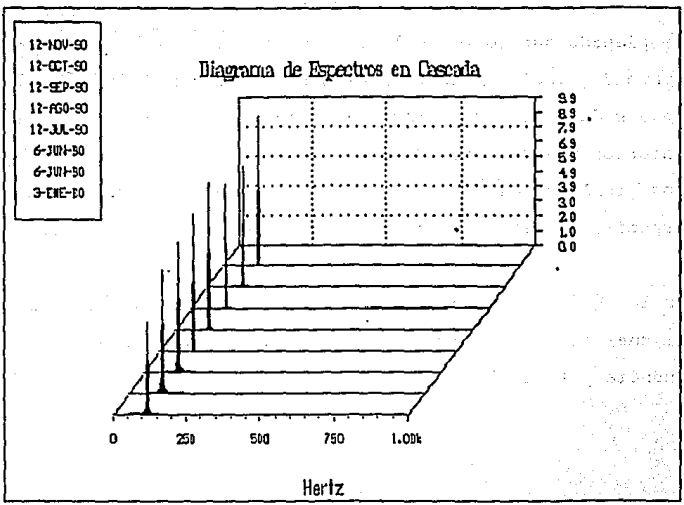

FIGURA 1.12 DIAGRAMA DE ESPECTROS EN CASCADA

mente provienen del último recorrido y de un recorrido anterior. Esta gráfica indica los incrementos o decrementos que han ocurrido en las diferentes componentes de frecuencia, así como la magnitud de dichos incrementos.

k) Evolución del nivel RMS por bandas de frecuencia. Desplegado que contiene la evolución del nivel RMS para un punto de medición de vibraciones. Se pueden definir hasta 4 bandas de frecuencia (las cuales se definen por sus límites inferior y superior). Esta opción es útil debido a que algunos problemas de vibraciones no se presentan a una frecuencia específica, sino que se presentan en un ancho de banda.

En las gráficas espectrales se dispone de un cursor gráfico, el cual al activarse permite al usuario visualizar la frecuencia y la amplitud de cualquier componente.

CAPITULO II.- EQUIPO PORTATIL DE ADQUISICION DE DATOS

2.1 Descripción General:

El Equipo Portátil de Adquisición de Datos, no es simplemente una unidad capaz de censar y almacenar información por períodos prolongados (modo fuera de línea), sino que además, es un instrumento de monitoreo continuo (modo en línea).

Debido a la energización por medio de baterías recargables, y lo reducido de tamaño y peso, este equipo resulta una herramienta de gran utilidad para cubrir recorridos extensos de adquisición de señales.

Los datos que adquiere el equipo cuando se encuentra en el modo de operación fuera de línea, pueden ser de tres tipos: de señales de vibración, de temperatura o de adquisición manual.

La adquisición de señales de vibración puede realizarse mediante transductores de desplazamiento, de velocidad o de aceleración.

Un paso de integración opcional permite convertir señales de velocidad a desplazamiento, así como las señales de aceleración a velocidad según el tipo de señal o parámetro que requiera el usuario.

El equipo cuenta con dos canales de entrada de señal de vibración; los cuales pueden ser adquiridos simultáneamente, además de un canal para la entrada de la señal de referencia

o tacómetro. La adquisición consiste en obtener simplemente el valor global o pico a pico de la vibración y el valor rms o bien muestrear la señal en el tiempo y/o en el dominio de la frecuencia.

La adquisición de señales de temperatura consiste en la obtención de las lecturas mediante una punta de prueba basada en un termopar, el cual se coloca directamente sobre la superficie a censar. El equipo entrega el valor capturado en grados centígrados.

La adquisición manual de un valor consiste en la posibilidad de introducir directamente datos en el equipo portátil; ésta información puede ser proveniente de barómetros, manómetros, o cualquier otro medidor local instalado en el equipo que se está monitoreando.

Una serie de menúes orientan al usuario en la operación del equipo, los cuales, al igual que los datos obtenidos de las mediciones y las gráficas correspondientes a la señal en el tiempo y al espectro lineal, se despliegan en un módulo LCD ("Liquid Crystal Display") o Display de cristal Líquido.

Cuando el equipo está trabajando en el modo en línea, éste opera como un instrumento de monitoreo continuo, que proporciona al usuario una serie de gráficas en el tiempo o en la frecuencia, que muestran el comportamiento del fenómeno vibratorio.

Los recorridos se programan en el equipo desde una microcomputadora tipo personal y a través de una interfaz serie,

mismo puerto por el cual se descarga la información medida una vez que se concluye una ruta de adquisición.
Los componentes principales a nivel hardware que conforman el Equipo Portátil de Adquisición de Datos son:

1) Tarjeta "Little Board" / 286.
2) Convertidor Analógico / Digital.
3) Generador de pulsos programable.
4) Circuito Censor de Temperatura.
5) Filtro Programable.
6) Display de Cristal Liquido.
7) Teclado de Membrana.

2.1.1 Tarjeta Little Board

La tarjeta Little Board /286 (LB286) proporciona a los diseñadores de sistemas, un computador de una sola tarjeta compatible con las microcomputadoras AT.
En un espacio no mayor al que ocupa una unidad de disco de 5 1/4 pulgadas, la tarjeta LB286 es funcionalmente equivalente a la tarjeta madre de una PC/AT con sus tres o cuatro tarjetas de expansión.
La LB286 está especialmente diseñada para adecuarse a las aplicaciones donde el "software", el hardware y el bus de las PC's IBM Pc y AT, requieren de bajo consumo de potencia, tamaño reducido y amplio rango de temperatura de operación.

En adición a esto, la tarjeta LB286 cuenta con la opción del uso de memorias EPROM y memoria no volátil RAM que pueden sustituir a las unidades normales de disco para ejecución de los programas de aplicación.

Las bases para las EPROM tienen capacidad para dos circuitos de 156Kb cada uno, mismos que están montados en la tarjeta.

La capacidad de la RAM es de 1MB, espacio en el cual puede accesarse el sistema operativo MSDOS de la EPROM o de las unidades de disco flexible.

Tipicamente, la tarjeta "little board" /286 es aplicable en equipos tales como:

1) Adquisición de Datos y Control.
2) Instrumentos Portátiles.
3) Conversión de Protocolos.
4) Telecomunicaciones.
5) Sistemas de Seguridad.
6) Terminales Inteligentes.
7) Estaciones sin disco flexible.
8) Entradas remotas de datos.
9) Despachadores de redes.
10) Procesos Distribuídos.
11) Cualquier otra aplicación que requiera bajo consumo de potencia; debido a que la mayoría de los circuitos integrados utilizados en el diseño de la tarjeta son de tecnología CMOS.

Los controladores de los sistemas periféricos como son los de la impresora, teclado, bocina e interfaz de unidad de discos, están igualmente incluídos en la misma tarjeta.
Otra interfaz incluída es la SCSI/BIOS ("Small Computer System Interfaz") utilizada para controlar una gran variedad de periféricos intercambiables como pueden ser unidades de disco duro, unidades de cinta, etc.
Los orificios de montaje se localizan directamente en la superficie de la tarjeta. Estos orificios convergen con los de los módulos de expansión como es el caso del módulo de controladores de video. Estos controladores al igual que las tarjetas de video son de cualquier tipo compatible con los estandares de IBM: EGA, CGA, MDA, Hércules etc.

Las conexiones con los módulos de expansión se realizan mediante cables de tipo listón.

2.1.1.1 Especificaciones:

1) Microprocesador 80C286.
2) Velocidad de 12 o 16 MHz .
3) Memoria de 512 Kb, 1Mb, 2 Mb, 4Mb.
4) Controlador de unidades de disco flexible de 5 ¼ ó de 3 ½ pulgadas.
5) 2 Puertos Serie RS-232-C.
6) 1 Puerto Paralelo con lineas de bidireccionamiento.

7) 1 Puerto de Teclado.

8) 1 Puerto de Bocina.

7) Batería de litio para reloj de tiempo real.

8) Soporte BIOS para formatos Estandar de 360 Kb, 720 Kb, 1.2 Kb y 1.4 Kb.

9) Dimensiones; 8.0" x 5,75" x 1.1".

10) Alimentación; +5 volts. +/- 5% a 1.6 A.

11) Temperatura para almacenamiento; - 55 a + 85 °C.

12) Temperatura para operación; 0 a 60 °C.

2.1.1.2 Configuración del Sistema

Debido a que la tarjeta "Little Board"/286 es funcionalmente idéntica a la tarjeta madre de una microcomputadora IBM PC, toda vez que se refiera a dicha microcomputadora, los datos son igualmente aplicables a la tarjeta LB/286.
Las cinco áreas en las cuales está dividida la tarjeta principal del sistema son las siguientes:

- CPU Microprocesador: 80C286
- Memoria de lectura solamente (ROM): Intérprete de Basic, Sistema Operativo, Autoprueba, habilitadores de entrada/salida, caracteres y patrones de graficación.
- Memoria de lectura y escritura o memoria "Random" (RAM): 512 KB.
- Adaptadores de entrada/salida: Audiocassette, teclado, bocina, puertos de comunicación serie y paralelo.

- Canales de entrada/salida : Para expander las capacidades de IBM PC, la tarjeta madre cuenta con "sockets" para conectores de 64 pines, los cuales se utilizan para las tarjetas de interfaz. Las funciones de cada pin se muestran en las tablas 2.1a y 2.1b:

2.1.1.3 CPU (Unidad Central de Procesos):

a) Generalidades

En el año de 1978, Intel Corporation, introduce al mercado el primer microprocesador de 16 bits de alto rendimiento: intel 8086/8088.

Los microprocesadores 8086 y 8088 son una extensión lógica del popular 8080 y el 8085. El 8086 y el 8088, son internamente iguales, pero el 8088 está diseñado para trabajar con un "bus" externo de 8 bits, siendo de esta manera compatible con la mayoría de los canales de este tamaño. El 8086, por el contrario, se conecta a un bus de 16 bits.

Para resolver las necesidades de velocidad, se crearon tres diferentes versiones para el 8086: a 5, 8 y 10 MHz, mientras que para el 8088 existen circuitos que operan a 5 y 8 MHz.

El procesador 8086 cuenta con un bus multiplexado de direcciones y datos, esto es, el mismo conjunto de líneas, pero en periodos de tiempo distintos, para enviar conjuntos de

PIN	SIGNAL NAME	FUNCTION	I/O/OUT
B1	GND	Ground	---
B2	RESET	System reset signal	out
B3	+5V	+5 volts power	---
B4	IRQ9	Interrupt request 9	in
B5	-5V	To J4 pin 9	---
B6	DRQ2	DMA request 2	in
B7	-12V	To J4 pin 10	---
B8	-OWS	Zero wait state	in
B9	+12V	To J4 pin 8	---
B10	GND	Ground	---
B11	-SMEMW	Memory Write (lower 1MB)	I/o
B12	-SMEMR	Memory Read (lower 1MB)	I/o
B13	-IOW	I/O Write	I/o
B14	-IOR	I/O Read	I/o
B15	-DACK3	DMA Acknowledge 3	in
B16	DRQ3	DMA Request 3	out
B17	-DACK1	DMA Acknowledge 1	out
B18	DRQ1	DMA Request 1	in
B19	-REFRESH	Memory Refresh	I/o
B20	CLK	CPU clock (e.g. 12 MHz)	out
B21	IRQ7	Interrupt Request 7	in
B22	IRQ6	Interrupt Request 6	in
B23	IRQ5	Interrupt Request 5	in
B24	IRQ4	Interrupt Request 4	in
B25	IRQ3	Interrupt Request 3	in
B26	-DACK2	DMA Acknowledge 2	out
B27	T/C	DMA Terminal Count	out
B28	BALE	Address latch enable	out
B29	+5V	+5 volts power	---
B30	OSC	14.3 MHz clock	out
B31	GND	Ground	out
B32	GND (*)	Ground	out

(*) Note: B32 is an added ground. It can be left unconnected when using standard backplanes or cards.

TABLA 2.1.a PUERTOS DE EXPANSION PARA LA TARJETA LB/286

PIN	SIGNAL NAME	FUNCTION	I/O/OUT
A1	-I/O CH CK	Memory parity error	in
A2	D7	Data bit 7	I/o
A3	D6	Data bit 6	I/o
A4	D5	Data bit 5	I/o
A5	D4	Data bit 4	I/o
A6	D3	Data bit 3	I/o
A7	D2	Data bit 2	I/o
A8	D1	Data bit 1	I/o
A9	D0	Data bit 0	I/o
A10	I/O CH RDY	Processor Ready Ctrl	in
A11	AEN	Address Enable	I/o
A12	A19	Address bit 19	I/o
A13	A18	Address bit 18	I/o
A14	A17	Address bit 17	I/o
A15	A16	Address bit 16	I/o
A16	A15	Address bit 15	I/o
A17	A14	Address bit 14	I/o
A18	A13	Address bit 13	I/o
A19	A12	Address bit 12	I/o
A20	A11	Address bit 11	I/o
A21	A10	Address bit 10	I/o
A22	A9	Address bit 9	I/o
A23	A8	Address bit 8	I/o
A24	A7	Address bit 7	I/o
A25	A6	Address bit 6	I/o
A26	A5	Address bit 5	I/o
A27	A4	Address bit 4	I/o
A28	A3	Address bit 3	I/o
A29	A2	Address bit 2	I/o
A30	A1	Address bit 1	I/o
A31	A0	Address bit 0	I/o
A32	GND (*)	Ground	---

(*) Note: A32 is an added ground. It can be left unconnected when using standard backplanes or cards.

TABLA 2.1b PUERTOS DE EXPANSION PARA LA TARJETA LB/286

señales diferentes. Esta técnica tiene como fin lograr una mayor eficiencia en el uso del encapsulado estandar de 40 pines.

b) Funciones Internas

Cada microprocesador está dividido, a su vez, en dos sub-procesadores según las funciones internas que realizan.

Estas subdivisiones son: La unidad de ejecución ("Execution Unit") o EU, y la unidad interfaz del bus ("BIU Bus Inferface Unit").

La EU realiza las operaciones, mientras que la BIU accesa al microprocesador las instrucciones y los datos.

La BIU extrae las instrucciones de la cola de instrucciones conforme se van necesitando, al mismo tiempo que la EU va ejecutando las instrucciones anteriores.

El concepto de cola, se define como la línea de espera de las instrucciones que llegan al procesador al tiempo que otra está siendo ejecutada. Este sistema tiene la ventaja de que cada instrucción puede extraerse de memoria mientras otras están ejecutándose, así el tiempo de proceso es menor. Para el caso del 8086 la cola es de 6 bytes u octetos y para el 8088 es de 4.

c) Registros

El microprocesador 8086/8088 cuenta con 14 registros; cada uno de ellos de dos octetos o 16 bits.

Cuatro registros son para uso general y son denominados con las letras AX, BX, CX y DX. Cada registro puede subdividirse, debido a que cuenta con dos octetos, en registro alto y bajo, es decir; AH y AL, BH y BL, CH y CL etc.. Donde la nomenclatura será "X" para registro completo (16 bits), "H" para parte alta del registro (8 bits) y "L" para la parte baja del registro (8 bits).

Las características de los registros de uso general A, B, C y D son semejantes; no obstante, algunos registros producen un código de máquina radicalmente diferente (mas corto) por ello algunas veces resulta mas eficiente usar un determinado registro para operaciones específicas; así el registro A se utiliza para realizar operaciones de almacenamiento inmediato en el acumulador, el registro B como registro base para los direccionamientos, el C para funciones de contador en casos como bucles, iteraciones, rotaciones, etc., y el D para manejo de Datos.

Los registros SP, BP, SI y DI son conocidos como registros punteros y de índice. Los registros de índice fuente (SI), índice destino (DI) y el puntero base (BP) se utilizan como parte de los modos de direccionamiento, es decir para ayudar en la localización de datos.

Existe un registro de indicadores de 16 bits que contiene varios bits de estado para el procesador. Estos incluyen: Indicador de cero (ZF), indicador de signo (SF), indicador de paridad (PF), indicador de acarreo (CF), indicador auxi-

liar (AF), indicador de dirección (DF), indicador de interrupción (IF), indicador de desbordamiento (OF) e indicador de desvío (TF).

Los cuatro registros de segmentación son: CS, DS, SS y ES y sus códigos representan a los registros segmento de código, datos, pila y extra respectivamente.

El puntero de instrucciones (IP) tiene como actividad señalar la instrucción que se está procesando dentro del circuito. Los registros punteros de instrucciones (IP) y puntero de pilas (SP), se encargan del control del flujo propio del programa.

d) Segmentación de la memoria

La segmentación es un método de acceso a memoria en el cual toda dirección se compone de dos cantidades; un identificador de segmento y un desplazamiento. El identificador de segmento, apunta a un área general de memoria (segmento), mientras que el desplazamiento apunta a una dirección dentro del segmento.

Este método es necesario para accesar correctamente a un megaocteto completo de memoria con referencias de direcciones de tan solo 16 bits.

El desplazamiento de 16 bits se compone de varias partes: un decalaje (un número fijo), una base (almacenada en el registro base) y un índice (almacenado en el registro índice). La base y el índice son variables dinámicas que pueden o no ser

utilizadas para calcular la dirección real, pero que son muy útiles para gestionar matrices de dos dimensiones o estructuras internas a otra estructura. El decalaje se utiliza para compilar datos, reorganizar la memoria y reubicar mas rápida y fácilmente; se fija el tiempo de ensamblaje (paso de código fuente a código máquina) y no puede cambiarse durante la ejecución del programa. La dirección del segmento se almacena en uno de los cuatro registros de segmentación (CS,DS,ES,OS). El procesador usa estas dos cantidades para calcular la dirección real de 20 bits según la fórmula 2.1.

Dirección Real = 16 * (dirección del segmento) +
Desplazamiento ..... (2.1)

2.1.1.4 Interfaz

Al proceso por el cual se realiza la comunicación entre la computadora y un dispositivo externo, se le conoce con el nombre de interfaz.

Este proceso está basado principalmente en dos conceptos: aumento del poder de conexión de los canales de comunicación ("Bus Buffering") y decodificación de direcciones ("Address decoding").

El primer concepto se refiere a la conexión de una circuitería especial en un canal especial de comunicación o "Bus". Esto se realiza con el fin de poder conectar varios dispositivos externos al "Bus" sin causar problemas en la habilita-

ción de cada uno de ellos.
Básicamente existen dos tipos de "buffers": Los habilitadores/receptores y los transmisores.

a) Aumento de poder de conexión
Un habilitador es capaz de proporcionar la cantidad de corriente necesaria para accionar un número determinado de dispositivos simultáneamente.
Un "Buffer" receptor, proporciona inmunidad al ruido o a un estado de histéresis lo cual reduce las posibilidades de que se reciban datos erróneos.
El circuito integrado 74LS241 combina las operaciones de un habilitador y de un receptor.
Un "buffer" transmisor, pasa la información por cualquier canal de comunicación o "bus" en ambas direcciones. La figura 2.1 muestra el uso de los "buffers" en varios sistemas simultáneos de canales de operación "bus".

b) Decodificación de direcciones
Cuando la computadora desea comunicarse con un dispositivo externo, ésta debe saber la dirección del dispositivo. La computadora comenzará por enviar dicha dirección, una vez que la dirección es puesta en el "bus" de direcciones se activa el decodificador. El decodificador es un circuito especial, que monitorea las líneas de dirección para una dirección en particular o para un rango de direcciones. Cuan-

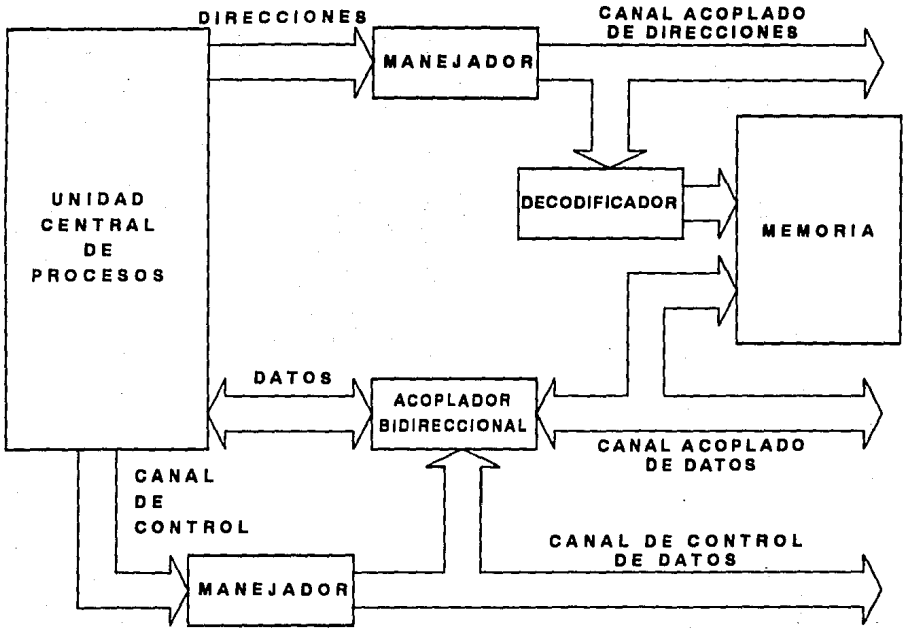

FIGURA 2.1 USO DE LOS BUFFERS EN UN SISTEMA DIGITAL.

do la dirección aparece en el "bus", el decodificador de direcciones genera un pulso de selección del dispositivo, el cual es usado para seleccionar el dispositivo con el cual deseamos comunicarnos.
La mayoría de los decodificadores de direcciones son interruptores seleccionables.

c) Contención de "Buses"
Cuado trabajamos con decodificadores de direcciones debemos estar seguros de que bajo ninguna circunstancia dos dispositivos externos tengan la misma dirección, ya que pueden presentarse problemas en la habilitación de éstos.
Los microprocesadores pueden comunicarse con los dispositivos externos de entrada y salida de dos maneras: seleccionando direcciones de memoria o como puertos de entrada/salida.
El primer tipo de comunicación es también conocido como mapeo de memoria de entrada/salida; tal efecto se puede lograr solamente si el microprocesador cuenta con un "bus" de entrada/salida tal es el caso del 8080,8085,8086,8088 o del 80286.
En el caso de la computadora IBM PC el decodificador es muy sencillo debido a que solamente se tienen que decodificar 16 líneas de direcciones del "bus" de entrada/salida, (65536 direcciones)

d) Entrada/salida y mapeo de memoria
Cuando se trabaja con el método de mapeo de memoria de entra-

da/salida, es necesario conocer las direcciones de memoria utilizables en la interfaz, para lo cual se utilizan mapas sencillos, con la descripción de cómo son utilizados en el sistema. Los mapas de entrada/salida para la IBM PC se muestran las figuras 2.2a y 2.2b.

e) Comandos de "Software"

Además de los dispositivos de "hardware", necesitamos de comandos de "software" que controlen dichos circuitos integrados para completar el proceso de interfaz. Las computadoras se comunican con las direcciones de memoria mediante los comandos in y out del lenguaje de máquina o ensamblador.

2.1.2 Convertidor Analógico / Digital

2.1.2.1 Descripción

El AD1334 es un sistema de conversión analógico/digital de 12 bits, en un circuito integrado.

El dispositivo consiste en cuatro amplificadores independientes, un multiplexor, un convertidor analógico/digital, un controlador, un buffer de 32 palabras y una interfaz digital asíncrona de alta velocidad.

El controlador de canales habilita los cuatro canales independientes del AD1334 para la captura de lecturas de 12 bits de información analógica.

La escala de conversión del A/D es de -5 a +5V a oscilacio

Memory Address	Function
FE0000-FFFFFFh	Duplicates onboard memory at 0E0000-0FFFFFh.
100000-FDFFFFh	"Extended memory" space. 384K or more can be supplied by onboard DRAM, if 1MB or more of DRAM is installed onboard, and jumpers are set to enable onboard extended memory.
0F0000-0FFFFFh	ROM-BIOS sockets at U10 and U11. U10 contains "odd" address byte, and U11 contains "even" address byte. Contains a startup vector at address 0FFFF0h.
0E0000-0EFFFFh	Duplicates onboard memory at 0F0000-0FFFFFh. Reserved for future expansion.
008000-00FFFFh	Byte-wide memory socket at U33.
000000-007FFFh	Byte-wide memory socket at U32.
0C0000-0CFFFFh	Reserved for expansion bus ROM's.
0A0000-0BFFFFh	Normally contains video RAM, as follows: CGA Video: B8000 - BFFFFh Monochrome: B0000 - B7FFFh EGA Video: A0000 - AFFFFh
080000-09FFFFh	128k bytes onboard DRAM (optional).
000000-07FFFFh	512K bytes onboard DRAM.

I/O Address	Function
03F8 - 03FFh	Primary serial port
03F0 - 03F7h	Floppy disk controller ports (WD37C65) 3F2 - FDC Digital output register (LDOR) 3F4 - FDC Main status register 3F5 - FDC Data register 3F7 - FDC Control register (LDCR)
0300 - 030Fh	CGA display adapter (option)
03C0 - 03CFh	EGA display adapter (option)
03B0 - 03BFh	Monochrome display adapter (option)
03A0 - 03AFh	Reserved for primary bisync port
0380 - 038Fh	Reserved for secondary bisync port
0378 - 037Ah	Primary parallel printer port
0372 - 0373h	Configuration EEPROM access register
0370 - 0371h	SCSI controller ports (AIC-6250)
0360 - 036Fh	Reserved
0300 - 031Fh	Reserved for prototype card
02F8 - 02FFh	Secondary serial port
0278 - 027Ah	Secondary parallel printer port
0200 - 0207h	Reserved for game port
01F0 - 01F8h	Reserved for AT bus hard disk controller
00F0 - 00FFh	Math coprocessor (80287) 00F1 - Reset 00F0 - Clear busy
00C0 - 000Fh	DMA controller 2 (8237 equivalent)
00A0 - 00BFh	Interrupt controller 2 (8359 equivalent)
0080 - 009Fh	DMA page registers (74LS612 equivalent)
0070 - 007Fh	Real time clock and NMI mask
0060 - 006Fh	Keyboard controller (8042 equivalent)
0040 - 005Fh	Programmable timer (8254 equivalent)
0020 - 003Fh	Interrupt controller 1 (8359 equivalent)
0000 - 001Fh	DMA controller 1 (8237 equivalent)

2.2a y 2.2b MAPAS DE ENTRADA/SALIDA PARA LA IBM PC

nes superiores a los 67 KHz para un solo canal en operación y 46 KHz para dos canales simultáneos.

2.1.2.2 Registro de Control

a) Registro de control

CTLEN	L/F	HF/F	ORNG	X	X	X	X
7	6	5	4	3	2	1	0 BIT

Figura 2.3

La palabra de control para el registro de control del convertidor analógico/digital se muestra en la figura 2.3.

7 CTLEN

Un cero en esta posición de bit deshabilitará al controlador.

6 L/F

Un cero en esta posición de bit desactivará el buffer de almacenamiento de conversiones y activará el bit de interrupción en el status cada vez que se efectúe una conversión.

5 HF/F

En caso de que el buffer esté activo, un cero en esta posición hará que éste se deshabilite cada 16 conversiones.

4 ORNG

Un cero en ésta posición inhabilita la capacidad de inte-

rrupción de almacenamiento en el buffer en el caso de producirse un sobreflujo.

3-0 X

Idefinidos.

2.1.2.3 Registro de Estatus.

La palabra de control del registro de estatus del convertidor analógico/digital se muestra en la figura 2.4

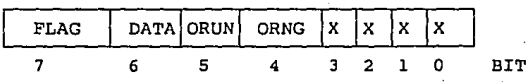

Figura 2.4

7 FLAG

Operación lógica OR entre los bits 4 y 6.

6 DATA

Enciende cada vez que el convertidor detecta 1, 16 o 32 conversiones por realizar.

5 ORUN

Se enciende cuando el buffer de almacenamiento de conversiones recibe mas información de la que puede guardar.

4 ORGN

Se enciende cuando a la entrada del convertidor se detecta sobreflujo.

3-0 X

Indefinidos.

2.1.3 Generador de Pulsos Programable

2.1.3.1 Descripción

El circuito COP452L de la firma National Semiconductor Co. es un generador de frecuencias y contador programable fabricado a base de tecnología MOS de canal N de silicio.

Este circuito integrado cuenta con 10 instrucciones de programación y 11 modos de operación.

El bit mas significativo es siempre un "1" y se conoce como el bit de inicio. El segundo bit mas significativo define la comunicación como una instrucción o un modo.

Los cuatro bits menos significativos contienen los comandos del dispositivo.

2.1.3.2 Instrucciones de Programación del COP

Las instrucciones de programación del generador de pulsos se muestra en la tabla 2.2.

1) Llamada de registros.

El registro seleccionado es llamado con 16 bits de datos en la entrada serial de datos (DI).

2) Lectura de registros.

El dato del registro seleccionado es colocado a la salida

serial de datos (DO) al mismo tiempo que el dato es reciclado al registro.

INSTRUCCION	PALABRA DE CONTROL MSB LSB	COMENTARIO
LDRB	100000	Llamada al registro B de Dl
LDRA	100001	Llamada al registro A de Dl
RDRB	100010	Lee el registro B de DO
RDRA	100011	Lee el registro A de DO
TRCB	100100	Transfiere el registro B al contador B
TRCA	100101	Transfiere el registro A al contador A
TCRB	100110	Transfiere el contador B al registro B
TCRA	100111	Transfiere el contador A al registro A
CK1	101000	CKl dividido entre 1
CK4	101001	CKl dividido entre 4
LDM	11xxxx	Modo de operación

Tabla 2.2

3) Llamada al contador

El contenido del registro seleccionado es transferido a su contador asociado. El contenido del registro no se afecta.

4) Copia del contador

El contenido del contador seleccionado es transferido a su registro asociado. El contenido del contador no se afecta.

5) CKI dividida entre 1

El divisor del oscilador en la entrada del cristal (CKI) es inicializado a dividir entre uno. La frecuencia interna es por tanto igual a la frecuencia del cristal. Esta instrucción no debe ser usada si la frecuencia del CKI es mayor a la frecuencia máxima interna.

6) CKI dividida entre 4

El divisor del oscilador en la entrada del cristal (CKI) es inicializado para dividir entre cuatro. La frecuencia interna es por tanto la cuarta parte de la frecuencia del cristal. Esta instrucción no debe ser usada si la frecuencia del CKI es menor que cuatro veces el valor de la frecuencia mínima interna.

7) Modo de operación .

El modo de operación se define a partir de los cuatro bits menos significativos.

2.1.3.3 Modos de Operación

Los modos de operación del generador de pulsos programable se muestran en la tabla 2.3:

MODO DE OPERACION	PALABRA DE CONTROL MSB LSB
Reset	111111
Frecuencia Dual	110000
Frecuencia y conteo	110100
Conteo Dual	110101
Número de pulsos	110010
Ciclo de Trabajo	110011
Forma de onda	110110
Pulso disparado	110001
Pulso disparado y conteo	110111
Ruido blanco y frecuencia	111000

Tabla 2.3

1) Reset

Este modo de operación inicializa las salidas de los contadores A y B (OA y OB) en 0.

2) Frecuencia Dual

Dos frecuencias son generadas: una a la salida de cada contador. El período de la onda cuadrada en la salida del contador A es determinada por el registro del contador A y lo mismo ocurre para el contador B.

En los modos de generación de frecuencia, los contadores regresan hasta 0, en ese punto la salida pulsa y automáticamente los contadores llaman sus respectivos registros.

3) Frecuencia y conteo

Una frecuencia sencilla sale por OA. El contador B cuenta pulsos internos desde la entrada (INB).

4) Cuenta Dual

En este modo los contadores A y B son habilitados como contadores de eventos externos.

5) Número de Pulsos

En este modo el contador A entrega un número específico de pulsos de un ancho determinado por OA.

6) Ciclo de Trabajo

Este modo entrega ondas rectangulares por OA cuyo ancho y alto es especificado en el registro A.

7) Forma de onda

Este modo mide el tiempo de estado alto y de estado bajo de una onda externa a la entrada del contador B.

8) Pulso disparado

Este modo entrega un pulso disparado cuando la señal senoidal de entrada (ZI) cruza por cero. El retardo del cruce por cero se especifica en el registro A. El ancho del pulso se define en el registro B.

9) Pulso disparado y conteo

Este modo realiza la misma operación que el anterior pero además, el contador B cuenta los eventos externos.

2.1.4 Circuito Sensor de Temperatura

2.1.4.1 Descripción

El circuito AD594 es un circuito acondicionador para señales de termopar que realiza las funciones de amplificación y compensación de junta de referencia, entregando una señal eléctrica de salida de 10 mV por cada grado centígrado de temperatura.

Este circuito ha sido precalibrado para operar con un sensor tipo J (hierro-constantan). La fuente de alimentación es +5V y con voltajes negativos puede proporcionar lecturas de temperatura inferiores a los 0 grados.

El circuito AD594 funciona como dos amplificadores diferenciales. Las salidas son sumadas y usadas para controlar un amplificador de alta ganancia.

Bajo condiciones normales de operación, la salida del amplificador principal, en el pin 9, se conecta a la red de retroalimentación (pin 8). Las señales provenientes del termopar en los pines 1 y 14 son amplificadas con una ganancia G del amplificador diferencial y posteriormente con una ganancia A en el amplificador principal. La salida del amplificador principal es retroalimentada a un segundo amplificador diferencial en conección invertida.

Los dos amplificadores diferenciales están diseñados para proporcionar ganancias idénticas G. Como resultado de lo anterior, la señal de retroalimentación aplicada al amplificador de la derecha en la figura 2.5, y la señal de entrada del termopar se restarán hasta que la diferencia sea cero, en ese momento, el pin de salida entregará 10mV / grado C.

2.1.4.2 Adquisición de Temperatura con el AD594

En lo que concierne a la programación de este dispositivo, el Equipo Portátil de Adquisición de Datos, habilita o deshabilita el canal del convertidor analógico/digital al cual está conectado el sensor de temperatura.

2.1.5 "Display" de Cristal Líquido

2.1.5.1 Descripción

El Display de Cristal Líquido utilizado en el equipo Portátil de Adquisición de Datos es el LM238B y el circuito controla

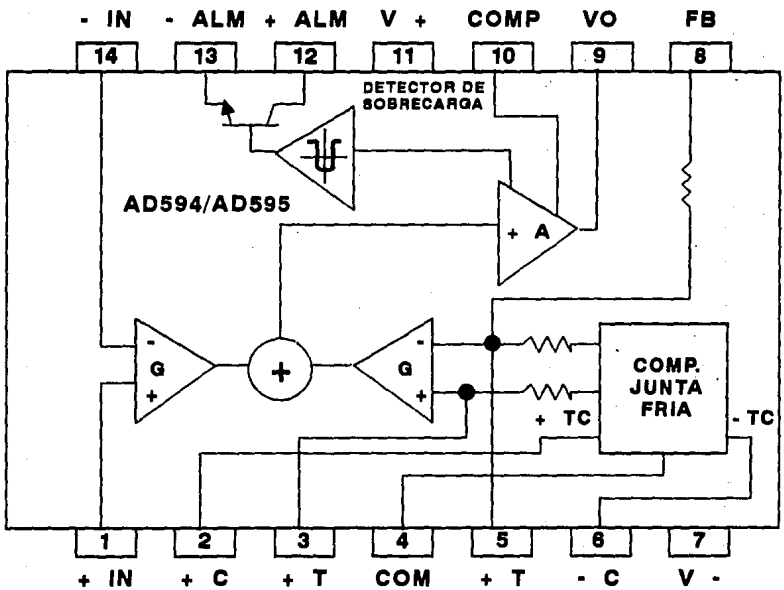

FIGURA 2.5 PRINCIPIO DE OPERACION DEL CIRCUITO AD594.

dor es el HD61830, ambos de la firma Hitachi Inc.

El HD61830 es un controlador para displays del tipo "dot matrix" (matriz de puntos) con capacidad gráfica, el cual almacena los datos del display enviados desde la computadora. La memoria utilizada es una RAM externa que posteriormente generará las señales que controlarán al display.

En el modo gráfico, el dato proveniente de la RAM prende o apaga un punto o "pixel" en el display, mientras que en el modo alfanumérico o caracter se accesa una matriz de ocho bytes que contienen un caracter previamente definido en la ROM en código ASCII.

Ambos modos pueden ser utilizados para múltiples aplicaciones, con la restricción de que no se pueden utilizar ambos modos en una misma pantalla.

El controlador maneja cinco tipos de registros:

IR registro de Instrucción

DIR Registro de entrada de datos

DOR Registro de salida de datos

DR Registro de Puntos

MCD Registro de control de modo

El registro IR es un registro de cuatro bits, en los cuales se almacena el código de la instrucción. En los otros cuatro bits (DB0 a DB3) se almacena el dato que se desea cargar.

El registro DIR es un registro de ocho bits usado para almacenar temporalmente el dato escrito en la RAM externa en DR,

MCR.

El Registro DOR es un registro de ocho bits usado para almacenar temporalmente el dato leído de la RAM externa.

El registro DR es un registro utilizado para almacenar la información de los puntos tales como caracteres, No de puntos verticales, etc.

El registro MCR es un registro de 6 bits usado para almacenar los datos que especifican los estados del LCD tales como si está prendido o apagado el display, el cursor destellando o no, etc.

Además existe una bandera de ocupado BF la cual indica que una instrucción se está llevando a cabo, por lo cual la siguiente operación no podrá ser realizada mientras ésta bandera esté prendida.

2.1.5.2 Instrucciones de Programación del LCD

El Display se controla mediante la escritura de un dato en el registro de instrucciones y en los trece registros de datos.

Existe una señal (RS) que distingue los registros de instrucciones de los de datos. Cuando RS=1 los datos de 8 bits pueden ser escritos en los registros de instrucciones, después de esto, RS=0 el dato pasa al registro de datos y se ejecuta la instrucción.

Las instrucciones de programación para el controlador son:

1) Modo de control

Registro	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg. Ins.	0	1	0	0	0	0	0	0	0	0
Reg. MCR	0	0	0	0						

Figura 2.6

El código es 0 hexadecimal y se escribe en el registro de instrucciones. Los dos primeros bits (L/E y RS) indican si se trata de una instrucción de lectura o escritura y si se trata de una instrucción o un dato, respectivamente.
El modo de control se escribe en los seis bits del registro MCR según la figura 2.6.

2) Número de puntos verticales por caracter
Esta instrucción determina el número de renglones que aparecen por pantalla, debido a que controla el número de puntos verticales que encienden en cada caracter mediante la variable VP. Esta variable solo toma valor en el modo alfanumérico de operación del display.

HP	DB2	DB1	DB0	
6	1	0	1	6 puntos por caracter horizontal
7	1	1	0	7 puntos por caracter horizontal
8	1	1	1	8 puntos por caracter horizontal

Figura 2.7

La variable HP indica el número de bits por caracter horizontal por display. HP indica el número de bits encendidos en un byte cuando se trabaja en el modo gráfico.
Existen tres valores para HP como muestra la figura 2.7.

3) Número de caracteres por renglón
HN indica el número de caracteres horizontales en el modo alfanumérico o el número de bytes horizontales en el modo gráfico.
Si consideramos "N" la suma total de los puntos en una pantalla.

$$n = HP \times HN \quad \ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

Donde HN puede tomar valores pares desde 2 hasta 128 (decimal)

4) Control de brillo
Esta instrucción controla el brillo en la pantalla de acuerdo con la figura 2.8:

Registro	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg. Ins.	0	1	0	0	0	0	0	0	1	1
Brillo	0	0	0							

Figura 2.8

Un número comprendido entre 0 y 128 corresponde al valor Nx. Nx-1 debe programarse en los bits db0 a db5.

5) Altura a la que aparece el cursor

La variable CP indica la altura a la que aparecerá el cursor si tomamos como referencia la altura de un caracter; es decir, si tenemos un caracter de 7 x 5 puntos verticales y horizontales respectivamente, y se desea que el cursor aparezca debajo del caracter, CP tomará el valor de 8. Ver figura 2.9.

Registro	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg. Ins.	0	1	0	0	0	0	0	1	0	0
Posición	0	0	0	0	0	0				

Figura 2.9

Los valores que puede tomar Cp son del 1 al 16.

Para determinar el ancho del cursor se utiliza la variable HP.

Si la variable VP (número de puntos verticales por caracter) es menor que CP, no se desplegará el cursor.

6) y 7) Dirección de la RAM para el despliegue de la pantalla

Reg.	R/W	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
IR	0	1	0	0	0	0	1	0	0	0
0	0	Parte baja de la dirección								

Figura 2.10

Reg.	R/W	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
IR	0	1	0	0	0	0	1	0	0	0
0	0	Parte alta de la dirección								

Figura 2.11

Las instrucciones 6 y 7 trabajan juntas, su función es indicar el punto en la memoria RAM a partir del cual se desea que se desplieguen los valores en el display. Su utilidad consiste en la posibilidad de contar con varias páginas de despliegue dependiendo de la RAM que se disponga. El caracter que se encuentre en la dirección indicada por estas instrucciones parecerá en la esquina superior izquierda de la pantalla. Figuras 2.10 y 2.11

8) y 9) Dirección de la RAM donde aparecerá el cursor
Esta instrucción determina la dirección de memoria en la cual ha de aparecer el cursor. Si la dirección especificada para el despliegue del cursor no cae dentro de la región de memoria seleccionada para la pantalla, el cursor no será visible. Figuras 2.12 y 2.13.

Reg.	R/W	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
IR	0	1	0	0	0	0	1	0	1	0
Cur.	0	0	Parte baja de la dirección							

Figura 2.12

Reg.	R/W	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
IR	0	1	0	0	0	0	1	0	1	1
Cur.	0	0	Parte alta de la dirección							

Figura 2.13

Regi	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg.	0	1	0	0	0	0	0	0	0	0
Ram	0	0								

Figura 2.14

10) Despliegue de caracteres de la RAM en la dirección del cursor

Una vez que el código 0c h, es escrito en el registro de instrucciones con la bandera RS encendida, un dato de 8 bits (1byte) pero con dicha bandera en estado bajo escrito en el registro de instrucciones, transfiere de la RAM el dato correspondiente al caracter a la dirección del cursor.

El valor hexadecimal correspondiente al caracter se define en los bits DB0 a DB7. Figura 2.14

Después de esta operación la dirección del cursor se ve incrementada en una unidad.

11) Lectura de datos del display

El código 0D h en el registro de instrucciones, habilita a la RAM para leer un dato.

Esta instrucción obtiene el contenido del registro de salida

de datos (DB0 a DB7) y después transfiere el dato de la ram especificado por la dirección del cursor, al registro de salida de datos, finalmente se incrementa la dirección del cursor en uno. Figura 2.15.

Registro	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg. Ins.	0	1	0	0	0	0	1	1	0	1
Ram	1	0								

Figura 2.15

El dato correcto no es entregado en la primera lectura por lo que es preciso realizar una lectura ociosa.

12) y 13) Apagado y encendido de un bit

Registro	L/E	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Reg. Ins.	0	1	0	0	0	0	0	0	1	1
Brillo	0	0	0	0						

Figura 2.16

Estas instrucciones 12 y 13 trabajan análogamente para apagar o encender respectivamente, un bit de un determinado octeto definido en la variable NB y la dirección de memoria correspondiente a la dirección del cursor.
NB toma valores de 1 a 8, luego, DB0 a DB2 debe tomar valores equivalentes a NB - 1. Figura 2.16

La operación de ésta instrucción finaliza con el incremento en una unidad de la dirección del cursor.

2.1.6 Teclado

El teclado del Equipo Portátil de Adquisición de datos fué realizado y programado en base a una matriz de interruptores, la cual es controlada a partir del circuito integrado 8255. A continuación se describe brevemente las características principales y los modos de operación de dicho dispositivo.

2.1.6.1 Circuito Integrado 8255

El 8255 es un circuito de interfaz programable, diseñado para utilizarse con microprocesadores intel y semejantes.
Este dispositivo cuenta con 24 pines de entrada/salida los cuales pueden ser programados individualmente en 2 grupos de 12 (modo 0), en 3 grupos de 8 (modo 1) o bien bidireccionalmente (modo 2).
En el modo 0, cada grupo de 12 pines puede subdividirse en grupos de 4, los cuales pueden operar tanto para entrada como para salida.
En el modo 1, cada grupo puede programarse para tener 8 líneas de entrada o salida.
En el modo 2 se utilizan 8 lineas de interfaz bidireccional y 5 para control.

Debido a que en el Equipo Portátil de Adquisición de Datos, tanto el teclado como el display de cristal líquido requieren de este circuito, el modo de operación que se utilizó fue el modo 0.

El motivo principal de ésta elección, es la facilidad que proporciona el modo 0 para operar el 8255 con tres puertos: A, B, y C.

Los puertos A y B fueron utilizados como puertos de entrada y el C que puede ser dividido en dos puertos de entrada o salida de 4 bits cada uno, se usó como puerto de salida.

La palabra de control para el modo 0 se define como se muestra en la figura 2.17.

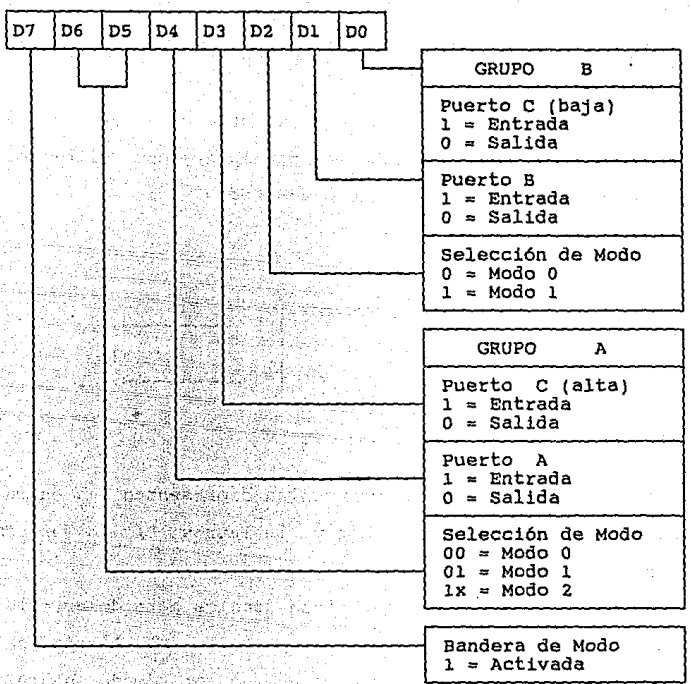
D7
D6
D5
D4
D3
D2
D1
D0
GRUPO B
Puerto C (baja)
1 = Entrada
0 = Salida
Puerto B
1 = Entrada
0 = Salida
Selección de Modo
0 = Modo 0
1 = Modo 1
GRUPO A
Puerto C (alta)
1 = Entrada
0 = Salida
Puerto A
1 = Entrada
0 = Salida
Selección de Modo
00 = Modo 0
01 = Modo 1
1x = Modo 2
Bandera de Modo
1 = Activada

Figura 2.17

CAPITULO III.- REQUERIMIENTOS DEL PROGRAMA DE CONTROL

3.1 Técnicas de Diseño

3.1.1 Cajas Negras

Una caja negra es usualmente una subrutina, la cual realiza una determinada función escondiendo completamente el proceso. Una representación común de una caja negra es la que se muestra en la figura 3.1.

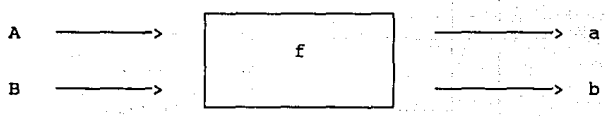

Figura 3.1

En ésta figura, las letras mayúsculas representan las entradas, las minúsculas las salidas y la letras "f" la función que se realiza.

Las ventajas que proporciona esta técnica para desarrollar programas de computación, se basan en la simplicidad del manejo de los bloques, los cuales pueden ser, desde una rutina del programa, hasta un dispositivo que opere externamente a la computadora; en cuyo caso será necesario conocer la manera de activarlo y desactivarlo.

Cuando las cajas negras son utilizadas como rutinas en un programa, y una vez que entradas, salidas y la función han sido definidas, se procede a diseñar su estructura interna

en base a dos técnicas principalmente "Bottom-Up" y "Top-Down".

3.1.2 Diseño "Bottom-Up" (de abajo hacia arriba)

Esta técnica de diseño de programas de cómputo se basa en el siguiente principio: Las funciones que realizan procesos complejos (funciones de alto nivel), se crean a partir de otras que realizan operaciones mas sencillas (funciones de bajo nivel). Figura 3.2.

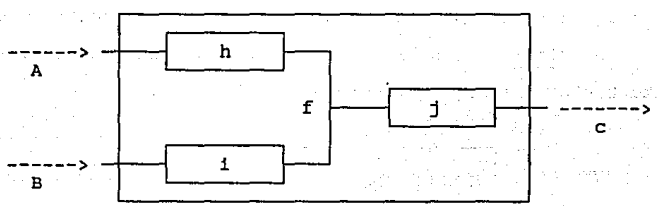

Figura 3.2

Al primer grupo de funciones pertenecen todas aquellas que no pueden defirse a partir de otras mas simples (h,i,j), y que realizan una, y solo una operación en su proceso. Debido a lo anterior a éstas funciones también se les conoce como funciones básicas.

La tarea del diseñador comienza por crear un conjunto de funciones de bajo nivel totalmente independientes unas de

las otras y, una vez logrado ésto, interconectarlas para conformar funciones de mayor nivel (f).
La principal ventaja que reporta éste método consiste en la facilidad para detectar un error debido a que cada una de las funciones básicas puede ser revisada separadamente.

3.1.3 Diseño "Top-Down" (de arriba hacia abajo).

El principio básico que encierra el diseño de "software" mediante la técnica "Top-Down", es básicamente el inverso al anterior.
El diseñador define los grandes bloques que conforman el programa, disminuyendo el nivel de las funciones paulatinamente hasta obtener las funciones mas básicas.

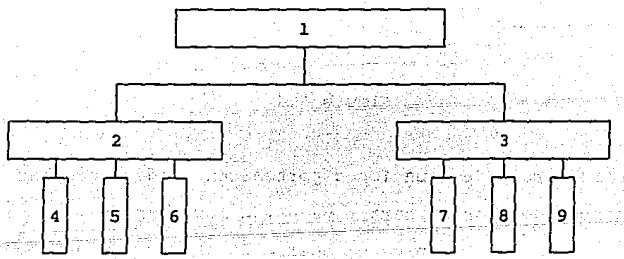

Figura 3.3

La lógica de los diagramas de arquitectura, es una herramienta muy útil y comúnmente empleada, en ésta técnica de diseño, debido a la facilidad de lectura y seguimiento de informa-

ción por parte del lector.
La figura 3.3 ilustra un ejemplo del empleo de ésta técnica, en la cual, el orden numérico también es el orden jerárquico.

3.2 Arquitectura

En las líneas anteriores se describieron las ventajas que proporcionan cada una de las técnicas mas comunes de diseño de programas de cómputo. La facilidad en la detección de errores de la primera, y la sencilla visualización de los procesos de la segunda, llevaron a la determinación de un criterio de diseño para el programa de este trabajo, que reuniera las mejores carcterísticas de cada método.
Como resultado de lo anterior, se procedió a la creación de funciones de bajo nivel de operación independiente, estructuradas en diagramas de arquitectura, definiendo los procesos de manera lógica.
Dos funciones generales se realizan en el programa principal: operación fuera de línea y operación en línea.
En el modo de operación fuera de línea, se realiza las siguientes actividades: 1) Despliegue de Parámetros, 2) Despliegue del estado y capacidad de la memoria, 3) Programación de un punto de no cargado en la ruta de adquisición, 4) Graficación, 5) Definición de comentarios por parte del operador, referentes al punto adquirido y 6) Protocolo de comunicación serial del equipo con la microcomputadora.

El diagrama de arquitectura se muestra en la figura 3.4.

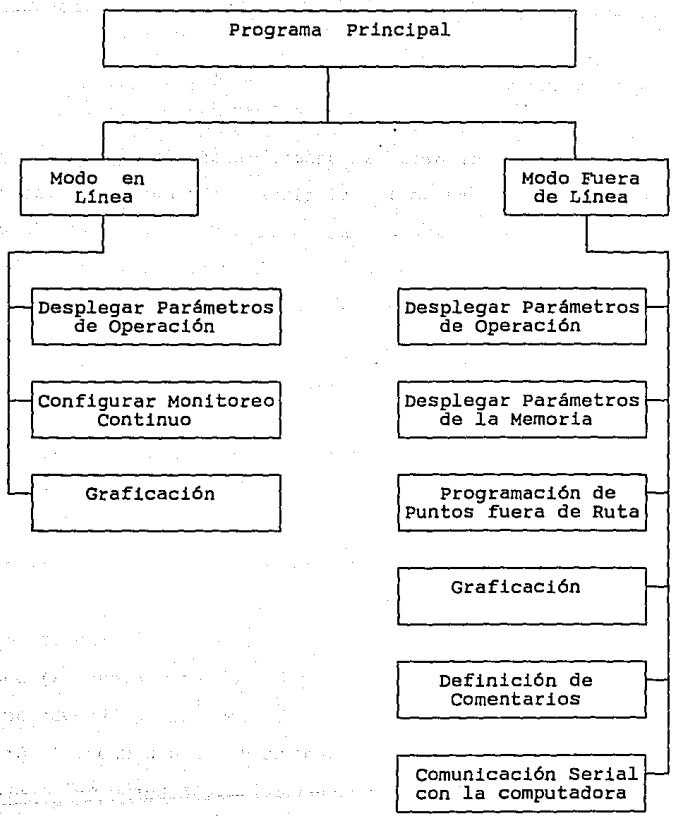

Figura 3.4

En el modo en línea, el equipo realiza los tres procesos que son: 1) Despliegue de Parámetros, 2) Configuración de Monitoreo Continuo y 3) Graficación.

3.3 Descripción Funcional y Requerimientos del Programa de control

Al activar el equipo o al teclear "reset", se desplegará el logotipo del Instituto de Investigaciones durante un tiempo aproximado de 3 segundos, inmediatamente después y con el mismo tiempo de duración, el nombre del equipo como se ilustra en la figura 3.5.

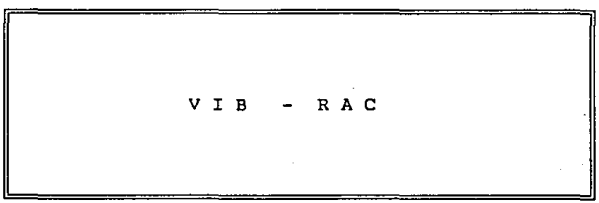

Figura 3.5

3.3.1 Modo de Operación "En Línea"

En éste modo de operación, el equipo es configurado para operar como un instrumento de monitoreo continuo, el cual, desplegará periódicamente gráficas de las variaciones de las señales de vibración.

El menú que la pantalla del equipo desplegará será como se muestra en la figura 3.6.

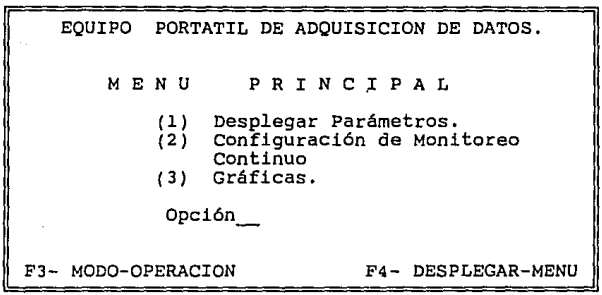

Figura 3.6

3.3.1.1 Menú Principal

(1) Desplegar Parámetros de operación

a) Modo de Operación: Esta opción proporciona al usuario información relativa al estado en que se encuentra el equipo portátil con respecto al equipo del cual ha de realizar la adquisición, es decir, si el equipo está operando como un equipo de adquisición de datos o como instrumento de monitoreo continuo.

b) Ruta Cargada: Indica si una ruta de adquisición de datos se ha cargado en la memoria del equipo portátil.

c) Velocidad de Transmisión: El equipo asume una velocidad

de transmisión de 1200 bit/s en la comunicación por el puerto serie; no obstante, el usuario puede alterar este parámetro tecleando el número correspondiente a la opción deseada; para 2400, 4800 o 9600 bits/s.

Los parámetros restantes para la configuración del puerto serie de comunicación están previamente definidos en el equipo según la tabla 3.1.

Paridad	Ninguna
Bits de paro	1
No. de bits	8

Tabla 3.1

La pantalla que presentará el equipo portátil se ilustra en la figura 3.7.

```
          DESPLIEGUE DE PARAMETROS
Modo de Operación:              Fuera de Línea

Ruta cargada en la memoria:          No

Vel. Transmisión [1200 bit/s]

1)2400    2)4800  3)9600 :
```

Figura 3.7

(2) Configuración de Monitoreo Continuo

El operador seleccionará del conjunto de opciones que presen-

ta el equipo de adquisición, la configuración apropiada para la adquisición periódica de la señal de vibración. El primer menú que presenta la pantalla del equipo para éste fin se ilustra en la figura 3.8.

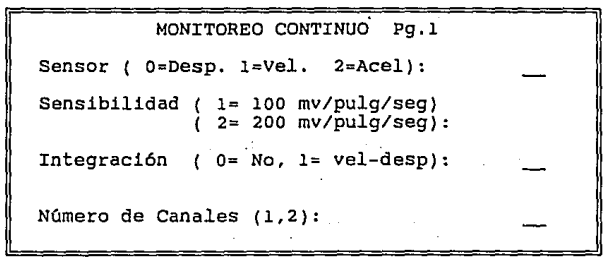

Figura 3.8

Tipo de sensor: Desplazamiento
Velocidad
Aceleración

Sensibilidad: 1) Desplazamiento; con una sensibilidad de 100 mv/mil o de 200 mv/mil.
2) Velocidad; con una sensibilidad de 100 mv/in/seg o de 500 mv/in/seg.
3) Aceleración; con una sensibilidad de 500mv/G.

Integración Opcional: En el caso de realizarse una adquisición con un tipo de sensor de aceleración o velocidad y se requiera información en función de velocidad o desplazamiento, respectivamente.

El usuario tecleará el número correspondiente a su elección, una vez se defina el número de canales simultáneos con los que se desa trabajar, el equipo desplegará la segunda página del menú como muestra la figura 3.9.

```
          MONITOREO CONTINUO  Pg.2
Frecuencia de Muestreo:

( 0= 500 Hz        1= 1   KHz        2= 2  KHz
  3= 5  KHz        4= 10  KHz        5= 20 KHz
  6= 30 KHz        7= 40  KHz        8= 50 KHz
                   9= 100 KHz):  __

No. de Muestras:
( 0= 512,  1= 1024,   2= 2048): __
```

Figura 3.9

Frecuencia de Muestreo: Se seleccionará la frecuencia de muestreo mas conveniente para la adquisición, según los requerimientos del punto.

Número de muestras por serie de tiempo; Esta opción permite al usuario establecer el número de muestreos a obtener para una señal de vibración.

(3) Gráficas

Se desplegará un submenú requiriendo la opción para el tipo de gráfica deseada como se muestra en la figura 3.10.

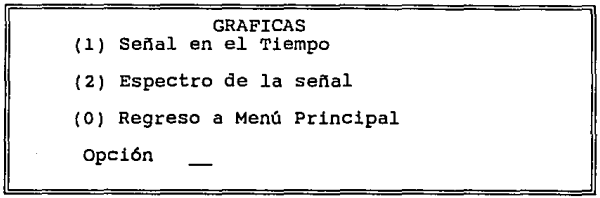

Figura 3.10

Se podrá obtener la gráfica del último punto adquirido solamente el caso en que éste sea de vibración.
El usuario contará con las teclas: <- y ->, las cuales le permitirán moverse a lo largo de la curva, así mismo aparecerán los valores de la coordenada del punto donde se posicione el cursor en la parte inferior de cada gráfica.
Las teclas correspondientes a las flechas que apuntan hacia arriba o hacia abajo tienen la función de aumentar o disminuír la velocidad de barrido del cursor a lo largo de la pantalla, cada que se presionen.

En la parte inferior se despliegan cuatro opciones que realizan las siguientes funciones:

1) Zoom: Acercamiento Se oprime la opción con el número correspondiente y aparece en la parte izquierda de la pantalla una línea vertical. Esta línea puede moverse lateralmente mediante las teclas <- y ->. Se selecciona el lugar donde se

desea que comience el acercamiento y se oprime la tecla <CR>. A continuación se vuelve a mover la línea hasta el lugar donde se desee que termine y se teclea nuevamente <CR>. Una vez realizados los pasos anteriores, el equipo despliega en la pantalla el acercamiento del intervalo definido.

2) Gráfica Original: Esta opción despliega la gráfica original si así se deseara después de haber realizado uno o varios acercamientos.

3) Cursor Gráfico: El usuario contará con las teclas: <- y ->, las cuales le permitirán moverse a lo largo de la curva, así mismo aparecerán los valores de la coordenada del punto donde se posicione el cursor en la parte superior de cada gráfica.
<CR>: Esta opción regresa el comando del programa al menú de graficas.

a) Gráfica de la Señal en el Tiempo:

El Formato de la gráfica de la señal es ilustrada en la figura 3.11.

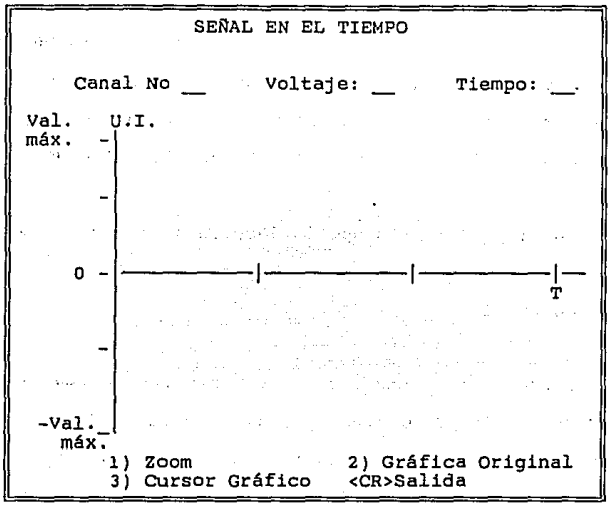

Figura 3.11

U.I.= Unidades de Ingenieria= Voltaje / Sensibilidad del sensor.

Si en el último punto de vibración se adquirieron dos canales simultáneamente se solicitará al usuario el número del canal deseado (figura 3.12).

Figura 3.12

b) Espectro de la señal

El formato del Espectro es mostrado en la figura 3.13.

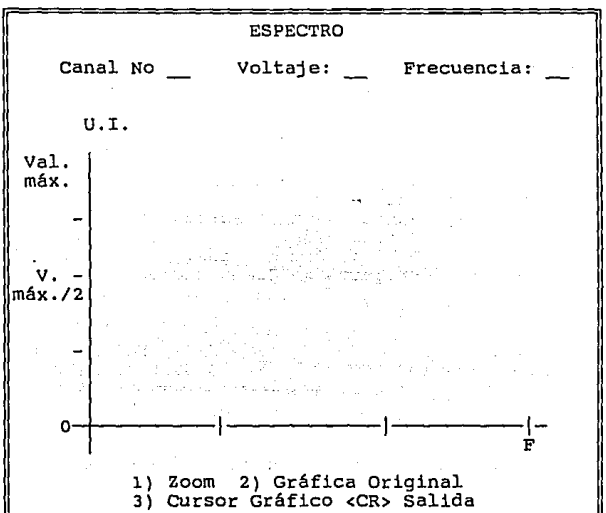

Figura 3.13

Así mismo en la gráfica del espectro, el equipo requerirá del número del canal, del cual se desea obtener la gráfica (figura 3.13). Si la adquisición se realizará en un solo canal, no aparecerá esta opción.

3.3.2 Modo de Operación "Fuera de Línea"

Una vez que desaparecen los mensajes de bienvenida, la panta-

lla despliega el menú principal del equipo (figura 3.14) cuando éste se encuentra funcionando en el modo fuera de línea.

3.3.2.1 Menú Principal

```
EQUIPO PORTATIL DE ADQUISICION DE DATOS.

      M E N U     P R I N C I P A L

          (1)  Desplegar Parámetros.
          (2)  Desplegar Memoria.
          (3)  Punto fuera de Ruta.
          (4)  Graficación.
          (5)  Definir Comentarios.
          (6)  Programa de Comunicación.

           Opción__

F1- ADQUIS                    F2- SIG-PUNTO
F3- MODO-OPERACION            F4- DESPLEGAR-MENU
```

Figura 3.14

El Equipo aceptará un número de un solo dígito, en el rango de 1 al 6 o las teclas para funciones especiales (F1, F2, F3, F4). Al oprimir cada uno de ellas se realizarán las funciones siguientes:

(1) Desplegar Parámetros de Operación

La pantalla que despliega y la información que proporciona el equipo portátil de adquisición de datos al teclear ésta opción, son análogas a las descritas en el modo de operación en línea (figura 3.7).

(2) Desplegar Memoria

Se desplegarán los siguientes datos correspondientes al estado actual de la memoria como se muestra en la figura 3.15.

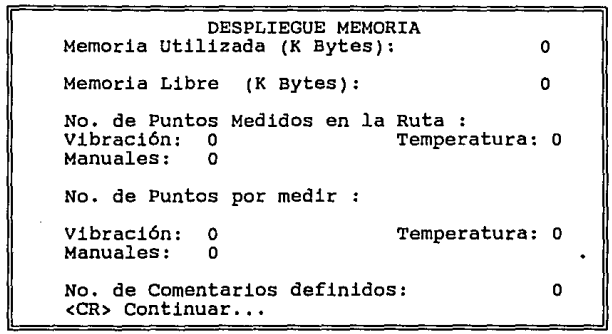

Figura 3.15

Los datos que proporciona este despliegue sirven para auxiliar al operador del equipo para dar un mejor uso de la memoria disponible.

Primeramente se despliega un número que indica la cantidad de memoria utilizada, en seguida otro que hace lo propio con la memoria restante o libre.

De la misma manera, se despliega el número de puntos adquiridos y el número de puntos por adquirir en la ruta indicando la naturaleza de la adquisición, es decir, si se trata de adquisición de valores de vibración de temperatura o manual, así como la cantidad de ellos que corresponden a cada tipo.

Por último se proporciona el número total de comentarios que se han definido en la ruta hasta el punto en que se encuentra el usuario.

(3) Definir Puntos Fuera de Ruta

Se permitirá definir y adquirir un punto no programado en la ruta cargada en el equipo portátil de adquisición de datos y que el usuario desee incluír.

El equipo desplegará el menú correspondiente al tipo de información que se desea adquirir (figura 3.16).

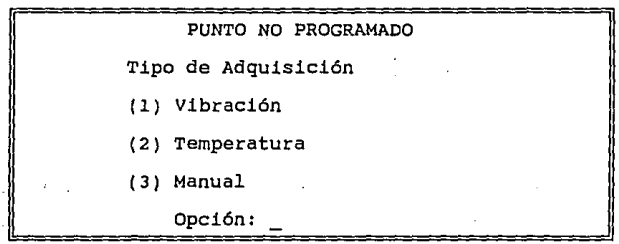

Figura 3.16

a) Vibración: Para esta opción, se desplegarán los siguientes menús:

Punto de Vibración

Este encabezado permanecerá en la parte superior del LCD e irán desplegándose los recuadros siguientes conforme sean seleccionadas las opciones.

Para todos los casos, el cursor se posicionará en el espacio subrayado en espera de la opción, la cual será aceptada de manera inmediata al presionar la tecla correspondiente, excepto en aquellos casos en los que el equipo requiera de un valor no especificado en el menú. Para tales valores el usuario deberá oprimir la tecla <CR> para accesar el dato.
La primera página del menú para un punto de vibración fuera de ruta se muestra en la figura 3.17.

```
          PUNTO DE VIBRACION  Pg. 1
Punto: XXXX

Sensor ( 0=Desp. 1=Vel.  2=Acel):          __

Sensibilidad ( 1= 100 mv/pulg/seg)
             ( 2= 200 mv/pulg/seg):

Integración  ( 0= No, 1= vel-desp):        __


Número de Canales (1,2):                   __
```

Figura 3.17

Punto: XXXX Es el indicador del punto que se está adquiriendo.
Si se realiza una adquisición de un punto no programado en la ruta cuando ésta no ha sido completada, se asignará un número comprendido entre dos múltiplos de diez. Esto es debido a que los puntos programados en la ruta, tienen ésta numeración previendo la inserción de puntos no programados.

El número de dígitos asignado para este fin es cuatro
Las características del tipo de sensor, sensibilidad, integración opcional, número de canales y frecuencia de muestreo, son las mismas que se emplean en la configuración del equipo de adquisición de datos como instrumento de monitoreo continuo, y que fueron descritas en el apartado correspondiente al funcionamiento en modo en línea.
La figura 3.18 muestra la manera como se despliega la segunda página de éste menú.

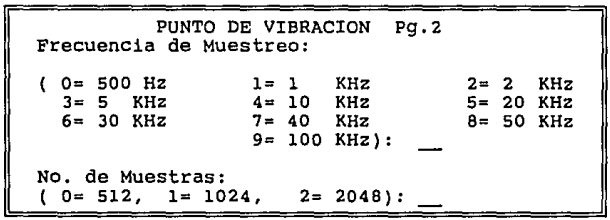

Figura 3.18

Función de adquisición (figura 3.19): Esta opción permite seleccionar los parámetros de vibración que se desean muestrear en la adquisición, donde Vpp es el valor pico a pico y Vrms es el Valor cuadrático medio. Si el usuario selecciona amplitud y fase, con estos datos se podrán realizar diagramas de Nyquist, y si se seleccionan señal y espectros, se desplegarán las gráficas correspondientes.

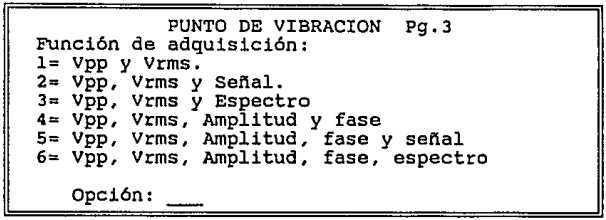

Figura 3.19

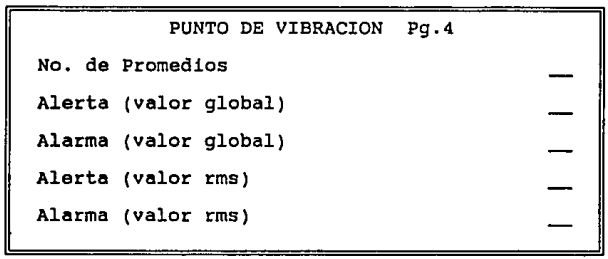

Figura 3.20

No. de promedios (figura 3.20): Opción para establecer el número de promedios que se deberán realizar para obtener los valores requeridos por la función de adquisición. El número de promedios puede definirse en el rango de 1 a 20, pero solamente en caso de que el usuario haya seleccionado las opciones 3 y 6 en la función de adquisición.

Valores globales y rms de Alerta y Alarma: Estos valores podrán ser accesados directamente, y serán del tipo punto

flotante.

(4) Gráficas

Se desplegará un submenú requiriendo la opción para el tipo de gráfica deseada de la misma manera que ocurre cuando el equipo opera en el modo en línea. Las opciones de graficación y los formatos presentados son análogos. Ver figuras 3.10, 3.11, 3.12 y 3.13.

(5) Definir Comentarios

Se seleccionará un comentario de un menú el cual será definido al momento de cargar la ruta en el equipo.

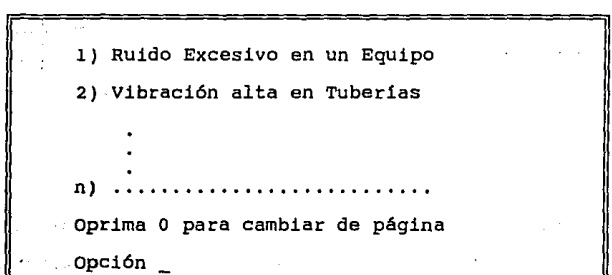

Figura 3.21

El usuario tecleará el número correspondiente al comentario deseado (figura 3.21)

El equipo aceptará el comentario seleccionado, en caso de

no encontrarse éste en la página desplegada, el usuario oprimirá la opción 0 (cambiar de página) para continuar su búsqueda.
Por tratarse de un menú circular, al desplegarse la última página y teclearse la opción 0, se pasará inmediatamente a la primera y continuará el orden progresivo.

(6) Programa de Comunicaciones: El usuario realizará la comunicación con la microcomputadora por medio del puerto serie RS232 definido y habilitado mediante la opción (1).
Durante esta comunicación de desplegarán mensajes indicando el procedimiento que se está llevando a cabo.

Ejem: "Error de comunicación"
" ....leyendo commando"
"Enviando datos de vibración, punto xxx"

3.3.3 Funciones Programadas

3.3.3.1 F-3 MODO-OP :

Al oprimir ésta tecla, el equipo cambia su modo de operación de manera que se desplegará el menú correspondiente al modo de operación deseado (figuras 3.6 y 3.14).

3.3.3.2 F-1 ADQUIS:

Realiza la adquisición. Botón de presión instantanea inicia la adquisición de la señal según las características definidas. Se desplegará alguno de los siguientes mensajes:

"Configurando el Hardware..."
"Calculando la serie de tiempo..."
"Calculando valores pico a pico y rms..."
"Calculando el espectro..."

Ejemplo: Si la función de aquisición requiere que se calculen los valores Vpp y Vrms y espectro, el equipo desplegará el mensaje durante el proceso de cálculo de Vpp y Vrms:

"Calculando valores p-p y rms..."

Y posteriormente se desplegarán los valores.
Estos valores permanecerán en el LCD hasta que el usuario oprima <CR>; tecla con la cual se iniciará el cálculo del espectro y se desplegará el mensaje :

"Calculando Espectro Lineal.."

Una vez realizado este proceso, se desplegará el espectro de la señal en el LCD. Nuevamente el equipo requerirá al usuario presionar <CR> para que, en ésta ocación despliegue, de exis-

tir, los estados de alerta o alarma.

3.3.3.3 F4- MENU:

Desplegar menú: Botón de presión instantanea.

Desplegar menú principal.

3.3.3.4 F2- SIG-PNT:

Siguiente punto. Botón de presión inmediata; desplegará las características del siguiente punto de la ruta.

Estas características son cargadas desde la computadora y en el caso en que el punto sea de vibración, la pantalla desplegará los valores correspondientes a la figura 3.22 .

El Equipo tiene reservado para la escritura del número del punto, nombre del equipo y descripción del mismo, para todos los tipos de adquisición, cuatro, diez y treinta caracteres respectivamente.

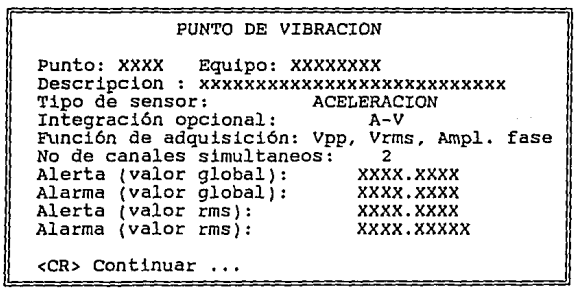

Figura 3.22

La figura 3.23 muestra la manera como se observan los datos, en el caso en que el punto sea de captura manual:

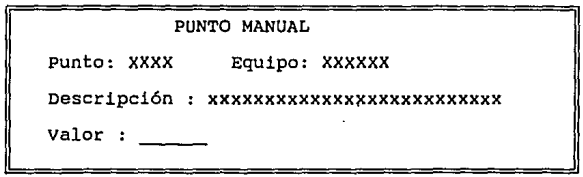

Figura 3.23

Así mismo para los puntos de temperatura (figura 3.24):

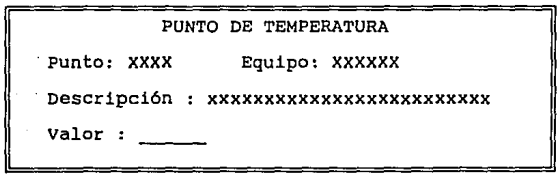

Figura 3.24

Cada Punto llevará un número progresivo múltipo de diez acompañado del nombre y una breve descripción del mismo, que facilitarán al usuario la identificación del equipo.

3.3.4 Programa de comunicaciones

El Protocolo de comunicación se basa en un conjunto de comandos. Para cada comando existe una respuesta que puede consistir de un "buffer" de datos, cuya lectura y escritura

se lleva a cabo mediante las funciones leer_buffer() y escribe_buffer(), o simplemente una salida y/o entrada de octetos por los puertos de comunicación (leer_puerto() y escribe_puerto).
Cada comando es identificado por medio de un número (1 octeto) codificado en hexadecimal.

Los comandos de comunicación son:

1. Solicitud de Reconocimiento.
   El ADQPORT responde "Ack". Comando utilizado para iniciar la comunicación con la computadora central.
   Los reconocimientos "Ack" y "NoAck" son indicativos del reconocimiento de los comandos de comunicación.

   Ack = FD
   NoAck = FE

2. Solicitud de fecha y hora.
   El ADQPORT enviará la fecha y la hora que actualmente tiene en memoria en el formato:

   dd mm aa hh mm ss

   dd = día
   mm = mes
   aa = año
   hh = hora
   mm = minutos
   ss = segundos

Todo especificado en formato BCD (1 octeto por parámetro).

3. Envío de fecha y hora.
   El ADQPORT recibirá la hora y fecha actual desde la computadora central en el buffer de comunicación (buffer [0 al 5]). El formato es el mostrado en el comando anterior.

4. Comando de definición de modo de operación.
   Este comando define el modo en que deberá operar el equipo. El formato se muestra en la tabla 3.2:

<CODIGO>

Código	Modo
00	Fuera de Línea
01	En línea

Tabla 3.2

5. Envío de configuración de punto de vibración. La computadora central enviará la configuración del punto a capturar. Esta información es la siguiente:

<IdP><Equipo><Descr><TipS><IntOpc><SenalF>
<No_canales><Fmuestreo><NoMuestras>
<Promedios><AlertaPP><AlarmaPP><AlertaR>
<AlarmaR>

<IdP> = Identificador del punto (2 octetos BCD)

buffer [0 y 1].

<Equipo> = Nombre del equipo.

buffer [2 al 11]

<Descr> = Descripción del punto (30 caracteres)

buffer [12 al 41]

<TipS><IntOpc><SenalF> = 1 palabra (2 octetos) que especifica el tipo de sensor, si se va a integrar la señal y la función de adquisición, de acuerdo con el siguiente formato:

Buffer [42]

BIT

```
b2  b1  b0  = Tipo de sensor:
 0   0   0  : Desplazamiento
 0   0   1  : Velocidad
 0   1   0  : Aceleración
```

BIT

```
b5 b4 b3   = Número de canales simultáneos
 0  0  1   : 1
 0  1  0   : 2
```

BIT

```
b7 b6      = Sensibilidad.
 0  0
 0  1
```

La sensibilidad será decodificada en base al tipo de sensor.

buffer [43]

BIT

```
b1   b0 = Integración Opcional
0    0  : Sin integración.
0    1  : Integración velocidad a Desplazamiento
          (100 mV/in/seg a 100 mV/mil)
1    0  : Velocidad a Desplazamiento
          (500 mV/in/seg a 100 mV/mil)
1    1  : Aceleración a velocidad
          (500 mV/G a 100 mV/pulgada/segundo)
```

BIT

```
b5  b4 b3 b2 = Función de adquisición
 0   0  0  0  : Valor pico a pico y valor RMS
 0   0  0  1  : Vpp, Vrms y Señal en el tiempo
 0   0  1  0  : Vpp, Vrms y Espectro
```

buffer [44]
BIT

```
b1    b0  = Número de muestras por serie de
            tiempo
0  0 :  512
0  1 :  1024
1  0 :  2048

b5 b4  b3  b2= Frecuencia de muestreo:
 0  0  0  0 :     500 Hz
 0  0  0  1 :    1000 Hz
 0  0  1  0 :    2000 Hz
 0  0  1  1 :    5000 Hz
 0  1  0  0 :   10000 Hz
 0  1  0  1 :   20000 Hz
 0  1  1  0 :   30000 Hz
 0  1  1  1 :   40000 Hz
 1  0  0  0 :   50000 Hz
 1  0  0  1 :  100000 Hz
```

<Promedios> Octeto que indica (en hexadecimal) el número de promediados a efectuar, de acuerdo con la siguiente especificación:

buffer [45]

<AlertaPP> Nivel de alerta pico a pico

buffer [46 al 53]

<AlarmaPP> Nivel de alarma pico a pico

buffer [54 al 61]

<AlertaR> Nivel de alerta RMS

buffer [62 al 69]

<AlarmaR> Nivel de alarma RMS

buffer [70 al 77]

6. Envío de configuración de temperatura a adquirir con sensor integrado, con el siguiente formato:

<IdP><Equipo><Descripcion><Alerta><Alarma>

<IdP> = Identificador del punto (2 octetos, BCD)

buffer [0 y 1]

<Equipo> = Nombre del equipo

buffer [2 al 11]

<Descripcion> = 30 caracteres

buffer [12 al 41]

<Alerta> = Nivel de alerta (grados C.)

buffer [42 al 49]

<Alarma> = Nivel de alarma (grados C.)

buffer [50 al 57]

Estos dos niveles especificados en formato flotante de 4 octetos.

7. Envío de configuración de variable a capturar manualmente, con el siguiente formato:

<IdP><Equipo><Descripcion><AleB><AlaB><AleA><AlaA>

<IdP> = Identificador del punto (2 octetos BCD)

buffer [0 y 1]

<Equipo> = Nombre del equipo

buffer [2 al 11]

<Descripcion> = Descripción 30 caracteres

buffer [12 al 41]

<AleB> = Nivel de alerta baja

buffer [42 al 49]

<AlaB> = Nivel de alarma baja

buffer [50 al 57]

<AleA> = Nivel de Alerta alta

buffer [58 al 65]

<AlaA> = Nivel de alarma alta

buffer [66 al 72]

8. Solicitud de puesta en modo de envío de "buffers" de vibración.
Con éste comando, el equipo portátil de adquisición de datos busca el principio de la lista de puntos de vibración capturados y se prepara para enviarlos cuado la microcomputadora los requiera.

9. Solicitud de puesta en modo de envío de variables de temperatura y variables capturadas manualmente.
El ADQPORT localiza el principio de la lista de

puntos de temperatura adquiridos.

10. Solicitud de puesta en modo de envío de "Buffer" de comentarios.
El equipo localiza el comienzo de la lista de puntos de captura manual adquiridos y se prepara para la transmisión a la PC.

11. Solicitud de siguiente "buffer" de vibraciones. El ADQPORT enviará el siguiente "buffer" de vibraciones. Este "buffer" tiene el siguiente formato:

Buffer[i]	Código	Función
0	BCD	Dig. mas significativo del Ident.
1	BCD	Dig. menos significativo del ident.
2	BCD	Fecha : mes
3	BCD	Fecha : día
4	BCD	Tiempo : Hora
5	BCD	Tiempo : Minutos
6	HEX	Sin Uso.
7	HEX	Sin Uso.
8	HEX	Sin Uso.
9	HEX	Sin uso.
10	HEX	Valor pico a pico canal 1 (MSB)
18	HEX	Valor pico a pico canal 1 (LSB)
26	HEX	Valor pico a pico canal 2 (MSB)
34	HEX	Valor pico a pico canal 2 (LSB)

buffer[42 a (42+ (número de muestras de la Señal en el tiempo) * (número de canales simultáneos)]
Función de adquisición que incluye señal en el tiempo.
buffer [(42 + número de muestras de la Señal en el tiempo) a (42 + 2*(número de muestras de la señal en

el tiempo) * (número de canales simultáneos)]
Función de adquisición que requiere además de la señal en el tiempo, las muestras del espectro .

12.- Solicitud de siguiente "buffer" de variable de temperatura o adquirida manualmente. Tienen el siguiente formato:

<IdP><Hora><Valor>
<IdP> Identificador del punto (2 octetos, BCD)
<Valor> Valor de la variable en formato flotante.

Buffer[i]	código	Función
0	BCD	Dig. mas significativo del Ident.
1	BCD	Dig. menos significativo del ident.
2	BCD	Fecha : mes
3	BCD	Fecha : día
4	BCD	Tiempo : Hora
5	BCD	Tiempo : Minutos
6 al 13	Variable en formato flotante.	

13.- Solicitud de siguiente comentario. El comentario tiene el siguiente formato:

<No><Equipo><Hora><Comentario>
<No> Número de comentario (1 octeto, BCD)
<Equipo> Nombre del equipo
<Comentario> Identificador del comentario (octeto)

14. Solicitud de re-envío de "buffer" anterior.
Rutina que repite la transmisión del buffer inmediatamento anterior.

15. Configuración de canales en modo en línea:

<Canal><TipoSens><TipoSinc>
<Canal> : Número de canal (1 o 2)
<TipoSens> : Tipo de sensor.

16. Solicitud de lista de comentarios.
Este comando solicita a la microcomputadora la lista de comentarios posibles que se pueden definir a lo largo de la ruta que se está cargando.

17. Fin de transmisión.
Este comando realiza las siguientes funciones:
a) Verificación de una ruta cargada en el equipo.
b) Si la ruta fué cargada, se prepara al equipo para el recorrido localizando el primer punto y determinando el tipo del que se trata.

3.4 El Lenguaje de Programación

En la selección del lenguaje de programación se considerarán tres opciones:

- Lenguaje Macroensamblador 80286.
- Lenguaje Turbo Pascal 4.0.
- Lenguaje C.

La primera opción parecería la mas apropiada para el desarrollo óptimo de programas fuertemente ligados a la operación del "Hardware"; sin embargo presenta la desventaja de su complicada codificación y mantenimiento.

El Turbo Pascal es un lenguaje de Alto nivel con instrucciones para el manejo directo del "Hardware", sin embargo, como el programa de control debía residir en memoria ROM, se requería realizar una conversión del programa ejecutable para éste fin.

Por otra parte el lenguaje C es un lenguaje de nivel medio con el mismo tipo de instrucciones del "hardware" pero que además cuenta con las herramientas necesarias para crear el programa para la ROM.

Debido a las razones antes expuestas, se optó por utilizar el Lenguaje C de programación que, además de proporcionar ventajas considerables en la codificación, se apegaba al criterio de diseño.

En un principio se utilizó el compilador de Turbo-C 2.0; sin embargo, debido a algunos problemas en el uso del "stack", se decidió utilizar el compilador Quick-C.

A continuación se presenta una breve descripción de las principales instrucciones, comandos y operadores del Lenguaje C.

3.4.1 El Lenguaje C de Programación

El Lenguaje C un lenguaje estructurado.
La característica básica de un lenguaje estructurado, es la utilización de bloques o subrutinas. Un bloque es un conjunto de sentencias que están relacionadas lógicamente.
Un lenguaje estructurado permite compilar por separado las subrutinas sin necesidad de que formen un solo programa propiamente dicho. Este hecho permite crear una librería de subrutinas con funciones útiles y ya probadas a las que se puede accesar desde cualquier parte del programa principal.
Todos los compiladores cuentan con una librería estandar que proporcionan las funciones necesarias para la realización de las tareas mas comunes (en la elaboración de éste trabajo se utilizaron tanto las librerías estandar del compilador de C como algunas librerías especiales del paquete QuickC de la firma Microsoft.
El programa principal es a su vez una función la cual encierra todas las "llamadas" a las subrutinas.
Esta función principal lleva el nombre de "main()" y es la primera función que busca el compilador.
Todas las instrucciones o llamadas que se encuentran en una función están encerradas entre llaves ({}) para definir el inicio y el final del proceso que realizan. Las llaves de la función main () definen el inicio y el final del programa.

3.4.1.1 Variables y Operadores

1) Variables : El nombre de las variables en el lenguaje C puede consistir de uno o mas caracteres, debiendo ser el primero de éstos forzosamente una letra.
El tipo de variable define el tamaño y rango de éstas como se muestra en la tabla 3.3.

Tipo	Número de Bits	Rango
char	8	0 a 255
int	16	-32768 a 32767
short int	8	-128 a 127
unsigned int	16	0 a 65535
long int	32	-4294967296 a 4294967295
float	32	aprox. 6 dígitos de precisión.
double	64	aprox. 12 dígitos de precisión.

Tabla 3.3

Todas las variables deben haber sido declaradas antes de usarlas. Estas declaraciones pueden realizarse básicamente en tres sitios del programa: dentro de las funciones (variables locales), en la definición de los parámetros de la función (parámetros formales), o fuera de todas las funciones (variables globales).
Las variables locales solamente pueden ser referenciadas o utilizadas dentro de la función en la cual fueron declaradas. Estas variables son dinámicas, es decir, sus contenidos

se pierden cada vez que se retorna de la función y son creadas y destruídas en cada llamada de la función.
Después del nombre de la función y antes de abrir la llave, se declaran los parámetros formales, los cuales alojan los valores que se transfieren desde la llamada a la función.
Las variables globales son estáticas, sus datos se mantienen durante todo el programa y pueden ser referenciadas por cualquier expresión independientemente de la función en que se hallen.
El Lenguaje C cuenta con el modificador de variable "register", el cual es solo aplicable a los tipos int y char. Este modificador obliga al compilador a mantener el valor de las variables declaradas en un registro de la CPU en lugar de la memoria, lo cual repercute en una mayor velocidad en la ejecución de las operaciones.

2) Operadores: Los operadores son símbolos que indican al compilador que se están realizando manipulaciones matemáticas y lógicas.

a) Los operadores aritméticos: Estos operadores son aquellos que realizan las operaciones mas comunes (*,-,+ y /). Cuando el operador / se aplica a un entero o a un caracter, el residuo de la operación será truncado.
El operador % de división obtiene el resto de una división entera.

Los operadores ++ y -- incrementan o decrementan en una unidad a la variable. (Tabla 3.4)

b) Los operadores Relacionales y Lógicos

Estos operadores se muestran en la Tabla 3.5.

Tanto los operadores racionales como los operadores lógicos tienen un nivel de precedencia menor que los operadores aritméticos, no obstante, al igual que éstos últimos, devolverán el valor de 1 si la aseveración es cierta y 0 si es falsa.

Código	Significado
-	Resta, también menos unario.
+	Suma.
*	Multiplicación.
/	División.
%	División Módulo.
--	Decremento.
++	Incremento.

Tabla 3.4

c) Operadores de Bits

Una de las principales ventajas que presenta el lenguaje C es el complejo juego de operadores de bits con el que cuenta. Las operaciones con bits se refieren a la comprobación, colocación o desplazamiento de los bits actuales de una variable entera o de caracter. Estas operaciones no pueden aplicarse a los tipos float y double.

Los operadores sobre bits se muestran en la Tabla 3.6

Operador Relacional	Acción
>	Mayor que
<	Menor que
>=	Mayor o Igual
<=	Menor o Igual
==	Igual
!=	Diferente
Operador Lógico	**Acción**
&&	AND
||	OR
!	NOT

Tabla 3.5

Operador	Acción
&	AND
|	OR
^	OR exclusivo.
~	Complemento a uno.
>>	Desplazamiento a la Derecha.
<<	Desplazamiento a la Izquierda.

Tabla 3.6

d) Operadores de Punteros o Apuntadores: Un apuntador en lenguaje C se refiere a la dirección en la memoria de una variable. El operador & devuelve dicha dirección, mientras que el operador * devuelve el valor de la variable ubicada en la dirección que se le asigna.

Tanto & como * tienen una precedencia mayor que cualquier operador aritmético, excepto el del menos unario el cual la tiene igual.

3.4.1.2 Sentencias de Control

1) Sentencias Condicionales

a) La sentencia "if": El formato general de la sentencia "if" es:

```
if (prueba de condición) sentencia1;
else sentencia2
```

Si la condición es cierta (cualquier valor distinto de 0), se ejecutará la sentencia1, de lo contrario, será la sentencia2 la que se realice.

Si cualquiera de las sentencias exceden de una instrucción, se deberán utilizar llaves para marcar el bloque.

b) La sentencia "switch": Esta sentencia de decisión cuenta con bifurcación múltiple, es decir, realiza la comparación de varias sentencias y, cuando se obtiene una igualdad, se ejecuta dicho proceso. El formato general de la sentencia "switch" es:

```
switch (variable){  case constante1:
                            sentencia1;
                            break;
                    case constante2:    sentencia2;
                            break;

                    default :  sentencia:
                }
```

La instrucción BREAK finaliza la operación de las sentencias comprendidas en el "case".

2) Bucles

a) El bucle "for": La utilidad del bucle "for" reside en la capacidad para ejecutar una sentencia mas de una vez. El formato general de "for" para repetir una sentencia única es:

```
for (inicialización; condición; incremento)
      sentencia;
```

Para repetir un bloque, el formato general es:

```
for (inicialización; condición; incremento;)
  { sentencia 1;
    sentencia 2;
    .
    .
    sentencia n;
  }
```

b) El bucle "while": En éste bucle, la sentencia se ejecuta mientras que la condición sea cierta, cuando la condición no lo es, el control del programa pasa automaticamente a la línea siguiente al código del bucle.

El formato general del bucle "while" es:

```
while (condición) sentencia;
```

Si se trata de mas de una sentencia, el bloque se encerrará entre llaves.

Un caso particular del bucle "while" es el ciclo "do-while", en el cual la condición se evalúa al finalizar la operación

de las sentencias comprendidas en el bucle. El formato general del ciclo "do-while" es el siguiente:

```
do{
   senntencias;
  } while (condición);
```

3.4.1.3 Estructuras

Se le llama estructura a un conjunto de variables que están referidas bajo un mismo nombre.

El Formato general de una estructura es el siguiente:

```
struct nombre_estructura{
       tipo     nombre_variable;
       tipo     nombre_variable;
       .            .
       .            .
       };
```

Las variables encerradas dentro de las llaves son el conjunto de variables que contiene el registro (otro nombre con el que se conoce a las estructuras), la variable tipo registro, se declara mediante la sentencia "struct" y el nombre del registro:

```
struct nombre_estructura variable_estructura;
```

Para accesar una variable comprendida dentro del registro se utiliza el operador punto:

```
variable_estructura.nombre_variable = constante;
```

Otra manera de desplazarse dentro del registro es mediante la utilización de un apuntador a la estructura, para ello, hace falta definir una variable tipo registro del tipo de la

estructura. Esta definición se realiza mediante la sentencia "typedef":

typedef struct nombre_estructura (variable tipo registro);

"typedef" define a nombre_estructura como un tipo de variable. Variable tipo registro queda declarada como variable del tipo nombre_estructura.

A partir de éste momento variable tipo registro puede ser utilizado como un tipo de variable, como lo sería int o float.

```
variable tipo registro *apuntador;
```

La sentencia anterior declara un apuntador tipo variable registro.

Una vez declarado el apuntador el acceso se realiza mediante el operador flecha:

```
apuntador->nombre_variable = constante;
```

3.4.1.4 Instrucciones de I/O

Las instrucciones de I/O básicas son las instrucciones de lectura y escritura en puertos y registros con los cuales un programa tiene comunicación con el mundo exterior.

Estas instrucciones son inp y outp.

Los formatos de éstas instrucciones son los siguientes:

```
variable = inp(numero_puerto);
outp (número_puerto, valor_salida);
```

El número del puerto puede ir desde el 0 hasta el 65537. Este valor puede ser accesado en sistema decimal o bien en hexadecimal, en cuyo caso deberá hacerse de la manera siguiente:

```
0xnumero h
```

la "h" indica al compilador el uso del sistema hexadecimal. El valor de salida puede teclearse directamente o mediante el uso de una variable que lo contenga.

La variable donde se almacenan los datos de entrada/salida deben ser del tipo "unsigned char" o caracter sin signo. Cabe aclarar que si ciertamente para una variable tipo "char" se reserva un byte o lo que es lo mismo ocho bits, solo siete de ellos almacenan el valor mientras, que el bit restante el signo.

Por ésta razón las variables que almacenan datos de entrada/salida requieren ser declaradas como "unsigned char" ya que de esta forma todos los bits del caracter se utilizan para éste fin.

Capítulo IV.- Desarrollo y Codificación

En este capítulo se presenta la codificación del programa de control del equipo portátil de adquisición de datos.

Las funciones o rutinas se agruparon en un conjunto de módulos, los cuales se describen a continuación:

. PORT.C

Contiene la declaración de las variables globales del sistema, la función main() y los procedimientos principales.

. GRAFICAS.C

Contiene las rutinas para la realización de las gráficas, desde las rutinas básicas para la generación de pixels.

. FFT.C

Contiene las funciones para el cálculo de la transformada rápida de Fourier, que es utilizada para calcular el espectro de las señales de vibración.

. LOGO.C

Contiene las funciones para el despliegue del logotipo del Instituto de Investigaciones Eléctricas.

. LCD.C

Contiene las funciones de inicialización y programación del LCD.

. P_COMLCD.C

Contiene las rutinas de protocolo de comunicaciones.

. RS232C.C

Contiene las funciones de comunicación básicas entre el equipo portátil y una microcomputadora, vía el puerto serie.

. COP.C

Contiene las funciones de programación del generador de pulsos programable.

. KEYBOARD.C

Contiene las funciones para la lectura del teclado del equipo portátil.

Esta rutinas fueron clasificadas en base a la complejidad de la función que realizan. Primeramente se presentan aquellas de caracter básico del sistema y al final las de mayor nivel de complejidad.

4.1 Funciones Básicas

4.1.1 Funciones de control del display de cristal líquido

Todas las funciones de control del LCD son un conjunto de llamadas a la rutina básica de comunicación.

4.1.1.1 Función básica de comunicación al LCD

La función básica de comunicación al Display de Cristal Líquido requiere de dos valores: instrucción y función. El primero es el indicador de la función que se desea realizar mientras que el segundo es el arreglo de octetos necesario para cumplir los requerimientos de la función.

```
lcd (byte instruccion, byte funcion)

{
   /* 1-4, 2-5, 3, 6 */
  outp ( 0x35b, 0x08);      /*
  outp ( 0x35b, 0x09);

  outp ( 0x35f, instruccion );
  delay_us ( 0 );
  outp ( 0x35b, 0x08);
  delay_us ( 0 );
  outp ( 0x35b, 0x00);

  outp ( 0x35b, 0x01);
  outp ( 0x35f, funcion );
  delay_us ( 0 );
  outp ( 0x35b, 0x00);
  delay_us ( 0 );
}
```

4.1.1.2 Configuración del LCD

```
void set_lcd_mode (byte blink,byte cursor, byte
             graphmode)

{  byte by;
   by = 48 ;                        /* siempre master, encendido
                                       y gen. de car. interna */
   if ( blink == 1 ) by = by | 8 ;    /* Blink / no Blink*/
   if (  cursor == 1 ) by = by | 4 ;   /* Cursor / no
                                      Cursor*/
   if ( graphmode == 1 ) by = by | 2 ;   /* modo  grafico /
                                            alfanumerico */

   lcd ( 0x00, by );
   currblink = blink ;
   cursorhab = cursor ;
   modelcd = graphmode ;     }
```

4.1.1.3 Inicialización del modo caracter

```
void init_lcd (int no_reng, int n_c_r )

/* no_reng = Número de renglones, n_c_r = Número de caracte-
res por renglon.*/

  {
    outp (0x35b,0x04);
    outp (0x35b,0x00);
    outp (0x359,0x92);

    set_address (0x00,0x00);
    set_cursor_address ( 0x00 , 0x00 ) ;

/*Pone el LCD en modo alfanumerico, con cursor y con blinking
*/

    read_busy_flag () ;
    set_lcd_mode ( 1, 1, 0 ) ;
    set_char_size (  no_reng,  n_c_r );


    lcd ( 0x03, 63) ;            /* Numero de renglones = 2 *
                                    int(64 /vp) */
    lcd ( 0x04, 0x06 );          /* Posicion del cursor desde
                                    arriba del caracater
                                    ( 0 a 15) y =< vp*/
    no_renglones = no_reng ;
    no_car_ren = n_c_r ;
  }
```

4.1.1.4 Inicialización del modo gráfico

```
void init_graph (void)

  /* Define el modo grafico*/

  {
   byte by;

   by =  0x32 ;           /* siempre master, encendido y
                             grafico*/
   lcd ( 0x00, by );
   lcd (0x01 ,0xf7) ;    /* tamano del caracter
                            (vp= no tiene sentido, hp-1)
                            hp= Numero de bits que se prenden
                            de un byte*/
   lcd ( 0x02, 0x1d) ;       /* (Hn-1 )
                            Hn= Numero  de bytes horizontales
                            por renglon*/
   }
```

4.1.1.5 Finalizar modo gráfico

```
void close_graph (void)

 /* Termina el modo grafico */

 { init_lcd ( no_renglones, no_car_ren );  }
```

4.1.1.6 Limpieza parcial de la pantalla (modo caracter).

```
void clean_lcd (int no_car_lim)

/* no_car_lim = Número de caracteres a limpiar*/
{
  int i;
  int clean;
  clean = no_car_lim ;

/*Llenar de caracteres blancos a partir de la dirección donde
esta el cursor*/
  for ( i=0; i<=(clean); i++)
  lcd (  0x0c, 0x00);     /* Blancos que se escriben desde el
                            espacio anterior a la posición
                          del cursor */
}
```

4.1.1.7 Limpieza parcial de la pantalla (modo gráfico)

```
void clean_graph (void)

{
  int i;

/*Llenar de caracteres blancos a partir de la dirección donde
está el cursor*/
  for ( i=1; i<=3840 ; i++)
  lcd ( 0x0c, 0x00);        /*10) Caracteres o bytes  que se
                            escriben en espacio anterior
                              a la posición del cursor */
}
```

4.1.1.8 Limpieza total de la pantalla

```
void clscr_lcd (void)

 /* Limpia la pantalla en el modo alfanumerico */

  {
    register int i ;

    if ( modelcd == 0 ) {
      set_cursor_address ( 0, 0 ) ;
      for ( i = 0 ; i < 4096 ; i++ ) writechar ( ' ' ) ;
      }
  }
```

4.1.1.9 Cursor

```
void draw_cursor (byte x, byte ymin,byte ymax)


/* Dibuja la línea del cursor */

{
  register int iy ;
  byte status ;
  for ( iy = ymin ; iy <= ymax ; iy++ ) {
    /* Determinar el pixel */
    read_pixel ( x, iy, &status ) ;
    if ( status == 0 ) putpixel ( x, iy, 1 ) ;
    else putpixel ( x, iy, 0 ) ;
    }
}  /* draw_cursor */
```

4.1.1.10 Ubicación del cursor (modo caracter)

```
void set_cursor_address (byte bylo, byte byhi)

/* bytelo= parte baja de la dirección, bytehi= parte alta de
la dirección*/

  {
/* Poner el cursor después del último caracter */
  lcd ( 0x0a,     bylo  );
/* 8) Parte baja de la dirección donde aparecerá el cursor*/
  lcd ( 0x0b,     byhi  );
/* 9) Parte alta de la dirección donde aparecerá el cursor*/
  }
```

4.1.1.11 Ubicación del cursor (modo gráfico)

```
void gotoxy_lcd (byte col, byte row)

/* Posiciona el cursor en la columna COL y linea ROW */

{
  int address ;

  address = no_car_ren * row + col ;
  set_cursor_address ( address % 256, address / 256 ) ;
  currline = row ;
  currcol = col ;

}
```

4.1.1.12 Definición del tamaño de los caracteres

```
void set_char_size (int no_renglones, int
        no_car_ren)

 /*  no_renglones   =  Número  de  renglones  por  pantalla,
no_car_ren = Número de caracteres por renglón */

  {
   int hp,vp ;
    if ( no_car_ren == 30 ) hp = 8 ;
    else hp = 6 ;

  switch ( no_renglones){

    case 9: vp = 14;
      break;
    case 10: vp = 12;
      break;
    case 11: vp = 11;
      break;
    case 12: vp = 10;
      break;
    case 13: vp = 9;
      break;
    case 16: vp = 7;
      break;
    default: vp = 15;
     }

 lcd ( 0x01, ((vp * 16 ) + hp-1)) ;
                    /* tamano del caracter ( vp-1, hp-1),
                       vp-1 [ 6(18 líneas) hasta f(8
                       líneas) ]  hp-1= 5 o 7 */
```

```
  if ( hp == 6 )  lcd ( 0x02, 0x27 ) ;
  else
  lcd ( 0x02, 0x1d ) ;     /* número de caracteres
                                por renglón
                           27(40 caracteres, hp-1= 5 )
                           1d(30 caracteres, hp-1= 7 )*/

  }
```

4.1.1.13 Escritura de caracteres

```
void writechar (char ch)

/* Escribe un caracter (ch) en la posición actual */

  {
    read_busy_flag () ;
    lcd ( 0x0c, ch );
    ++currcol ;
  }
```

4.1.1.14 Escritura de cadena de caracteres

```
void writestring (char *str)


/* Escribe una cadena de caracteres en la posición actual */

  {
    register int i ;
    for ( i = 0 ; i < strlen ( str ) ; i++ )
      writechar ( str[i] ) ;
  }
```

4.1.1.15 Escritura de un caracter en modo gráfico

```
void write_graph_char (byte x, byte y, byte ch)

/* Escribe un caracter  grafico. X  y Y  es la  dirección en
posiciones  (línea  y  columna)  donde  se  va a escribir el
caracter */

{
  byte count ;
  byte index ;
 /* Indice en el arreglo de caracteres gráficos */
  int address ;
```

```
  index = ch - 32 ;
  /* Define la dirección donde va a escribir */
  address = 240 * (  x - 1 ) + ( y - 1 ) ;
  for ( count = 0 ; count < 7 ; count++ ) {
    set_cursor_address ( address % 256, address / 256  ) ;
    lcd ( 0x0c, graphics_chars[index][count] ) ;
    address += 30 ;
    }
}
```

4.1.1.16 Escritura de una cadena de caracteres en modo gráfico

```
void write_graph_string (byte x,byte y, byte *str)

/* Escribe  la cadena de caracteres gráficos STR a partir de
la posición X, Y  (línea y columna) */

{
  byte cols ;
  byte i ;

  cols = y ;
  for ( i = 0 ; i < strlen ( str ) ; i++ ) {
    write_graph_char ( x, cols, str[i] ) ;
    cols++ ;
    }
}
```

4.1.1.17 Escritura de un caracter grande en modo gráfico

```
void write_graph_char_big (byte x, byte y, byte ch)

/* Escribe un caracter grande (ch), en la posición X, Y */

{
  byte count ;
  byte index ;  /* Indice en el arreglo de caracteres
                   gráficos */
  int address ;
  unsigned uns ;

  index = ch - 32 ;
  address = 480 * ( x - 1 ) + 2 * ( y - 1 ) ;
  for ( count = 0 ; count < 7 ; count++ ) {
    set_cursor_address ( address % 256, address / 256 ) ;
    uns = chr_to_big ( count, index ) ;
```

```
    lcd ( 0x0c, (byte) ( uns % 256 ) ) ;
    lcd ( 0x0c, (byte) ( uns / 256 ) ) ;
    address += 30 ;
    set_cursor_address ( address % 256, address / 256 ) ;
    lcd ( 0x0c, (byte) ( uns % 256 ) ) ;
    lcd ( 0x0c, (byte) ( uns / 256 ) ) ;
    address += 30 ;
    }
}
```

4.1.1.18 Escritura de una cadena de caracteres grandes en modo gráfico

```
void write_graph_string_big (byte x, byte y, byte
                             *str)

/* Escribe una cadena de caracteres grandes en el modo
gráfico */

{
  byte cols ;
  byte i ;

  cols = y ;
  for ( i = 0 ; i < strlen ( str ) ; i++ ) {
    write_graph_char_big ( x, cols, str[i] ) ;
    cols++ ;
    }
}
```

4.1.1.19 Transformación a un caracter grande

```
unsigned chr_to_big (byte count, byte index)

{
  byte niblehi ;
  byte niblelo ;
  byte by ;
  unsigned sum ;
  by = graphics_chars [index][count] ;
  niblehi = ( ( by & 0xF0 ) >> 4 ) ;
  niblelo = ( by & 0x0F ) ;
  switch ( niblehi / 4 ) {
    case 0 : sum = 0x0000 ;
       break ;
    case 1 : sum = 0x3000 ;
       break ;
    case 2 : sum = 0xc000 ;
```

```
        break ;
      case 3 : sum = 0xf000 ;
        break ;
      }

    switch ( niblehi % 4 ) {
      case 0 : sum += 0x0000 ;
        break ;
      case 1 : sum += 0x0300 ;
        break ;
      case 2 : sum += 0x0c00 ;
        break ;
      case 3 : sum += 0x0f00 ;
        break ;
      }

    switch ( niblelo / 4 ) {
      case 0 : sum += 0x00 ;
        break ;
      case 1 : sum += 0x30 ;
        break ;
      case 2 : sum += 0xc0 ;
        break ;
      case 3 : sum += 0xf0 ;
        break ;
      }

    switch ( niblelo % 4 ) {
      case 0 : sum += 0x00 ;
        break ;
      case 1 : sum += 0x03 ;
        break ;
      case 2 : sum += 0x0c ;
        break ;
      case 3 : sum += 0x0f ;
        break ;
      }
    return ( sum ) ;
}
```

4.1.1.20 Definición de la pantalla de visión

```
setviewport_lcd (int xmin, int xmax, int ymin, int
                 ymax)

/* xmin = ordenada mínima, xmax = ordenada máxima
 ymin = abcisa mínima, ymax = abcisa máxima */
```

```
{
  xvpmin = xmin ;
  xvpmax = xmax ;
  yvpmin = ymin ;
  yvpmax = ymax ;
}
```

4.1.1.21 Limpieza de la pantalla de visión

```
void clearviewport_lcd (void)

/* Limpia el puerto de visión actual */

{
  int address ;
  int addressmin ;
  int addressmax ;
  int nochars ;
  int iy ;

  addressmin = 30 * yvpmin + xvpmin / 8 ;
  addressmax = 30 * yvpmin + xvpmax / 8 ;
  nochars = addressmax - addressmin ;
  /* Hacer un ciclo */
  for ( iy = yvpmin ; iy <= yvpmax ; iy++ ) {
    /* Limpiar bytes enteros */
    address = 30 * iy + xvpmin / 8 ;
    set_cursor_address ( address % 256, address / 256 ) ;
    clean_lcd ( nochars ) ;
    }
}
```

4.1.1.22 Verificación de un punto en la pantalla de visión

```
boolean in (int ix, int iy)

/* Verifica si un punto está dentro del puerto de visión */
{
  boolean stat ;
  stat = TRUE ;
  if (  ix <  xvpmin ||  ix >  xvpmax || iy < yvpmin || iy >
yvpmax ) stat = FALSE ;
  return ( stat ) ;
}
```

4.1.1.23 Dirección de memoria de despliegue

```
void set_address (byte bylo, byte byhi)

/* bylo = parte baja de la dirección,  byhi =  parte alta de
la dirección */
```

```
  {
  lcd ( 0x08,     bylo );
 /* 6) Parte baja de la Dirección de la ram  */
  lcd ( 0x09,     byhi );
  /* 7) Parte alta de la dirección de la ram (hasta 0f)*/
  }
```

4.1.1.24 Lectura de la RAM de la pantalla

```
void read_pixel (byte x, byte y, byte *onoff)

/* Lectura  de la  RAM del  display a partir de la dirección
(x,y) donde está el cursor */
/* La primera lectura es ocsiosa */

{
  int address ;
  byte temp ;
  byte bit ;
  address = 30 * y + x / 8 ;
  /* Define la dirección */
  set_cursor_address ( address % 256, address / 256 ) ;
  /* Lee el byte */
  outp ( 0x35b, 0x08);
  outp ( 0x35b, 0x09);
  outp ( 0x35f, 0x0d);
  delay_us ( 0  );
  outp ( 0x35b, 0x08);
  delay_us ( 0  );
  outp ( 0x35b, 0x12);

  outp ( 0x35b, 0x13);
  inp ( 0x349 );
  delay_us ( 0  );
  outp ( 0x35b, 0x12);
  delay_us ( 0  );

  outp ( 0x35b, 0x13 ) ;
  temp = inp ( 0x349 ) ;
  delay_us ( 0 ) ;
  outp ( 0x35b, 0x12 ) ;
  delay_us ( 0 ) ;

  /* Determinar el estado del bit */
  bit = x % 8 ;
  *onoff = 0 ;
  switch ( bit ) {
    case 0 : if ( ( temp & 0x01 ) > 0 ) *onoff = 1 ;
       break ;
    case 1 : if ( ( temp & 0x02 ) > 0 ) *onoff = 1 ;
```

```
        break ;
    case 2 : if ( ( temp & 0x04 ) > 0 ) *onoff = 1 ;
        break ;
    case 3 : if ( ( temp & 0x08 ) > 0 ) *onoff = 1 ;
        break ;
    case 4 : if ( ( temp & 0x10 ) > 0 ) *onoff = 1 ;
        break ;
    case 5 : if ( ( temp & 0x20 ) > 0 ) *onoff = 1 ;
        break ;
    case 6 : if ( ( temp & 0x40 ) > 0 ) *onoff = 1 ;
        break ;
    case 7 : if ( ( temp & 0x80 ) > 0 ) *onoff = 1 ;
    }
}
```

4.1.1.25 Definición de puntos (modo gráfico)

```
void putpixel (int x, int y, int on_off)

/* Coloca el pixel en la posición (X,Y) dependiendo del valor
de ON/OFF.
La numeración de pixels se inicia en (0,0)  */


  { int address;
    address = 30 * y + ( x / 8)  ;
    set_cursor_address (( address % 256), (address / 256) );

   if (on_off==1 ) lcd ( 0x0f, x % 8 );
   else lcd ( 0x0e, x % 8 );

  }
```

4.1.1.26 Despliegue de líneas

```
void line_lcd (int ix1, int iy1, int ix2, int iy2,
               byte onoff)

/*  Dibuja  una  línea  entre  los  puntos  (ix1,iy1)  hasta
(ix2,iy2) */

{
  float m ;  /* Pendiente */
  float b ;  /* Coeficiente */
  register int iy ;
  register int ix ;

  if ( iy1 == iy2 ) {
    /* Lineas horizontales */
    if ( ix2 > ix1 )
```

```
        for ( ix = ix1 ; ix <= ix2 ; ix++ ) {
    if ( in ( ix, iy1 ) ) putpixel ( ix, iy1, onoff ) ;
    }
      else {
        ix = ix1 ;
        do {
    if ( in ( ix, iy1 ) ) putpixel ( ix, iy1, onoff ) ;
    ix-- ;
        } while ( ix >= ix2 ) ;
        }
      }
    else {
      if ( ix1 == ix2 ) {
        /* Lineas verticales */
        if ( iy2 > iy1 )
    for ( iy = iy1 ; iy <= iy2 ; iy++ ) {
      if ( in ( ix1, iy ) ) putpixel ( ix1, iy, onoff ) ;
      }
        else {
    iy = iy1 ;
    do {
      if ( in ( ix1, iy ) ) putpixel ( ix1, iy, onoff ) ;
      iy-- ;
    } while ( iy >= iy2 ) ;
    }
        }
      else {
        /* Linea inclinada */
        m = (float) ( iy2 - iy1 ) / (float) ( ix2 - ix1 ) ;
        b = (float) ( iy1 - m * ix1 ) ;
        if ( ix2 > ix1 ) {
    /* Incrementos de X */
    for ( ix = ix1 ; ix <= ix2 ; ix++ ) {
      iy = round ( m * ix + b ) ;
      if ( in ( ix, iy ) ) putpixel ( ix, iy, onoff ) ;
      }
    }
        else {
    /* Decrementos de X */
    ix = ix1 ;
    do {
     iy = round ( m * ix + b ) ;
     if ( in ( ix, iy ) ) putpixel ( ix, iy, onoff ) ;
     ix-- ;
    } while ( ix >= ix2 ) ;
    }
        }
      }
}
```

4.1.1.27 Marco de la pantalla

```
void box_lcd (void)

/* Dibuja un rectángulo en la periferia de la superficie del
LCD */

{
  line_lcd ( 0, 0, 239, 0,1 ) ;
  line_lcd ( 239, 0, 239, 127,1 ) ;
  line_lcd ( 239, 127, 0, 127,1 ) ;
  line_lcd ( 0, 127, 0, 0,1 ) ;
}
```

4.1.2 Funciones de control para el Puerto de comunicación Serie RS-232.

4.1.2.1 Inicialización de la interfaz serie RS-232.

```
void inic_rs232 (int vel_trans, int numero_bits, int
                 paridad,int bit_paro)

/* Inicializa la interfaz serie RS-232-C */
/*vel_trans = velocidad de transmisión, número_bits = Número
de bits de información por byte, paridad= paridad en el byte,
bit_paro = Número de bits de paro
{
  union REGS regs;
  byte initcode ;
  switch ( vel_trans ) {
    case 1200 : initcode = 128 ;
       break ;
    case 2400 : initcode = 160 ;
       break ;
    case 4800 : initcode = 192 ;
       break ;
    case 9600 : initcode = 224 ;
       break ;
    )
```

```
  switch ( paridad ) {
    case 0 : break ;
    case 1 : initcode = initcode | 8 ;
       break ;
    case 2 : initcode = initcode | 24 ;
       break ;
    }
    switch ( bit_paro ) {
    case 1 : break ;
    case 2 : initcode = initcode | 4 ;
       break ;
    }
  switch ( numero_bits ) {
    case 8 : initcode = initcode | 3 ;
       break ;
    case 7 : initcode = initcode | 2 ;
       break ;
    }
  regs.h.ah = 0x00 ;
  regs.h.al = initcode ;
  regs.x.dx = 0x0000;
  int86 ( 0x14, &regs, &regs );
  /* Deshabilitar las interrupciones */
  outp ( 0x3f9, 0x00 ) ;
  /* Forzar DTR y RTS a bajo */
  outp ( 0x3fc, 0x00 ) ;
}
```

4.1.2.2 Lectura del puerto serie

```
byte leer_puerto (void)

/* Lee un byte del puerto serie */

{
  byte by ;
  /* Forzamos DTR y RTS */
  outp ( 0x3fc, 0x03 ) ;
  /* Verificar si hay algún caracter disponible */
  do {
  } while ( ( (inp ( 0x3fd )) & 0x01 ) == 0x00 ) ;
  /* Bajamos DTR y RTS mientras procesamos la información */

  outp ( 0x3fc, 0x00 ) ;
  by = inp ( 0x3f8 ) ;

  return ( by ) ;
}
```

4.1.2.3 Lectura del buffer del puerto serie

```
void leer_buffer (int dim, byte *buffer)

/* Lee un buffer del puerto serie */

{ int i;

  byte by ;
  /* Forzamos DTR y RTS */
  outp ( 0x3fc, 0x03 ) ;
  /* Verificar si hay algun caracter disponible */
  for ( i=0 ; i < dim ; i++ ) {
    do {
    } while ( ( (inp ( 0x3fd )) & 0x01 ) == 0x00 ) ;
    buffer [i] = inp ( 0x3f8 ) ;
    }
  /* Bajamos DTR y RTS mientras procesamos la información */

  outp ( 0x3fc, 0x00 ) ;
}
```

4.1.2.4 Escritura en el puerto serie

```
void escribe_puerto (byte by)

/* Escribe un byte por la interfaz serie */

{
  /* Checar DSR y DTE */
  do {
  } while ( ( inp ( 0x3fe ) & 0x30 ) == 0x00 ) ;
  /* Verificar si ya se escribió el caracter anterior */
  do {
  } while ( ( inp ( 0x3fd ) & 0x20 ) == 0x00 ) ;
  /* Escribir */
  outp ( 0x3f8, by ) ;
  delay ( 2 ) ;
}
```

4.1.2.5 Escritura en el buffer del puerto serie

```
void escribe_buffer (int dim,, byte *buffer)

/* Escribe un buffer por la interfaz serie */

{ int i;
  byte by;
  for (i=0; i< dim; i++){
    /* Checar DSR y DTE */
```

```
    do {
      by = inp ( 0x3fe ) ;
    } while ( ( by & 0x30 ) == 0x00 ) ;
    /* Verificar si ya se escribio el caracter anterior */
    do {
    } while ( ( inp ( 0x3fd ) & 0x20 ) == 0x00 ) ;
    /* Escribir */
    outp ( 0x3f8, buffer[i] ) ;
    }
}
```

4.1.2.6 Estatus del puerto serie

```
unsigned status_puerto (void)

/* Obtiene el estado de la interfaz serie */

{
  union REGS regs;
  regs.h.ah = 0x03;
  regs.x.dx = 0x0000 ;
  int86 ( 0x14, &regs, &regs );
  return ( regs.x.ax ) ;
}
```

4.1.3 Funciones de control del generador de pulsos programable

4.1.3.1 Programación del generador de pulsos (COP)

```
void far cop (float fmuestreo)
{
  unsigned  rega ;
  float fclock ;
  byte bylorega ;
  byte byhirega ;

  /* Define la frecuencia del reloj de alimentación al COP.
*/
  fclock = 2100000.0 ;
  if ( fmuestreo < fclock )   {
    rega = (unsigned) ( fclock / ( 8 * fmuestreo ) -1 );
    bylorega = (byte) ( rega % 256 ) ;
    byhirega = (byte) ( rega / 256 ) ;

  outp (0x353, 0x30);       /*Carga el registro paralelo-
                             serie con el dato adecuado*/
  outp (0x34b, 0x00);      /* habilita al cop*/
```

```
outp (0x348,  0x02);      /*Vacian el registro
                           paralelo_serie y cargan los
                           datos al COP bit por bit*/
outp (0x348, 0x02);
outp (0x348, 0x00);
outp (0x348, 0x00);
outp (0x348, 0x02);

outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);
outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);
outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x00);   /*....se termina de programar*/

outp (0x34b, 0x00);      /*Deshabilita el COP*/

outp (0x353,  0x21);      /*Carga el siguiente dato al
                           registro paralelo-serie*/

outp (0x34b, 0x00);      /*Rehabilita el COP*/

outp (0x348, 0x02);      /*Vacía los datos del  registro
```

```
                                paralelo-serie y  los carga en
                                el COP......*/
outp (0x348, 0x02);
outp (0x348, 0x00);
outp (0x348, 0x00);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);
outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);

outp (0x353, byhirega);  /*carga la parte alta del
                          registro con  la frecuencia de
                          muestreo*/

outp (0x348, 0x00);     /*se repite el proceso*/
outp (0x348, 0x00);

outp (0x348, 0x02);
outp (0x348, 0x02);
```

```
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x353, byloregа);   /*parte baja del registro que
                           contine la  frecuencia de
                           muestreo*/
outp (0x348, 0x00);
outp (0x348, 0x00);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
outp (0x348, 0x02);
outp (0x348, 0x04);
outp (0x348, 0x04);

outp (0x348, 0x02);
```

```
    outp (0x348, 0x02);
    outp (0x348, 0x04);
    outp (0x348, 0x04);

    outp (0x348, 0x02);
    outp (0x348, 0x02);
    outp (0x348, 0x04);
    outp (0x348, 0x04);

    outp (0x348, 0x02);
    outp (0x348, 0x02);
    outp (0x348, 0x04);
    outp (0x348, 0x04);

    outp (0x348, 0x02);
    outp (0x348, 0x02);
    outp (0x348, 0x04);
    outp (0x348, 0x04);

    outp (0x348, 0x02);
    outp (0x348, 0x02);
    outp (0x348, 0x04);
    outp (0x348, 0x04);

    outp (0x348, 0x02);
    outp (0x348, 0x02);
    outp (0x348, 0x00);
    outp (0x34b, 0x00);      /*Inhabilitar el cop*/
    }
}
```

4.1.4 Funciones de control del teclado

```
/* Define los codigos a enviar para activar las columnas */

byte codes [4] = { 112, 176, 208, 224 } ;

/* Define los codigos de las letras en función de la línea y
la columna */

char matrix [5][4] = { '0', '1', '4', '7',
                       '.', '2', '5', '8',
                       13, '3', '6', '9',
                       25,  24,  8,  26,
                       'D', 'M', 'S', 'A' } ;
```

4.1.4.1 Programación del circuito 8255

```
char tec (void)

{  byte columna ;
   byte linea ;
   byte lineadecodificada ;
   byte decodificar () ;
   char c ;
/* Palabra de control hacia el 8255 */
   outp ( 857, 146 ) ;


   for (;;) {
/* Para todas las columnas */
     for ( columna = 1 ; columna <= 4 ; columna++ ) {
/* Envia el codigo hacia el 8255 para activar las columnas*/

       outp ( 849, codes[columna-1] ) ;
/* Lee el byte del puerto A que indica la línea activa */
       linea = inp ( 833 ) ;
/* Decodifica el número de la columna */
       lineadecodificada = decodificar ( linea ) ;
       if ( lineadecodificada != 0 ) {
   c = matrix [lineadecodificada-1][columna-1] ;
   return ( c ) ;
   }
       }
     }
}
```

4.1.4.2 Verificación de tecleo

```
char tectime (void)

/* Verifica  si hay  caracteres tecleados  durante un cierto
intervalo de tiempo,
   si no regresa 0 */

{  byte columna ;
   byte linea ;
   byte lineadecodificada ;
   byte decodificar () ;
   int count ;
   char c ;
/* Palabra de control hacia el 8255 */
   outp ( 857, 146 ) ;

 for ( count = 0 ; count < 250 ; count++ ) {
/* Para todas las columnas */
     for ( columna = 1 ; columna <= 4 ; columna++ ) {
```

```
/* Envia  el codigo  hacia el 8255 para activar las columnas
*/
        outp ( 849, codes[columna-1] ) ;
/* Lee el byte que indica la linea activa */
        linea = inp ( 833 ) ;
/* Decodifica el número de la columna */
        lineadecodificada = decodificar ( linea ) ;
        if ( lineadecodificada != 0 ) {
    c = matrix [lineadecodificada-1][columna-1] ;
    return ( c ) ;
    } /*if*/
        } /*columna*/
      } /*count*/     return ( 0x00 ) ; }
```

4.1.4.3 Decodificación

byte decodificar (byte linea)

/* Decodifica el número de la columna. Solo se utilizan 5 columnas que contienen los bits menos significativos. Los 3 bits mas significativos son 1s. Cuando se activa una línea regresa un 0. Las líneas se numeran como 1 la superior y 5 la inferior. La función regresa un 0 al llamador si no se activo ninguna línea. */

```
{ byte by ;
  switch ( linea ) {
    case 239 : by = 1 ;
          break ;
    case 247 : by = 2 ;
          break ;
    case 251 : by = 3 ;
          break ;
    case 253 : by = 4 ;
          break ;
    case 254 : by = 5 ;
          break ;
    default  : by = 0 ;
          break ;
    }
  if ( by != 0 ) delay ( 1200 ) ;
  return ( by ) ;
}
```

4.1.5 Funciones de conversión

4.1.5.1 Conversión de Octeto a BCD

```
byte bytetobcd (byte by)

   /* Convierte un número BY a su representacion en BCD */

  { byte decenas, unidades, bcd ;
  /* Determina si se puede convertir */
  if ( by > 99 ) return ( 0 ) ;
  else {
    decenas = by / 10 ;
    unidades = by % 10;
    bcd = ( decenas << 4 ) + unidades ;
    return ( bcd ) ;
    }
  } /* bytetobcd */
```

4.1.5.2 Conversión de BCD a Octeto

```
 bcdtobyte (byte bcd)

/* Convierte un número BCD a representación hexadecimal */

  { byte decenas, unidades, by ;
  decenas = ( bcd & 0xF0 ) >> 4 ;
  unidades = bcd & 0x0F ;
  by = 10 * decenas + unidades ;
  return ( by ) ;
  }
```

4.1.5.3 Conversión de entero a BCD

```
unsigned inttobcd (int in)

/* Convierte el entero IN a su representación en BCD */

  {
   unsigned bcd;
   if ( in > 9999 )  return ( 0 );
   else {
     bcd = in % 10 ;
     in = in /10;
     bcd = bcd + ((in % 10) << 4) ;
     in = in / 10;
     bcd = bcd + ((in %10) << 8) ;
     in = in / 10;
     bcd = bcd + (( in % 10 ) << 12 ) ;
     return ( bcd );
  }
   }
```

4.1.5.4 Conversión de BCD a entero

```
int bcdtoint (unsigned bcd)

/* Convierte el número BCD a su representación entera */

   { int millares, centenas;
     int decenas, unidades;
     int in;

   millares = ((bcd & 0xf000) >> 12) * 1000 ;
   centenas = ((bcd & 0x0f00) >> 8) * 100 ;
   decenas = (( bcd & 0x00f0) >> 4) * 10 ;
   unidades = (bcd & 0x000f);
   in = millares + centenas + decenas + unidades;
   return ( in );
   }
```

4.1.5.5 Redondeo de enteros

```
int round (float f)

/* Redondea el valor F */

{
  int temp ;

  temp = (int) f ;
  if ( ( f - temp ) >= 0.5 ) temp += 1 ;
  return ( temp ) ;
}
```

4.1.6 Funciones Varias

4.1.6.1 Captura de un caracter

```
char gettec ()

{
  char tecla;
  tecla = tec ();
  return (tecla);
}
```

4.1.6.2 Captura de un número entero

```
int getint ( void )

/* lee un entero */

{
  char s[80];
  gets_kbd (s);
  return ( atoi(s));
}
```

4.1.6.3 Captura de Números con punto flotante

```
float getfloat (void)

/* Lee un número flotante */

{
  char s[80] ;
  gets_kbd ( s ) ;
  return ( atof ( s ) ) ;
}
```

4.1.6.4 Capturta de cadenas de caracteres

```
void gets_kbd (char *s)

/* Captura una cadena de caracteres en la posicion actual */


{
  char ch ;
  byte linea, columna ;
  register int i ;
  extern currline ;
  extern currcol ;

  linea = currline ;
  columna = currcol ;
  i = 0 ;
  do {
    ch = gettec () ;
    switch ( ch ) {
      case 0x08 : /* Backspace */
          if ( i > 0 ) {
            --columna ;
            gotoxy_lcd ( columna, linea ) ;
            writestring ( " " ) ;
```

```
            gotoxy_lcd ( columna, linea ) ;
            --i ;
            }
         break ;
      case 13   : /* carriage return */
         break ;
      default   : /* Caracter correcto */
         s[i] = ch ;
         writechar ( ch ) ;
         ++i ;
         ++columna ;
         break ;
      }
  } while ( ch != 13 ) ;
  s[i] = '\0' ;
}
```

4.1.6.5 Intercambio de variables

```
void swap ( float *s1, float *s2 )

/*Intercambia los contenidos de las variables *s1 y *s2*/

{
  float temp ;
  temp = *s1 ;
  *s1 = *s2 ;
  *s2 = temp ;
}
```

4.1.6.6 Retardo (microsegundos).

```
void delay_us (int ret)

/* Retraso de RET microsegundos */
{
  long i, lt ;
  lt = ret / 7 ;
  for ( i = 0 ; i < lt ; i++ ) ;
}
```

4.1.6.7 Retardo (milisegundos)

```
void delay (int ret)

/* Retraso de RET milisegundos */

{ register int i ;
  int veces ;
  veces = 1000 / 7 ;
```

```
  for ( i = 0 ; i < ret ; i++ ) delay_us ( veces ) ;
}
```

4.1.1.8 Retardo (segundos).

```
void sleep (int seconds)
/* Retraso  SECONDS en segundos */
{
  register int i ;
  for ( i = 0 ; i < seconds ; i++ ) delay ( 1000 ) ;
}
```

4.1.1.9 Verificación de la bandera de comunicación del puerto serie.

```
void read_busy_flag (void)
 /* Regresa hasta que la bandera de BUSY este en 0 */
  {
    byte by ;
    /* do {
      by = inportb ( 0x35c ) ;
    } while ( by & 128 > 0 ) ; */
  }
```

4.2 Presentación del Equipo, definiciones globales y Programa Principal

4.2.1 Presentación del Portátil de Adquisición de Datos.

El logotipo del Instituto de Investigaciones Eléctricas, que se despliega al activar el equipo, está diseñado en base al modo gráfico del display de cristal líquido. Las características de dicho modo de operación de la pantalla, permiten encender o apagar los pixeles de manera independiente.

La matriz que a continuación se muestra, es el conjunto de

números hexadecimales que corresponden a cada grupo de 8 pixeles (byte) que logran tal efecto.

```
/* Arreglo de pixels  para la  graficación del  Logotipo del
IIE */

byte logotipo [96][9] = {
  0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f,   /*
1 */
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xc3, 0xff, 0xff, 0xff, 0xff, 0xff,   /*
5 */
  0xff, 0xff, 0xff, 0x81, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x00, 0xfe, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x7f, 0x00, 0xfe, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff,  0x7f, 0x00,  0xfc, 0xff,  0xff, 0xff, 0xff,
/* 10 */
  0xff, 0xff, 0x3f, 0x00, 0xfc, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x3f, 0x00, 0xf8, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x1f, 0x00, 0xf0, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x0f, 0x00, 0xe0, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x0f, 0x00, 0xe0, 0xff, 0xff, 0xff, 0xff,   /*
15 */
  0xff, 0xff, 0x07, 0x00, 0xc0, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x07, 0x00, 0x80, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x03, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x01, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x01, 0x00, 0x00, 0xfe, 0xff, 0xff, 0xff,   /*
20 */
  0xff, 0xff, 0x01, 0x00, 0x00, 0xfc, 0xff, 0xff, 0xff,
  0xff, 0xff, 0x00, 0x00, 0x00, 0xfc, 0xff, 0xff, 0xff,
  0xff, 0x7f, 0x00, 0x00, 0x00, 0xf8, 0xff, 0xff, 0xff,
  0xff, 0x7f, 0x00, 0x00, 0x00, 0xf0, 0xff, 0xff, 0xff,
  0xff, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /*
25 */
  0xff, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x0f, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x07, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00,   /*
30 */
  0xff, 0x07, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x03, 0x00, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x03, 0x00, 0x7e, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x01, 0x00, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xff, 0x01, 0x00, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00,   /*
35 */
```

```
  0xff, 0x00, 0x80, 0xff, 0x01, 0x00, 0x00, 0x00, 0x00,
  0x7f, 0x00, 0x80, 0xff, 0x03, 0x00, 0xf0, 0xff, 0xff,
  0x7f, 0x00, 0xc0, 0xff, 0x07, 0x00, 0xf8, 0xff, 0xff,
  0x3f, 0x00, 0xe0, 0xff, 0x0f, 0x00, 0xfc, 0xff, 0xff,
  0x1f, 0x00, 0xe0, 0xff, 0x0f, 0x00, 0xfc, 0xff, 0xff,   /*
40 */
  0x00, 0x00, 0xf0, 0xff, 0x1f, 0x00, 0xfe, 0xff, 0xff,
  0x00, 0x00, 0xf0, 0xff, 0x1f, 0x00, 0xfe, 0xff, 0xff,
  0x00, 0x00, 0xf8, 0xff, 0x3f, 0x00, 0xff, 0xff, 0xff,
  0x00, 0x00, 0xf8, 0xff, 0x7f, 0x00, 0xff, 0xff, 0xff,
  0x00, 0x00, 0xfc, 0xff, 0xff, 0x00, 0xff, 0xff, 0xff,   /*
45 */
  0x00, 0x00, 0xfe, 0xef, 0xff, 0x80, 0xff, 0x01, 0x00,
  0x00, 0x00, 0xfe, 0xe7, 0xff, 0x81, 0xff, 0x00, 0x00,
  0x00, 0x00, 0xfe, 0xc7, 0xff, 0xc3, 0x7f, 0x00, 0x00,
  0x00, 0x00, 0xff, 0xc3, 0xff, 0xc3, 0x7f, 0x00, 0x00,
  0x00, 0x00,  0xff, 0x81, 0xff, 0xc7, 0x3f, 0x00, 0x00,  /*
50 */
  0x00, 0x80, 0xff, 0x01, 0xff, 0xef, 0x3f, 0x00, 0x00,
  0x00, 0xc0, 0xff, 0x00, 0xfe, 0xff, 0x1f, 0x00, 0x00,
  0xff, 0xff, 0xff, 0x00, 0xfe, 0xff, 0x1f, 0x00, 0x00,
  0xff, 0xff, 0x7f, 0x00, 0xfc, 0xff, 0x0f, 0x00, 0x00,
  0xff, 0xff, 0x3f, 0x00, 0xf8, 0xff, 0x0f, 0x00,  0x00,  /*
55 */
  0xff, 0xff, 0x3f, 0x00, 0xf8, 0xff, 0x07, 0x00, 0x00,
  0xff, 0xff, 0x3f, 0x00, 0xf0, 0xff, 0x07, 0x00, 0x00,
  0xff, 0xff, 0x1f, 0x00, 0xf0, 0xff, 0x03, 0x00, 0xff,
  0xff, 0xff, 0x1f, 0x00, 0xe0, 0xff, 0x03, 0x80, 0xff,
  0xff, 0xff,  0x0f, 0x00, 0xc0, 0xff, 0x01, 0x80, 0xff,  /*
60 */
  0x00, 0x00, 0x00, 0x00, 0xc0, 0xff, 0x01, 0xc0, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x80, 0xff, 0x00, 0xc0, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x00, 0xe0, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0xe0, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0xf0,  0xff,  /*
65 */
  0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x00, 0xf8, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x00, 0xf8, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0xf8, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0xfc, 0xff,
  0x00, 0x00,  0x00, 0x00, 0x00, 0x10, 0x00, 0xfe, 0xff,  /*
70 */
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xff,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0xff,
  0xff, 0xff, 0xff, 0x0f, 0x00, 0x00, 0x00, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x1f, 0x00, 0x00, 0x00, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x3f, 0x00, 0x00, 0x80, 0xff,  0xff,  /*
75 */
  0xff, 0xff, 0xff, 0x3f, 0x00, 0x00, 0x80, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x7f, 0x00, 0x00, 0xc0, 0xff, 0xff,
  0xff, 0xff, 0xff, 0x7f, 0x00, 0x00, 0xc0, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0xe0, 0xff, 0xff,
```

```
  0xff, 0xff,  0xff, 0xff, 0x01, 0x00, 0xe0, 0xff, 0xff,  /*
80 */
  0xff, 0xff, 0xff, 0xff, 0x03, 0x00, 0xf0, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x07, 0x00, 0xf0, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x07, 0x00, 0xf8, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x0f, 0x00, 0xfc, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x1f, 0x00, 0xfc, 0xff,  0xff,  /*
85 */
  0xff, 0xff, 0xff, 0xff, 0x1f, 0x00, 0xfc, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x3f, 0x00, 0xfe, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0x7f, 0x00, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0xff, 0xff, 0xff,
  0xff, 0xff,  0xff, 0xff, 0xff, 0x00, 0xff, 0xff, 0xff,  /*
90 */
  0xff, 0xff, 0xff, 0xff, 0xff, 0x81, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xc3, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x7f
  );
```

4.2.1.1. Despliegue del logotipo del (IIE)

```
logo ()
{
  register int i, j ;
  int address ;
  int colpixel ;
  int rowpixel ;

  init_lcd ( 16, 40 ) ;
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  init_graph();

  /* Dibujar el logotipo del IIE */

  colpixel = 0 ;
  rowpixel = 16 ;
  for ( i = 0 ; i < 96 ; i++ ) {
   address = 30 * ( i + rowpixel ) + colpixel ;
   set_cursor_address ( address % 256, address / 256 ) ;
   for ( j = 0 ; j < 9 ; j++ ) writechar ( logotipo[i][j] );

    }
  /* Escribir los caracteres */
  write_graph_string ( 8, 13, "INSTITUTO DE" ) ;
  write_graph_string ( 11, 13, "INVESTIGACIONES" ) ;
  write_graph_string ( 14, 13, "ELECTRICAS" ) ;
  sleep ( 5 ) ;
```

```
  /* Título del Equipo */

  win_init () ;
  write_graph_string_big ( 4, 1, " V I B - R A C" ) ;
  box_lcd () ;
  sleep ( 5 ) ;
  close_graph () ;
}
```

4.2.2 Definiciones Globales del programa de control del equipo portátil de Adquisición de Datos

4.2.2.1 Inclusión de librerias estandar (.h)

Con el comando "#include xx.h" (header), se indiaca al preprocesador que se va a incluir archivos externos al archivo fuente; los cuales contienen variables, estructuras y funciones estandar del lenguaje "C".

```
#include <stdio.h>
#include <bios.h>
#include <conio.h>
#include <math.h>
#include <ctype.h>
#include <dos.h>
#include <memory.h>
#include <malloc.h>
```

4.2.2.2 Definición de constantes globales.

```
#define FALSE 0                 /*FALSE = 0*/
#define TRUE  1                 /*TRUE = 1*/
#define MAXPUNTOS 4097          /*MAXPUNTOS =4097*/
```

4.2.2.3 Definición de los Códigos de la téclas.

```
#define DESP_MENU 'D'   /* Desplegar menu */
#define ADQUIS    'A'   /* Realizar la Adquisición */
#define SIG_PUNTO 'S'   /* Despl. párametros  del sig. punto
en la ruta */
#define MODO_OP   'M'   /* Cambiar el modo de operación */
```

4.2.2.4 Definición de los Códigos de reconocimiento y de no_reconocimiento en la comunicación del equipo con la computadora

```
#define ACK     0xFD        /* ACK = 253*/
#define NOACK   0xFE        /* NOACK =254*/
```

4.2.2.5 Definición de Tipos Simples.

```
typedef unsigned char byte ;      /*byte = unsigned char*/
typedef unsigned char boolean ;   /*boolean = unsigned char*/
```

4.2.2.6 Inclusión de librerias prototipo

La longitud del programa abarcó mas de la memoria disponible en el editor de quick C. Motivo por el cual se procedió a la creación de módulos independientes con funciones específicas. Cada uno de éstos módulos cuenta con un archivo conocido como libreria de prototipo, en el cual se declaran las rutinas que lo componen.

```
#include "cop.h"        /*Rutinas del generador de pulsos*/
#include "rs232.h"      /*Rutinas del puerto serie*/
#include "p_comlcd.h"   /*Rutinas del programa de
                          comunicación*/
#include "lcd.h"        /*Rutinas del LCD*/
#include "graflcd.h"    /*Rutinas del módulo graficador*/
#include "logo.h"       /*Rutinas del módulo de logo*/
#include "fft.h"        /*Rutina del cálcilo de la fft*/
#include "keyboard.h"   /*Rutina del teclado*/
#include "port.h"       /*Rutinas del módulo principal*/
```

4.2.2.7 Definición del Apuntador Básico

El apuntador básico es el puntero mas general, debido a que es el que direcciona al programa de control hacia el tipo de adquisición (vibracion,temperatura o manual) que ha de

realizarse.

```
typedef byte far *apuntador_general ;
```

4.2.2.8 Definición de las Estructuras de Datos

Debido a la necesidad de realizar varias capturas para cada tipo de adquisición, se definieron tres estructuras, cada una con los campos correspondientes a cada adquisición.

a) Estructura para un punto de adquisiscion de vibración:

```
typedef struct {
    byte tipov ;              /* Tipo de Registro
                                 (0=Vibr.,1=Temp.,2=Manual) */
  int identificadorv ;        /* Identificador del Punto */
  char equipov[11] ;          /* Nombre del Equipo */
  char descripcionv[31] ;     /* Descripción del punto */
  byte tipo_sensor ;          /* Tipo de sensor */
  float sensibilidad ;        /* Sensibilidad del sensor */
  byte integracion ;          /* Integración */
  byte funcion_adq ;          /* Función de Adquisición */
  byte no_canales ;           /* Número de Canales a adquirir */
  byte fmuestreo ;            /* Frecuencia de Muestreo */
  byte no_muestras ;          /* Número de Muestras a Adquirir /*
  byte no_promedios ;         /* Número de  Promedios a Adquirir
                                 */
  float alertapp  ;            /* Límite de Alerta Valor pico a
                                  pico */
  float alarmapp ;            /* Límite de Alarma Valor pico  a
                                  pico */
  float alertarms ;           /* Límite de Alerta Valor RMS */
  float alarmarms ;           /* Límite de Alarma Valor RMS */
  float vpp1 ;                /* Valor pico a pico */
  float vrms1;                /* Valor RMS */
  float vpp2 ;                /* Valor pico a pico canal 2 */
  float vrms2 ;               /* Valor RMS canal 2 */

  struct dosdate_t  fecha_adq_vib ;  /* Fecha de adquisición
                                        del punto */
    struct dostime_t hora_adq_vib ;   /* Hora de adquisición
                                        del punto */
    apuntador_general buffer_vib ;
    apuntador_general siguientev ;
    } punto_vibracion ;
```

```
typedef punto_vibracion far *apuntador_vibracion ;
```

b) Estructura para un punto de adquisición de temperatura:

```
typedef struct {
    byte tipot ;              /* Tipo de Registro (0,1,2) */
    int identificadort ;      /* Identificador del punto */
    char equipot [11]  ;      /* Nombre del Equipo */
    char descripciont [31] ; /* Descripción del Punto */
    float alertat ;           /* Límite de Alerta */
    float alarmat ;           /* Límite de Alarma */
    float valort  ;            /* Valor de la temperatura
                                  medida */
    struct dosdate_t fecha_adq_temp ;  /*  Fecha de adquisi-
                                  ción del punto */
    struct dostime_t  hora_adq_temp ;   /* Hora de adquisi-
                                  ción del punto */
    apuntador_general siguientet ;

    } punto_temperatura ;

typedef punto_temperatura far *apuntador_temperatura ;

typedef struct {
    byte tipom ;              /* Tipo de Registro (0,1,2) */
    int identificadorm ;      /* Identificador del punto
                                  manual */
    char equipom [11] ;       /* Nombre del Equipo */
    char descripcionm [31] ;  /* Descripción del Punto */
    float alarmabajam ;        /* Límite de Alarma bajo */
    float alertabajam ;        /* Límite de Alerta bajo */
    float alertaaltam ;        /* Límite de Alerta alto */
    float alarmaaltam ;        /* Límite de Alerma alto */
    float valorm ;             /* Valor capturado */
    struct dosdate_t  fecha_adq_man ;  /* Fecha de adquisi-
                                  ción del punto */
    struct dostime_t hora_adq_man ;   /* Hora de adquisición
                                  del punto */
    apuntador_general siguientem ;
    } punto_manual ;
```

c) Estructura para un punto de adquisición manual:

```
typedef punto_manual far *apuntador_manual ;

typedef struct {
    char equipoc [11] ;           /* Nombre del equipo */
    struct dosdate_t  fecha_adq_com ;  /* Fecha de adquisi-
                                  ción del comentario */
    struct dostime_t hora_adq_com ;   /* Hora de adquisición
```

```
                                    del comentario */
    byte indice_comentario ;      /* Número del comentario en
                                    la tabla */
    apuntador_general siguientec ;
    } comentario ;

typedef comentario far *apuntador_comentario ;
```

4.2.2.9 Declaración de las Variables Globales

Las variables globales, como su nombre lo indica, son aquellas, cuyo valor interesa al diseñador que sea reconocido en cualquier parte del programa principal o en lo módulos que lo conforman. Debido a lo anterir, la declaracion de dichas variables se debe llevar a cabo fuera de toda rutina.

```
byte modo_operacion ;          /* Modo de Operacion: 0=Fuera
                                  de Linea, 1=Linea */
boolean ruta_cargada ;        /* Existe una ruta en la memoria
                                  ? */
int numero_bits_rs232 ;        /* Número de bits para la
                                  transmisión */
int vel_trans_rs232 ;         /* Velocidad de Transmisión */
int paridad_rs232 ;            /* Paridad en puerto de
                                  comunicación*/
int bit_paro_rs232 ;           /* Bits de paro para comunica-
                                  ción */
long memoria_libre ;          /* Memoria libre */
long memoria_utilizada ;      /* Memoria utilizada */
int no_puntos_vibracion ;      /* Número de Puntos de vibra-
                                  ción en la ruta */
int no_puntos_manuales ;      /* Número de Puntos manuales en
                                  la ruta */
int no_puntos_temperatura ;   /* Número de Puntos de
                                  temperatura en la ruta */
int no_punt_vib_medidos ;      /* Número de Puntos de
                                  Vibración Medidos */
int no_punt_temp_medidos ;    /* Número de puntos de
                                  temperatura medidos */
int no_punt_man_medidos ;      /* Número de Puntos manuales
                                  medidos */
```

```
int no_punt_vib_noprogs ;     /* Número de Puntos de
                                 vibración no programados */
int no_punt_man_noprogs ;     /* Número de Puntos manuales no
                                 programados */
int no_punt_temp_noprogs ;    /* Número de Puntos de
                                 temperatura no programados*/
int no_comentarios ;          /* Número de Comentarios
                                 definidos */
comentario coment_definidos [50] ; /*Comentarios definidos*/
int no_com_posibles ;         /* Número de Comentarios
                                 posibles en la tabla */
char coment[80][35] ;         /* Tabla de comentarios */
int cont_buffer ;             /* Número de bytes en el buffer
                                 de comunicaciones*/
byte buffer_out [2000 /*17000*/] ; /* Buffer de salida */
apuntador_general primer_punto ;  /*Primer Registro en la
                                    Ruta */
apuntador_general actual_punto ;  /* Actual Registro en la
                                    Ruta */
```

4.2.2.10 Información del último punto de vibración capturado

La graficación se realiza en base al último punto de vibración adquirido. Se definió una estructura que contuviera dechos datos, la cual es refrescada con cada adquisición. El registro en cuestión es el siguiente:

```
struct {
   int identificadorv ;  /* identificador del punto */
   float sensibilidad ;  /* sensibilidad del sensor */
   byte integracion ;    /* Integración ? */
   byte no_canales  ;    /* número de canales */
   byte no_muestras ;    /* número de muestras */
   byte funcion_adq ;    /* función de adquisición*/
   byte fmuestreo ;      /* Frecuencia de muestreo */
   int x[MAXPUNTOS];     /* arreglo de muestras */
   float ampy1[1025 /*512*/] ;
   float ampy2[1025 /*512*/] ;
   }  ultimo_punto ;
```

4.2.3 Rutina Principal

4.2.3.1 Inicialización de variables

```
void inicializar_variables (void)


/* Inicializa las variables globales del Equipo Portátil */

{
  modo_operacion = 0 ; /* Fuera de línea */
  ruta_cargada = FALSE ;
  numero_bits_rs232 = 8 ;
  vel_trans_rs232 = 1200 ;
  bit_paro_rs232 = 1 ;
  paridad_rs232 = 0 ;
  memoria_utilizada = 0 ;
  no_puntos_vibracion = 0 ;
  no_puntos_temperatura = 0 ;
  no_puntos_manuales = 0 ;
  no_punt_vib_medidos = 0 ;
  no_punt_temp_medidos = 0 ;
  no_punt_man_medidos = 0 ;
  no_punt_vib_noprogs = 0 ;
  no_punt_temp_noprogs = 0 ;
  no_punt_man_noprogs = 0 ;
  no_comentarios = 0 ;
  primer_punto = NULL ;
  actual_punto = NULL ;

  /* Determinar la memoria libre */

  memoria_libre = (long) 1024 * _bios_memsize () ;

  /* Inicializar el modulo LCD */
  outp (0x35b,0x04);
  outp (0x35b,0x00);
  outp (0x359,0x92);

}
```

4.2.3.2 Programa Principal

El diagrama de flujo del programa principal se muestra en la figura 4.1

PROGRAMA PRINCIPAL

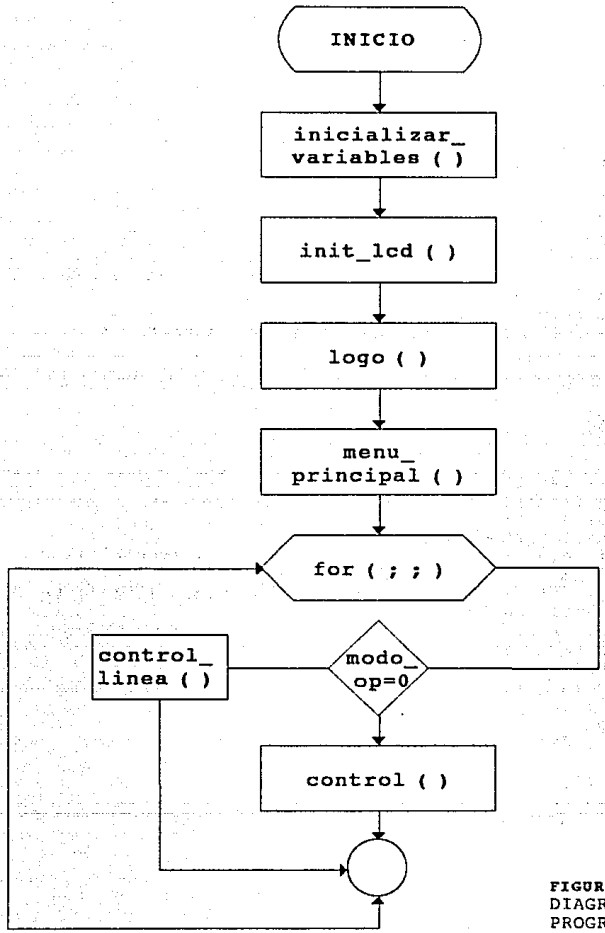

FIGURA 4.1
DIAGRAMA DE FLUJO DEL PROGRAMA PRINCIPAL

```
main()

{
  inicializar_variables () ;
  init_lcd ( 16, 40 ) ;
  logo () ;
  menu_principal  () ;
  for ( ;; ) {
    if ( modo_operacion == 0 ) control () ;
    else control_linea () ;
    }
}
```

a) void control (void)

```
/*Rutina de Control del Equipo de Adquisición de Datos
Portátil. Realiza las siguientes funciones:
1. Lee una tecla del teclado
2. Decodifica y llama a la rutina correspondiente */


{
  char tecla ;
  char st [80];

  do{
  gotoxy_lcd ( 14,11);
  writestring (" ");
  gotoxy_lcd (14,11);
  tecla = (char) gettec () ;
  } while ( (tecla != '1') && (tecla != '2') && (tecla  !=
  '3') && (tecla != '4') && (tecla != '5') &&
     (tecla !=  '6') && (tecla != 'M') && (tecla != 'A') &&
(tecla != 'S') && (tecla != 'D'));
  gotoxy_lcd ( 14, 11);
  writechar ( tecla ) ;
  delay_us (500);
  switch ( tecla ) {
    case '1' : desplegar_parametros () ;
          break ;
    case '2' : desplegar_memoria () ;
          break ;
    case '3' : punto_fuera_ruta () ;
          break ;
    case '4' : graficas () ;
          break ;
    case '5' : definir_comentarios () ;
          break ;
    case '6' : programa_comunicaciones () ;
          break ;
    case DESP_MENU : menu_principal () ;
             break ;
```

```
    case ADQUIS     : adquis () ;
            break ;
    case SIG_PUNTO : sig_punto () ;
            break ;
    case MODO_OP    : cambiar_modo_operacion () ;
            break ;
    default :  gotoxy_lcd (14,11);
         writestring (" ");
         gotoxy_lcd (14,11);
    }

}

b) void control_linea (void)

/* Control de las operaciones en línea */

{
  char tecla ;
  char st[80];

  do{
    gotoxy_lcd ( 14,11);
    writestring (" ");
    gotoxy_lcd (14,11);
    tecla = (char) gettec () ;
    } while  ( (tecla != '1') && (tecla != '2') && (tecla !=
    '3') && (tecla != 'M') ) ;
  gotoxy_lcd ( 14, 11);
  writechar ( tecla ) ;
  delay_us (500);
  switch ( tecla ) {
    case '1' : desplegar_parametros () ;
          break ;
    case '2' : definir_canales () ;
          break ;
    case '3' : graficas () ;
          break ;
    case MODO_OP : cambiar_modo_operacion () ;
           break ;
    default : ;
    }
}
```

4.3 Modo de Operación en Linea

4.3.1 Desplegar Menú Principal

```
menu_principal (void)

/* Despliega el menu de opciones de operación en línea */
{
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 0, 0 ) ;
  writestring ( " EQUIPO PORTATIL DE ADQUISICION DE DATOS" )
;
  gotoxy_lcd ( 15, 2 ) ;
  writestring ( "M E N U " ) ;
  gotoxy_lcd ( 5, 4 ) ;
  writestring ( "(1) Desplegar Parametros" ) ;
  gotoxy_lcd ( 5, 6 ) ;
  writestring ( "(2) Configurar Monitoreo " ) ;
  gotoxy_lcd ( 5, 8 ) ;
  writestring ( "(3) Graficas de Monitoreo Continuo" ) ;
  gotoxy_lcd ( 5, 11 ) ;
  writestring ( "Opcion : " ) ;
  gotoxy_lcd ( 2, 13 ) ;
  writestring ( "F3- Modo-Operacion" ) ;
  control_linea ();
}
```

4.3.1.1 Desplegar Parámetros

```
void desplegar_parametros (void)

/*  Despliega  los  parametros  principales de Operación del
Equipo */

{  char vel;

  gotoxy_lcd ( 0,0);
  clean_lcd ( 3840);
  gotoxy_lcd ( 8, 1);
  writestring ( "DESPLIEGUE DE PARAMETROS" ) ;
  gotoxy_lcd ( 5, 5 ) ;
  if ( modo_operacion == 1 )
    writestring ("Modo de operacion: En Linea");
  else writestring("Modo de operacion: Fuera de Linea");
  gotoxy_lcd ( 5, 7 ) ;
  if ( ruta_cargada ) writestring ("Ruta cargada en la
  Memoria:  SI" ) ;
  else writestring ( "Ruta cargada en la Memoria:  NO" ) ;
  gotoxy_lcd ( 5, 9 ) ;
  writestring ("Vel. Transmision [1200 bit/s] ");
  gotoxy_lcd ( 5, 11 );
  writestring ("  1) 2400, 2) 4800, 3) 9600 : ");
  do{
    gotoxy_lcd ( 35, 11 );
```

```
    writestring (" ");
    gotoxy_lcd ( 35,11);
    vel = gettec ();
    } while  ( ( vel != '1') && (vel != '3' ) && (vel !='2')
    && (vel != 13) );
    gotoxy_lcd ( 35,11);
    writechar ( vel );
    delay_us (500);
    switch ( vel ) {
   case '1' :  vel_trans_rs232 = 2400;    /* 2400 b/s */
               break;
   case '2' :  vel_trans_rs232 = 4800;    /* 4800 b/s */
               break;
   case '3' :  vel_trans_rs232 = 9600;    /* 9600 b/s */
               break;
   case 13 :   vel_trans_rs232 = 1200;    /* 1200 b/s */
               break;

   }
  if ( modo_operacion == 0 ) menu_principal ();
  else menu_linea () ;

}
```

4.3.1.2 Configurar monitoreo Continuo

```
void definir_canales (void)

/* Define los canales para la operación en línea */

{ int i;
  char ch ;
  byte tipo_sensor ;
  float getfloat () ;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 7, 0 ) ;
  writestring ( "MONITOREO CONTINUO  Pg. 1" ) ;
  /* Captura el tipo de sensor */
  gotoxy_lcd ( 0, 3 ) ;
  writestring ( "Sensor (0=Desp., 1=Vel., 2=Acel.) : " ) ;
  do {
    gotoxy_lcd ( 36, 3);
    writechar ( ' ' );
    gotoxy_lcd ( 36, 3 ) ;
    ch = gettec () ;
  } while ( ( ch < '0' ) || ( ch > '2' ) ) ;
```

```
  gotoxy_lcd (36,3);
  writechar ( ch );
  delay (500);
tipo_sensor = ch - '0' ;

/* Captura la sensibilidad del sensor */
gotoxy_lcd ( 0, 7 ) ;
writestring ( "Sensibilidad " ) ;
switch ( tipo_sensor ) {
  case 0 : gotoxy_lcd ( 14, 7 ) ;
     writestring ( "(1=100 mV/mil" ) ;
     gotoxy_lcd ( 14, 8) ;
     writestring ( "(2=200 mV/mil) : " ) ;
     do {
       gotoxy_lcd ( 31, 8 ) ;
       writechar ( ' ' );
       ch = gettec () ;
     } while ( ( ch != '1' ) && ( ch !='2' ) ) ;
       gotoxy_lcd ( 31,8);
       writechar ( ch );
       delay (500);
     switch ( ch ) {
     case '1' : ultimo_punto.sensibilidad = 0.100 ;
              break ;
     case '2' : ultimo_punto.sensibilidad = 0.200 ;
              break ;
     }
     break ;
  case 1 : gotoxy_lcd ( 14, 7 ) ;
     writestring ( "(1=100 mV/pulg/seg" ) ;
     gotoxy_lcd ( 14, 8 ) ;
     writestring ( "(2=500 mV/pulg/seg) :  " ) ;
     do {
       gotoxy_lcd ( 36, 8 ) ;
       writechar ( ' ' );
       ch = gettec () ;
     } while ( ( ch != '1' ) && ( ch != '2' ) ) ;
       gotoxy_lcd ( 36,8 );
       writechar (ch);
       delay (500);
     switch ( ch ) {
     case '1' : ultimo_punto.sensibilidad = 0.10 ;
              break ;
     case '2' : ultimo_punto.sensibilidad = 0.50 ;
              break ;
     }
     break ;
  case 2 : gotoxy_lcd ( 14, 7 ) ;
     writestring ( ": 100 mV/G " ) ;
     ultimo_punto.sensibilidad = 0.10;
     break ;
  }
```

```
switch ( tipo_sensor ) {
  case 0 : /* sensor de desplazamiento, no hay integración
                        */
     ultimo_punto.integracion = 0 ;
     break ;
  case 1 : /* sensor de velocidad */
     gotoxy_lcd ( 0, 10 ) ;
     writestring (  "Integracion (0=No, 1=Vel.-Despl.) : "
          ) ;
     do {
       gotoxy_lcd ( 36, 10 ) ;
       writechar ( ' ' );
       ch = gettec () ;
     } while ( ( ch != '0' ) && ( ch != '1' ) ) ;
       gotoxy_lcd ( 36, 10);
       writechar ( ch );
       delay (500);
     switch ( ch ) {
       case '0' : ultimo_punto.integracion = 0 ;
       case '1' : if ( ultimo_punto.sensibilidad == 0.10 )

              ultimo_punto.integracion = 1 ;
            else ultimo_punto.integracion = 2 ;
       }
     break ;
  case 2 : /* sensor de aceleración */
     gotoxy_lcd ( 0,9 ) ;
     writestring ( "Integracion (0=No, 1=Acel.-Vel.) : " )
                        ;
     do {
       gotoxy_lcd ( 36, 9 ) ;
       writechar ( ' ' );
       ch = gettec () ;
     } while ( ( ch != '0' ) && ( ch != '1' ) ) ;
       gotoxy_lcd ( 36,9);
       writechar ( ch );
       delay (500);
     switch ( ch ) {
       case '0' : ultimo_punto.integracion = 0 ;
       case '1' : ultimo_punto.integracion = 3 ;
       }
     break ;
  }
gotoxy_lcd ( 0, 12 ) ;
writestring ( "Numero de Canales (1,2) : " ) ;
do {
  gotoxy_lcd ( 26, 12 ) ;
  writechar ( ' ' );
  ch = gettec () ;
} while ( ( ch != '1' ) && ( ch != '2' ) ) ;
  gotoxy_lcd ( 26, 12);
  writechar ( ch );
```

```
    delay (500);
  switch ( ch ) {
    case '1' : ultimo_punto.no_canales = 1 ;
         break ;
    case '2' : ultimo_punto.no_canales = 2 ;
         break ;
    }

  /* Segunda pagina */

  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 7, 0 ) ;
  writestring ( "MONITOREO CONTINUO Pg. 2" ) ;
  gotoxy_lcd ( 0, 2 ) ;
  writestring ( "Frecuencia de Muestreo " ) ;
  gotoxy_lcd ( 2, 4 ) ;
  writestring ( "( 0=500 Hz    1= 1 kHz    2= 2 kHz " ) ;
  gotoxy_lcd ( 2, 6 ) ;
  writestring ( "  3= 5 kHz    4=10 kHz    5=20 kHz" ) ;
  gotoxy_lcd ( 2, 8 ) ;
  writestring ( "  6=30 kHz    7=40 kHz    8=50 kHz " ) ;
  gotoxy_lcd ( 2, 10 ) ;
  writestring ( "              9=100 kHz ) :" ) ;

  do {
    gotoxy_lcd ( 29, 10 ) ;
    writechar ( ' ' );
    ch= gettec () ;
  } while (ch<'0' || ch>'9' ) ;
    gotoxy_lcd ( 29, 10);
    writechar ( ch );
    delay (500);

  /* Decodificar los valores de la frecuencia de muestreo */
  ultimo_punto.fmuestreo = ch - '0' ;
  gotoxy_lcd ( 0, 13 ) ;

  menu_linea () ;
}
```

4.3.1.3 Gráficas

a) void gráfica_señal_linea (void)

```
/* Grafica continuamente una senal */

{
  register int i,  j ;
  char ch ;
  float deltat ;
```

```
float t[1025], y[1025] ;
float mint, maxt ;
float sminy, smaxy ;
float miny, maxy ;
int canal ;
float fmuestreo;
int no_muestras;
int no_veces ;
boolean refresh ;
double fabs () ;

/* Limpiar la pantalla */
gotoxy_lcd ( 0, 0 ) ;
clean_lcd ( 3840 ) ;
/* Decodificar los valores de la frecuencia de muestreo */

switch (ultimo_punto.fmuestreo ) {
  case 0 : fmuestreo = 500.0 ;
     break ;
  case 1 : fmuestreo = 1000.0 ;
     break ;
  case 2 : fmuestreo = 2000.0 ;
     break ;
  case 3 : fmuestreo = 5000.0 ;
     break ;
  case 4 : fmuestreo = 10000.0 ;
     break ;
  case 5 : fmuestreo = 20000.0 ;
     break ;
  case 6 : fmuestreo = 30000.0 ;
     break ;
  case 7 : fmuestreo = 40000.0 ;
     break ;
  case 8 : fmuestreo = 50000.0 ;
     break ;
  case 9 : fmuestreo = 100000.0 ;
     break;
  }

no_muestras = 512 ;

/* Preparar el hardware /
/* gotoxy_lcd ( 0,2);
writestring ("Configurando al Hardware..."); */

cop (  fmuestreo );

/* Programacion de los integradores */
outp (  832, 16 ) ;  /* Descargar el capacitor de realimen
                        tación */
switch (ultimo_punto.integracion){
  case 0 : outp ( 832, 17 ); /* Sin Integración */
```

```
      break;
    case 1  : outp  ( 832,  66 );  /* Integracion vel-despl.
sensor 100 mv/pulg/seg */
      break;
    case 2  : outp  ( 832,  68 );  /* Integracion vel-despl.
sensor 500 mv/pulg/seg */
      break;
    case 3 : outp (  832,  72  );  /*  Integracion acel-vel.
sensor 500 mv/G */
      break;
    }

  canal = 1 ;

  /* Inicializa el LCD en modo gráfico */
  init_graph () ;

  no_veces = 1 ;

  do {
    /* Realizar la adquisición */
    /* gotoxy_lcd (0,4);
    writestring ("Adquiriendo serie de tiempo..."); */

    senaltiempo  (  no_muestras, ultimo_punto.no_canales, 0,
ultimo_punto.x ) ;

    /* Calcular los arreglos de tiempos y de senal en
unidades de ingenieria */

    deltat = 1.0 / fmuestreo ;
    if ( ultimo_punto.no_canales == 1 ) {
      for ( i=0 ; i < no_muestras ; i++ ) {
  t [i] = deltat * i ;
  y [i] = ( 0.0024414 * ultimo_punto.x[i] )
                      / ultimo_punto.sensibilidad ;
  }
      }
    else {
      if ( canal == 1 ) {
  j = 0 ;
  for ( i=0 ; i < 2 * no_muestras  ; i++ ) {
    if ( ( i % 2 ) == 0 ) {
      t [j] = deltat * j ;
      y [j] = ( 0.0024414 * ultimo_punto.x[i] )
                          / ultimo_punto.sensibilidad ;
      ++j ;
      }
    }
  }
      else {
  j = 0 ;
```

```
  for ( i=0 ; i < 2 * no_muestras ; i++ ) {
    if ( ( i % 2 ) != 0 ) {
      t [j] = deltat * j ;
      y [j] = ( 0.0024414 * ultimo_punto.x[i] )
                          / ultimo_punto.sensibilidad ;
      ++j ;
      }
    }
  }
      }
    /* Determinar los límites de la gráfica */
    f_min_max ( no_muestras, t, &mint, &maxt ) ;
    f_min_max ( no_muestras, y, &miny, &maxy ) ;
    /* Determinar el valor absoluto mayor en Y y agregar un
20 % */
    if ( fabs ( maxy ) > fabs ( miny  ) )  miny =  - (float)
fabs ( maxy ) ;
    else maxy = (float) fabs ( miny ) ;
    if ( no_veces == 1 ) {
      refresh = TRUE ;
      sminy = 1.2 * miny ;
      smaxy = 1.2 * maxy ;
      }
    else {
      /* Verificar si hay overflow en la gráfica */
      if ( ( fabs ( maxy ) > fabs ( smaxy ) ) ||
     ( fabs ( miny ) > fabs ( sminy ) ) ) {
  refresh = TRUE ;
  sminy = 1.2 * miny ;
  smaxy = 1.2 * maxy ;
  }
      else refresh = FALSE ;
      }
    if ( refresh ) {
      /* Inicializar la ventana */
      win_init () ;

      /* Llamar a las escalas */
      scale_lin_x ( mint, maxt ) ;
      scale_lin_y ( sminy, smaxy ) ;

      /* Trazar los ejes */
      dr_lin_x_ax () ;
      dr_lin_y_ax () ;

      /* Dibuja la curva */
      scatter_plot_data_2 ( no_muestras, t, y ) ;

      /* Colocamos las escalas en los ejes */
      lab_lin_x_ax ( "%6.4f" ) ;
      lab_lin_y_ax () ;
      /* Colocamos los títulos en los ejes */
```

```
      title_x_ax ( "Tiempo (Segs)" ) ;
      title_y_ax ( "mils" ) ;
      title_window ( "SERIE DE TIEMPO" ) ;
      }
    else {
      refresh_scatter ( no_muestras, t, y ) ;
      }
    /* Leer */
    ch = tectime () ;
    if ( ch != 0x00 ) {
      switch ( ch ) {
  case '1' : if ( canal == 1 ) {
             }
           else {
             canal = 1 ;
             refresh = TRUE ;
             }
           break ;
  case '2' : if ( canal == 2 ) {
             }
           else {
             canal = 2 ;
             refresh = TRUE ;
             }
  default : ;
  }
      }
    ++no_veces ;
  } while ( ch != 13 ) ;
  close_graph () ;

}


b) void gráfica_espectro_línea (void)

/* Grafica continuamente un espectro */

{
  register int i,  j ;
  char ch ;
  float deltaf ;
  float y[1025], f[1025], a[1025], cero[1025] ;
  float minf, maxf ;
  float miny, maxy ;
  float sminy, smaxy ;
  int canal ;
  float fmuestreo;
  int no_muestras;
  boolean refresh ;
  int no_veces ;
```

```
float deltat ;
double fabs () ;
double sqrt () ;

gotoxy_lcd ( 0, 0 ) ;
clean_lcd ( 3840 ) ;
/* Decodificar los valores de la frecuencia de muestreo */

switch (ultimo_punto.fmuestreo ) {
  case 0 : fmuestreo = 500.0 ;
     break ;
  case 1 : fmuestreo = 1000.0 ;
     break ;
  case 2 : fmuestreo = 2000.0 ;
     break ;
  case 3 : fmuestreo = 5000.0 ;
     break ;
  case 4 : fmuestreo = 10000.0 ;
     break ;
  case 5 : fmuestreo = 20000.0 ;
     break ;
  case 6 : fmuestreo = 30000.0 ;
     break ;
  case 7 : fmuestreo = 40000.0 ;
     break ;
  case 8 : fmuestreo = 50000.0 ;
     break ;
  case 9 : fmuestreo = 100000.0 ;
     break;
  }

 no_muestras = 512 ;
 canal = 1 ;

/* Preparar el hardware */
cop (  fmuestreo );

/* Programación de los integradores */
outp ( 832, 16 ) ;  /* Descargar el capacitor de realimen-
                       tacion */
switch (ultimo_punto.integracion){
  case 0 : outp ( 832, 17 ); /* Sin Integración */
     break;
  case 1  : outp ( 832, 66 ); /* Integración vel-despl.
                                 sensor 100 mv/pulg/seg */
     break;
  case 2 : outp ( 832, 68 ); /* Integración vel-despl.
                                 sensor 500 mv/pulg/seg */
     break;
  case 3  : outp ( 832, 72 ); /* Integración acel-vel.
                                 sensor 500 mv/G */
     break;     }
```

```
/* Inicializa el modo gráfico del LCD */
init_graph () ;

no_veces = 1 ;

do {
   /* Realizar la adquisición */

   senaltiempo (  no_muestras, ultimo_punto.no_canales,0,
                  ultimo_punto.x ) ;

   /* Calcular los arreglos de tiempos y de senal en
      unidades de ingenieria */

   deltat = 1.0 / fmuestreo ;
   if ( ultimo_punto.no_canales == 1 ) {
     for ( i=0 ; i < no_muestras ; i++ ) {
y [i] = ( 0.0024414 * ultimo_punto.x[i] )
                      / ultimo_punto.sensibilidad ;
}
     }
   else {
     if ( canal == 1 ) {
j = 0 ;
for ( i=0 ; i < 2 * no_muestras  ; i++ ) {
   if ( ( i % 2 ) == 0 ) {
     y [j] = ( 0.0024414 * ultimo_punto.x[i] )
                      / ultimo_punto.sensibilidad ;
     ++j ;
     }
   }
}
     else {
j = 0 ;
for ( i=0 ; i < 2 * no_muestras ; i++ ) {
   if ( ( i % 2 ) != 0 ) {
     y [j] = ( 0.0024414 * ultimo_punto.x[i] )
                      / ultimo_punto.sensibilidad ;
     ++j ;
     }
   }
}
     }

   /* Limpiar el arreglo cero */

   for ( i = 0 ; i < no_muestras ; i++ ) cero[i] = 0.0 ;
   /* Calcular el espectro */
   fft ( y, cero, no_muestras ) ;
   /* Calcular los arreglos de amplitud y de frecuencia */
   deltaf = 0.5 * fmuestreo / no_muestras ;
   for ( i = 0 ; i < no_muestras / 2 ; i++ ) {
```

```
      f[i] = deltaf * i ;
      a[i] =  (float) 2.0 * sqrt ( y[i] * y[i] + cero[i] *
              cero[i] ) / no_muestras ;
      }

    /* Determinar los límites de la gráfica */
    f_min_max ( no_muestras / 2, f, &minf, &maxf ) ;
    f_min_max ( no_muestras / 2, a, &miny, &maxy ) ;
    if ( no_veces == 1 ) {
      refresh = TRUE ;
      sminy = 0.0 ;
      smaxy = 1.2 * maxy ;
      }
    else {
      /* Verificar si hay overflow en la gráfica */
      if ( fabs ( maxy ) > fabs ( smaxy ) ) {
refresh = TRUE ;
sminy = 0.0 ;
smaxy = 1.2 * maxy ;
}
      else refresh = FALSE ;
      }
    if ( refresh ) {
      /* Inicializar la ventana */
      win_init () ;

      /* Llamar a las escalas */
      scale_lin_x ( minf, maxf ) ;
      scale_lin_y ( sminy, smaxy ) ;

      /* Trazar los ejes */
      dr_lin_x_ax () ;
      dr_lin_y_ax () ;
      /* Dibuja la curva */
      scatter_plot_data_2 ( no_muestras / 2, f, a ) ;

      /* Colocamos las escalas en los ejes */
      lab_lin_x_ax ( "%8.1f" ) ;
      lab_lin_y_ax () ;
      /* Colocamos los títulos en los ejes */
      title_x_ax ( "    Frec. (Hz)" ) ;
      title_y_ax ( "mils" ) ;
      title_window ( "ESPECTRO LINEAL" ) ;
      }
    else {
      refresh_scatter ( no_muestras / 2, f, a ) ;
      }
    /* Leer */
    ch = tectime () ;
    if ( ch != 0x00 ) {
      switch ( ch ) {
case '1' : if ( canal == 1 ) {
```

```
            }
          else {
            canal = 1 ;
            refresh = TRUE ;
            }
          break ;
  case '2' : if ( canal == 2 ) {
            }
          else {
            canal = 2 ;
            refresh = TRUE ;
            }
  default : ;
  }
     }
    ++no_veces ;
  } while ( ch != 13 ) ;
  close_graph () ;

}
```

4.3.1.4 Cambio de Modo de Operación

```
void cambiar_modo_operación (void)

/* Cambia el modo de operación del equipo */

{
   char tecla;
  /* Limpiar la pantalla */
   gotoxy_lcd ( 0, 0 ) ;
   clean_lcd ( 3840 ) ;
   gotoxy_lcd ( 0, 2 );
   if ( modo_operacion == 1 )
     writestring ( "Modo de Operacion : En Linea" ) ;
   else writestring ( "Modo de Operacion : Fuera de Linea" )
;
   gotoxy_lcd ( 0, 4 );
   writestring ( " 0) Cambio de modo" ) ;
   gotoxy_lcd ( 0, 6 ) ;
   writestring ( " 1) Mismo modo ");
   gotoxy_lcd ( 0, 8 ) ;
   writestring ( " Opcion : " ) ;

   do {
      gotoxy_lcd ( 10, 8);
      writestring (" ");
      gotoxy_lcd (10,8);
      tecla = gettec ();
   } while ( tecla < '0' || tecla > '1' );
      gotoxy_lcd ( 10,8);
```

```
      writechar ( tecla);
      delay (500);
   if ( tecla  == '0' ) {
     if ( modo_operacion == 0 ) modo_operacion = 1 ;
     else modo_operacion = 0 ;
     }
   if ( modo_operacion == 0 ) menu_principal ();
   else menu_linea () ;

}
```

4.4 Modo de operación Fuera de Línea

4.4.1 Desplegar Menú Pricipal

```
void menu_principal (void)

/* Despliega el menu principal del Equipo */

{
  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 );
  clean_lcd ( 3840 );
  gotoxy_lcd ( 0, 0 );
  writestring ( " EQUIPO PORTATIL DE ADQUISICION DE DATOS");

  gotoxy_lcd ( 15, 2);
  writestring (" M E N U ");
  gotoxy_lcd ( 5, 4 );
  writestring ("(1) Desplegar Parametros");
  gotoxy_lcd ( 5, 5);
  writestring ("(2) Desplegar Memoria");
  gotoxy_lcd ( 5,6);
  writestring ("(3) Punto No Programado");
  gotoxy_lcd ( 5,7);
  writestring ("(4) Graficas");
  gotoxy_lcd ( 5,8);
  writestring ("(5) Definir Comentarios");
  gotoxy_lcd ( 5,9);
  writestring ("(6) Programa de Comunicaciones");
  gotoxy_lcd ( 5,11);
  writestring ("Opcion : ");
  gotoxy_lcd ( 1, 13);
  writestring ("F1- Adquisicion     F2- Siguiente-Punto");
  gotoxy_lcd ( 1,14);
  writestring ("F3- Modo-Operacion  F4- Desplegar-Menu ");

}
```

4.4.1.1 Desplegar Parámetros

4.4.1.2 Desplegar Memoria.

```
void desplegar_memeoria (void)

/* Despliega los parámetros de la memoria del Equipo */

{ char st [80] ;
  char cr;

  set_char_size (16,40);
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;  /* limpiar pantalla */
  gotoxy_lcd ( 10, 0 ) ;
  writestring ( "DESPLIEGUE DE MEMORIA " ) ;
  gotoxy_lcd ( 3, 2 ) ;
  sprintf ( st, "Memoria Utilizada (Kbytes) : %3d", memoria-
                                  _utilizada / 1024 ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 4 ) ;
  sprintf ( st, "Memoria Libre (Kbytes)     : %3d", memoria-
                                  _libre / 1024 ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 6 ) ;
  writestring ( "No. de Puntos Medidos en la Ruta " ) ;
  gotoxy_lcd ( 3, 7 ) ;
  sprintf ( st, "  Vibracion :%3d    Temperatura :%3d",
  no_punt_vib_medidos,
  no_punt_temp_medidos ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 8 ) ;
  sprintf ( st, "  Manuales  :%3d", no_punt_man_medidos ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 10 ) ;
  writestring ( "No. de Puntos por Medir " ) ;
  gotoxy_lcd ( 3, 11 ) ;
  sprintf ( st, "  Vibracion :%3d    Temperatura :%3d",
  no_puntos_vibracion - no_punt_vib_medidos,
  no_puntos_temperatura - no_punt_temp_medidos ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 12 ) ;
  sprintf ( st, "  Manuales  :%3d",
  no_puntos_manuales - no_punt_man_medidos ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 3, 14 ) ;
  sprintf (  st, "No. de Comentarios Definidos    :%3d",
  no_comentarios ) ;
  writestring ( st ) ;
  gotoxy_lcd (3,15);
  writestring ("<CR> Continuar...");
```

```
  do{
    cr = (char) gettec ();
    } while (cr != 13);

  menu_principal ();

}
```

4.4.1.3 Punto No Programado

```
void punto_fuera_ruta (void)

/* Permite definir al usuario un punto no programado */

{
  char tipo_punto ;
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;  /* Limpiar la pantalla */
  gotoxy_lcd ( 10, 1 ) ;
  writestring ( "PUNTO NO PROGRAMADO" ) ;
  gotoxy_lcd ( 2, 4 ) ;
  writestring ( "Tipo de Adquisicion  " ) ;
  gotoxy_lcd ( 2, 6 ) ;
  writestring ( "(1)  Vibracion");
  gotoxy_lcd ( 2, 8 ) ;
  writestring ( "(2)  Temperatura");
  gotoxy_lcd ( 2,10 );
  writestring ( "(3)  Manual") ;
  gotoxy_lcd ( 2,12 );
  writestring ( "     Opcion :");
  do {
    gotoxy_lcd ( 16, 12 ) ;
    writestring (" ");
    gotoxy_lcd ( 16,12);
    tipo_punto = (char) gettec () ;
    } while (( tipo_punto < '1') || (tipo_punto > '3'));
    gotoxy_lcd (16, 12);
    writechar ( tipo_punto ) ;
    delay_us ( 500 );
  switch ( tipo_punto ) {
    case '1' : punto_fuera_ruta_vibracion () ;
         break ;
    case '2' : punto_fuera_ruta_temperatura () ;
         break ;
    case '3' : punto_fuera_ruta_manual () ;
         break ;
    }
}
```

a) void punto_fuera-ruta_vibración (void)

```
/* Define un punto de vibración no programado */

{ int i;
  char ch ;
  apuntador_vibracion pvib ;
  apuntador_vibracion pv ;
  apuntador_temperatura pt ;
  apuntador_manual pm ;
  int tipo_registro ;
  /*float getfloat () ;*/
  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  pvib = (apuntador_vibracion) _fmalloc  ( (unsigned) sizeof
( punto_vibracion ) ) ;
  if ( pvib == NULL ) {
    gotoxy_lcd ( 0, 5 ) ;
    writestring ( "Error : Memoria Insuficiente " ) ;
    delay_us ( 400 ) ;
    menu_principal ();
    }
  else {
    if ( primer_punto == NULL ) {
      primer_punto = (apuntador_general)pvib ;
      actual_punto = (apuntador_general)pvib ;
      pvib->siguientev = NULL ;
      }
    else {
      /* Enlazar  el registro en la lista de puntos de la
         ruta */
      tipo_registro = (int) *actual_punto ;
      switch ( tipo_registro ) {
  case 0 : pv = (apuntador_vibracion) actual_punto ;
        pvib->siguientev = pv->siguientev ;
        pv->siguientev = (apuntador_general) pvib ;
        break ;
  case 1 : pt = (apuntador_temperatura) actual_punto ;
        pvib->siguientev = pt->siguientet ;
        pt->siguientet = (apuntador_general) pvib ;
        break ;
  case 2 : pm = (apuntador_manual) actual_punto ;
        pvib->siguientev = pm->siguientem ;
        pm->siguientem = (apuntador_general) pvib ;
        break ;
  }
      }
    pvib->tipov = 0 ;
    pvib->equipov [0] = '\0' ;
    pvib->descripcionv [0] = '\0' ;
    gotoxy_lcd ( 7, 0 ) ;
```

```
writestring ( "PUNTO DE VIBRACION  Pg. 1" ) ;
/* Captura el identificador del punto */
gotoxy_lcd ( 0, 3 ) ;
writestring ( "Punto : " ) ;
do {
  gotoxy_lcd ( 8, 3 ) ;
  writestring ( "      " ) ;
  gotoxy_lcd ( 8, 3 ) ;
  pvib->identificadorv = getint () ;
  } while ( pvib->identificadorv % 10 == 0 ) ;
/* Captura el tipo de sensor */
gotoxy_lcd ( 0, 5 ) ;
writestring ( "Sensor (0=Desp., 1=Vel., 2=Acel.) : " ) ;

do {
  gotoxy_lcd ( 36, 5 );
  writestring (" ");
  gotoxy_lcd ( 36, 5 ) ;
 ch = (char) gettec () ;
} while ( ( ch < '0' ) || ( ch > '2' ));
  gotoxy_lcd ( 36, 5 );
  writechar ( ch ) ;

pvib->tipo_sensor = ch - '0' ;

/* Captura la sensibilidad del sensor */
gotoxy_lcd ( 0, 7 ) ;
writestring ( "Sensibilidad " ) ;
switch ( pvib->tipo_sensor ) {
  case 0 : gotoxy_lcd ( 14, 7 ) ;
           writestring ( "(1=100 mV/mil" ) ;
           gotoxy_lcd ( 14, 8) ;
           writestring ( "(2=200 mV/mil) : " ) ;
           do {
               gotoxy_lcd ( 31,8);
               writestring (" ");
               gotoxy_lcd ( 31, 8 ) ;
               ch = (char) gettec () ;
               }while (( ch != '1') && ( ch != '2'));
           gotoxy_lcd ( 31, 8 );
           writechar ( ch ) ;
 switch ( ch ) {
    case '1' : pvib->sensibilidad = (float) 0.100 ;
               break ;
    case '2' : pvib->sensibilidad = (float) 0.200 ;
               break ;
                }
     break ;
  case 1 : gotoxy_lcd ( 14, 7 ) ;
     writestring ( "(1=100 mV/pulg/seg" ) ;
     gotoxy_lcd ( 14, 8 ) ;
     writestring ( "(2=500 mV/pulg/seg) :  " ) ;
```

```
      do {
     gotoxy_lcd ( 36, 8 ) ;
     writestring (" ");
     gotoxy_lcd ( 36,8);
     ch = (char) gettec () ;
    } while ( (ch != '1') && ( ch != '2' ));
     gotoxy_lcd ( 36, 8 );
     writechar ( ch ) ;
      switch ( ch ) {
     case '1' : pvib->sensibilidad = (float) 0.10 ;
              break ;
     case '2' : pvib->sensibilidad = (float) 0.50 ;
              break ;
     }
      break ;
   case 2 : gotoxy_lcd ( 14, 7 ) ;
      writestring ( ": 100 mV/G " ) ;
   /*   do {
     gotoxy_lcd ( 28, 8 ) ;
     ch = gettec () ;
     gotoxy_lcd ( 28, 8 );
     writestring ( ch ) ;
      } while ( ch != '1' ) ;
      switch ( ch ) {
     case '1' :   pvib->sensibilidad = (float)0.10 ;
     }  */
    pvib->sensibilidad = (float) 0.10;
      break ;
  }
switch ( pvib->tipo_sensor ) {
  case 0  : /* sensor de desplazamiento, no hay integra-
                ción */
     pvib->integracion = 0 ;
     break ;
  case 1 : /* sensor de velocidad */
     gotoxy_lcd ( 0, 10 ) ;
     writestring ( "Integracion (0=No,  1=Vel.-Despl.) :
    " ) ;
     do {
    gotoxy_lcd ( 36, 10 ) ;
    writestring (" ");
    gotoxy_lcd ( 36,10);
    ch = (char) gettec () ;
   } while ( ( ch != '0') && (ch != '1' ) );
    gotoxy_lcd ( 36, 10 );
    writechar ( ch ) ;
     switch ( ch ) {
    case '0' : pvib->integracion =  0.00 ;
    case '1' : if ( pvib->sensibilidad ==  0.10)
              pvib->integracion = 1 ;
            else pvib->integracion = 2 ;
    }
```

```
      break ;
   case 2 : /* sensor de aceleración */
      gotoxy_lcd ( 0,9 ) ;
      writestring ( "Integracion (0=No, 1=Acel.-Vel.) : "
                  ) ;
      do {
     gotoxy_lcd ( 36, 9 ) ;
     writestring (" ");
     gotoxy_lcd ( 36,9);
     ch = (char) gettec () ;
     }while ( ( ch != '0') && ( ch != '1' ));
     gotoxy_lcd ( 36, 9);
     writechar ( ch ) ;
      switch ( ch ) {
     case '0' : pvib->integracion = 0 ;
     case '1' : pvib->integracion = 3 ;
     }
      break ;
   }
gotoxy_lcd ( 0, 12 ) ;
writestring ( "Numero de Canales (1,2) : " ) ;
do {
   gotoxy_lcd ( 26, 12 ) ;
   writestring (" ");
   gotoxy_lcd ( 26,12);
   ch = (char) gettec () ;
   }while ( (ch != '1' ) && ( ch != '2'));
   gotoxy_lcd ( 26, 12 );
   writechar ( ch ) ;
   delay_us (800);
switch ( ch ) {
   case '1' : pvib->no_canales = 1 ;
     break ;
   case '2' : pvib->no_canales = 2 ;
     break ;
   }

/* Segunda pagina */

gotoxy_lcd ( 0, 0 ) ;
clean_lcd ( 3840 ) ;
gotoxy_lcd ( 7, 0 ) ;
writestring ( "PUNTO DE VIBRACION  Pg. 2" ) ;
gotoxy_lcd ( 0, 2 ) ;
writestring ( "Frecuencia de Muestreo " ) ;
gotoxy_lcd ( 2, 4 ) ;
writestring ( "( 0=500 Hz    1= 1 kHz    2= 2 kHz " ) ;
gotoxy_lcd ( 2, 6 ) ;
writestring ( "  3= 5 kHz    4=10 kHz    5=20 kHz" ) ;
gotoxy_lcd ( 2, 8 ) ;
writestring ( "  6=30 kHz    7=40 kHz    8=50 kHz " ) ;
gotoxy_lcd ( 2, 10 ) ;
```

```
writestring ( "              9=100 kHz ) :" ) ;

do {
  gotoxy_lcd ( 29, 10 ) ;
  writestring (" ");
  gotoxy_lcd ( 29,10 );
  ch= (char) gettec () ;
  )while (  ch < '0' || ch >'9');
  gotoxy_lcd ( 29, 10 );
  writechar ( ch ) ;
/* Decodificar los valores de la  frecuencia de muestreo
 */
pvib->fmuestreo = ch - '0' ;

gotoxy_lcd ( 0, 13 ) ;
writestring ( "No. de Muestras" ) ;
gotoxy_lcd ( 2, 15 ) ;
writestring ( " (0=512, 1=1024, 2=2048) : " ) ;
do {
  gotoxy_lcd ( 29, 15 ) ;
  writestring (" ");
  gotoxy_lcd ( 29,15);
  ch = (char) gettec () ;
  } while ( ( ch < '0' ) || ( ch > '2'));
  gotoxy_lcd ( 29, 15 );
  writechar ( ch ) ;
  delay_us (800);
pvib->no_muestras = ch - '0' ;

/* Tercera pagina */

set_char_size ( 12,40 );
gotoxy_lcd ( 0, 0 ) ;
clean_lcd ( 3840 ) ;
gotoxy_lcd ( 7, 0 ) ;
writestring ( "PUNTO DE VIBRACION  Pg. 3" ) ;
gotoxy_lcd ( 0, 2 ) ;
writestring ( "Funcion de Adquisicion  " ) ;
gotoxy_lcd ( 0, 4 ) ;
writestring ( "1= Vpp y Vrms," ) ;
gotoxy_lcd ( 0, 5 ) ;
writestring ( "2= Vpp, Vrms y Senal" ) ;
gotoxy_lcd ( 0, 6 ) ;
writestring ( "3= Vpp, Vrms y Espectro" ) ;
  ) ;
gotoxy_lcd ( 0, 11 ) ;
writestring ( "   Opcion : " ) ;
do {
  gotoxy_lcd ( 12, 11 ) ;
  writestring (" ");
  gotoxy_lcd ( 12,11 );
  ch = (char) gettec () ;
```

```
      } while ( ( ch < '1') || ( ch > '6') );
      gotoxy_lcd ( 12, 11);
      writechar ( ch ) ;
      delay_us (800);
    switch ( ch ) {
      case '1' : pvib->funcion_adq = 0 ;
        break ;
      case '2' : pvib->funcion_adq = 1 ;
        break ;
      case '3' : pvib->funcion_adq = 2 ;
        break ;
      }
     set_char_size ( 16, 40);

    /* Pagina cuatro */
    gotoxy_lcd ( 0, 0 ) ;
    clean_lcd ( 3840 ) ;
    gotoxy_lcd ( 5, 0 ) ;
    writestring ( "Punto de Vibracion  Pg. 4" ) ;
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "No. de Promedios : " ) ;
    do {
      gotoxy_lcd ( 19, 2 ) ;
      writestring ( "    " ) ;
      gotoxy_lcd ( 19, 2 ) ;
      pvib->no_promedios =  getint () ;
    } while  ( ( pvib->no_promedios < 0 )
              && ( pvib->no_promedios > 30 ) ) ;
    gotoxy_lcd ( 0, 4 ) ;
    writestring ( "Alerta (Valor Global) : " ) ;
    gotoxy_lcd ( 24, 4 ) ;
    pvib->alertapp =  getfloat () ;
    gotoxy_lcd ( 0, 6 ) ;
    writestring ( "Alarma (Valor Global) : " ) ;
    gotoxy_lcd ( 24, 6 ) ;
    pvib->alarmapp = getfloat () ;
    gotoxy_lcd ( 0, 8 ) ;
    writestring ( "Alerta (Valor RMS) : " ) ;
    gotoxy_lcd ( 21, 8 ) ;
    pvib->alertarms = getfloat () ;
    gotoxy_lcd ( 0, 10 ) ;
    writestring ( "Alarma (Valor RMS) : " ) ;
    gotoxy_lcd ( 21, 10 ) ;
    pvib->alarmarms = getfloat () ;
    delay_us ( 1000 ) ;
    }
  menu_principal () ;
}
```

```
b) void punto_fuera_ruta_temperatura (void)

/* Define un punto de temperatura no programado*/
{
  apuntador_temperatura ptemp ;
  apuntador_temperatura pt ;
  apuntador_vibracion pv ;
  apuntador_manual pm ;
  int tipo_registro ;
  float getfloat () ;
  char ch;
  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  ptemp  =  ( apuntador_temperatura)  _fmalloc ( (unsigned)
sizeof (punto_temperatura));
  if ( ptemp == NULL ) {
      gotoxy_lcd ( 0, 5 ) ;
      writestring (" Error : Memoria Insuficiente");
      sleep(3);
   menu_principal ();
      }
  else {
    if ( primer_punto == NULL ) {
      primer_punto = (apuntador_general)ptemp ;
      actual_punto = (apuntador_general)ptemp ;
      ptemp->siguientet = NULL ;
      }
    else {

      /* Enlazar el registro en la lista de puntos de la
         ruta */
      tipo_registro = (int) *actual_punto ;
      switch ( tipo_registro ) {
  case 0 : pv = (apuntador_vibracion) actual_punto ;
       ptemp->siguientet = pv->siguientev ;
       pv->siguientev = (apuntador_general) ptemp ;
       break ;
  case 1 : pt = (apuntador_temperatura) actual_punto ;
       ptemp->siguientet = pt->siguientet ;
       pt->siguientet = (apuntador_general) ptemp ;
       break ;
  case 2 : pm = (apuntador_manual) actual_punto ;
       ptemp->siguientet = pm->siguientem ;
       pm->siguientem = (apuntador_general) ptemp ;
       break ;
  }
      }
    ptemp->tipot = 1 ;
    ptemp->equipot [0] = '\0' ;
    ptemp->descripciont [0] = '\0' ;
```

```
    gotoxy_lcd ( 7, 0 ) ;
    writestring ( "PUNTO DE TEMPERATURA" ) ;
    /* Captura el identificador del punto */
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "Punto : " ) ;
    do {
      gotoxy_lcd ( 8, 2 ) ;
      writestring ( "     " ) ;
      gotoxy_lcd ( 8, 2 ) ;
      ptemp->identificadort = getint () ;
      } while ( ptemp->identificadort % 10 == 0 ) ;
    gotoxy_lcd ( 0, 4 ) ;
    writestring ( "Valor de Alerta : " ) ;
    gotoxy_lcd ( 18, 4 ) ;
    ptemp->alertat = getfloat () ;
    gotoxy_lcd ( 0, 6 ) ;
    writestring ( "Valor de Alarma : " ) ;
    gotoxy_lcd ( 18, 6 ) ;
    ptemp->alarmat = getfloat () ;
    delay_us (500);
    }
  menu_principal () ;
}
```

c) void punto_fuera_ruta_manual (void)

```
{ apuntador_manual pman;
  apuntador_manual pm ;
  apuntador_vibracion pv ;
  apuntador_temperatura pt ;
  int tipo_registro ;
  float getfloat () ;
  char ch;
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  pman  =  ( apuntador_manual) _fmalloc ( (unsigned) sizeof
(punto_manual));
  if ( pman == NULL ) {
      gotoxy_lcd ( 0, 5 ) ;
      writestring ("Error : Memoria Insuficiente");
      sleep(3);
      menu_principal();
      }
  else {
    if ( primer_punto == NULL ) {
      primer_punto = (apuntador_general) pman ;
      actual_punto = (apuntador_general)pman ;
      pman->siguientem = NULL ;
      }
```

```
    else {

      /* Enlazar  el registro  en la  lista de  puntos de la
ruta */
      tipo_registro = (int) *actual_punto ;
            switch ( tipo_registro ) {
  case 0 : pv = (apuntador_vibracion) actual_punto ;
        pman->siguientem = pv->siguientev ;
        pv->siguientev = (apuntador_general) pman ;
        break ;
  case 1 : pt = (apuntador_temperatura) actual_punto ;
        pman->siguientem = pt->siguientet ;
        pt->siguientet = (apuntador_general) pman ;
        break ;
  case 2 : pm = (apuntador_manual) actual_punto ;
        pman->siguientem = pm->siguientem ;
        pm->siguientem = (apuntador_general) pman ;
        break ;
  }
      }
    pman->tipom = 2 ;
    pman->equipom [0] = '\0' ;
    pman->descripcionm [0] = '\0' ;

    gotoxy_lcd ( 9, 0 ) ;
    writestring ( "PUNTO MANUAL" ) ;
    /* Captura el identificador del punto */
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "Punto : " ) ;
    do {
      gotoxy_lcd ( 8, 2 ) ;
      writestring ( "    " ) ;
      gotoxy_lcd ( 8, 2 ) ;
      pman->identificadorm = getint () ;
      } while ( pman->identificadorm % 10 == 0 ) ;
    gotoxy_lcd ( 0, 4 ) ;
    writestring ( "Alerta Inferior : " ) ;
    gotoxy_lcd ( 18, 4 ) ;
    pman->alertabajam = getfloat () ;
    gotoxy_lcd ( 0, 6 ) ;
    writestring ( "Alarma Inferior : " ) ;
    gotoxy_lcd ( 18, 6 ) ;
    pman->alarmabajam= getfloat () ;
    gotoxy_lcd ( 0, 8 ) ;
    writestring ( "Alerta Superior : " ) ;
    gotoxy_lcd ( 18, 8 ) ;
    pman->alertaaltam = getfloat () ;
    gotoxy_lcd ( 0, 10 ) ;
    writestring ( "Alarma Superior : " ) ;
    gotoxy_lcd ( 18, 10 ) ;
    pman->alarmaaltam = getfloat () ;
    delay_us (500);
```

```
    }
  menu_principal () ;
}
```

4.4.1.4 Gráficas

El acceso al menú de gráficas está cotrolado por la siguiente rutina:

```
void graficas (void)

/* Acceso al menú de gráficas */

{
  char tecla ;

  do {
    gotoxy_lcd ( 0, 0 ) ;
    clean_lcd ( 3840 ) ;  /* Limpiar la pantalla */
    gotoxy_lcd ( 5, 0 ) ;
    writestring ( "G R A F I C A S" ) ;
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "(1) Senal en el Tiempo" ) ;
    gotoxy_lcd ( 0, 4 ) ;
    writestring ( "(2) Espectro de la Senal" ) ;
    gotoxy_lcd ( 0, 6 ) ;
    writestring ( "(0) Regreso a Menu Principal" ) ;
    gotoxy_lcd ( 0, 8 ) ;
    writestring ( "    Opcion : " ) ;
    gotoxy_lcd ( 13, 8 ) ;
    tecla = gettec () ;
    gotoxy_lcd ( 13, 8 ) ;
    writechar ( tecla ) ;
    delay_us ( 100 ) ;
    if ( modo_operacion == 0 ) {
      switch ( tecla ) {
  case '1' : grafica_senal () ;
          break ;
  case '2' : grafica_espectro () ;
          break ;
  default : ;
  }
      }
    else {
      switch ( tecla ) {
  case '1' : grafica_senal_linea () ;
          break ;
  case '2' : grafica_espectro_linea () ;
          break ;
```

```
  default : ;
  }
     }
  } while ( tecla != '0' ) ;
  if ( modo_operacion == 0 ) menu_principal () ;
  else menu_linea () ;
}

a) void grafica_senal ( void )

/* Gráfica la señal en el  tiempo correspondiente  al último
punto capturado */

{
  register int i,  j ;
  char ch ;
  float deltat ;
  float t[2049 /*1025*/], y[2049 /*1025*/] ;
  float mint, maxt ;
  float smint, smaxt ;
  float miny, maxy ;
  int canal ;
  float fmuestreo;
  int no_muestras;
  boolean refresh ;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  /* Decodificar los valores de la frecuencia de muestreo */

  switch (ultimo_punto.fmuestreo ) {
    case 0 : fmuestreo = 500.0 ;
       break ;
    case 1 : fmuestreo = 1000.0 ;
       break ;
    case 2 : fmuestreo = 2000.0 ;
       break ;
    case 3 : fmuestreo = 5000.0 ;
       break ;
    case 4 : fmuestreo = 10000.0 ;
       break ;
    case 5 : fmuestreo = 20000.0 ;
       break ;
    case 6 : fmuestreo = 30000.0 ;
       break ;
    case 7 : fmuestreo = 40000.0 ;
       break ;
    case 8 : fmuestreo = 50000.0 ;
       break ;
    case 9 : fmuestreo = 100000.0 ;
       break;     }
```

```
/* Decodificar los numero de muestras */
switch (ultimo_punto.no_muestras ) {
  case 0 : no_muestras = 512 ;
     break ;
  case 1 : no_muestras = 1024 ;
     break ;
  case 2 : no_muestras = 2048 ;
     break ;
     }
/* Verificar el canal a graficar */
if (  ultimo_punto.no_canales == 2 ) {
  gotoxy_lcd ( 0, 2 ) ;
  writestring ( "Canal (1, 2) : " ) ;
  do {
    gotoxy_lcd ( 15, 2 ) ;
    ch = gettec () ;
    gotoxy_lcd ( 15, 2 ) ;
    writechar ( ch ) ;
    } while ( ( ch < '1' ) || ( ch > '2' ) ) ;
  canal = ch -'0' ;
  }


/* Calcular los arreglos de tiempos y de señal en unidades
    de ingenieria */

deltat = 1.0 / fmuestreo ;
if ( ultimo_punto.no_canales == 1 ) {
  for ( i=0 ; i < no_muestras ; i++ ) {
    t [i] = deltat * i ;
    y [i] = ( 0.0024414 * ultimo_punto.x[i] )
                          / ultimo_punto.sensibilidad ;
    }
  }
else {
  if ( canal == 1 ) {
    j = 0 ;
    for ( i=0 ; i < 2 * no_muestras  ; i++ ) {
if ( ( i % 2 ) == 0 ) {
  t [j] = deltat * j ;
  y [j] = ( 0.0024414 * ultimo_punto.x[i] )
                          / ultimo_punto.sensibilidad ;
  ++j ;
  }
}
    }
  else {
    j = 0 ;
    for ( i=0 ; i < 2 * no_muestras ; i++ ) {
if ( ( i % 2 ) != 0 ) {
  t [j] = deltat * j ;
  y [j] = ( 0.0024414 * ultimo_punto.x[i] )
```

```
                                / ultimo_punto.sensibilidad ;
  ++j ;
  }
}
    }
  }
/* Determinar los límites de la gráfica */
f_min_max ( no_muestras, t, &mint, &maxt ) ;
f_min_max ( no_muestras, y, &miny, &maxy ) ;
smint = mint ;
smaxt = maxt ;

/* Inicializa el LCD en modo gráfico */
init_graph () ;
refresh = TRUE ;
do {
  if ( refresh ) {
    /* Inicializar la ventana */
    win_init () ;

    /* Llamar a las escalas */
    scale_lin_x ( mint, maxt ) ;
    scale_lin_y ( miny, maxy ) ;

    /* Trazar los ejes */
    dr_lin_x_ax () ;
    dr_lin_y_ax () ;

    /* Dibuja las rejillas */
    dr_x_lin_grd () ;
    dr_y_lin_grd () ;

    /* Dibuja la curva */
    scatter_plot_data ( no_muestras, t, y, 1 ) ;

    /* Colocamos las escalas en los ejes */
    lab_lin_x_ax ( "%6.4f" ) ;
    lab_lin_y_ax () ;
    /* Colocamos los titulos en los ejes */
    title_x_ax ( "       Tiempo (s)" ) ;
    title_y_ax ( "mils" ) ;
    title_window ( "SERIE DE TIEMPO" ) ;
    /* Cursor grafico */
    /* outtextxy ( getmaxx()/2, getmaxy()-8, "<1> Zoom <2>
       Gráfica Normal <3> Cursor Gráfico <CR> Fin" ) ; */
    }
  refresh = FALSE ;
  ch = gettec () ;
  switch ( ch ) {
    case '1' : zoom_limits ( no_muestras, t, y, &mint,
                             &maxt, "T=%6.4f" ) ;
      refresh = TRUE ;
```

```
        break ;
      case '2' : mint = smint ;
        maxt = smaxt ;
        refresh = TRUE ;
        break ;
      case '3'  : cursor_grafico ( canal, no_muestras, t, y,
                          mint, maxt, "T=%6.4f A=%6.3f" );
        refresh = FALSE ;
        break ;
      }
  } while ( ch != 13 ) ;
  close_graph () ;
}
```

b) void gráfica_espectro (void)

```
/* Grafica el espectro de la  señal del  último punto captu-
rado */

{
  register int i,  j ;
  char ch ;
  float deltaf ;
  float f[1025], a[1025] ;
  float minf, maxf ;
  float miny, maxy ;
  float sminf, smaxf ;
  int canal ;
  float fmuestreo;
  int no_muestras;
  boolean refresh ;

  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  /* Decodificar los valores de la frecuencia de muestreo */

  switch (ultimo_punto.fmuestreo ) {
    case 0 : fmuestreo = 500.0 ;
       break ;
    case 1 : fmuestreo = 1000.0 ;
       break ;
    case 2 : fmuestreo = 2000.0 ;
       break ;
    case 3 : fmuestreo = 5000.0 ;
       break ;
    case 4 : fmuestreo = 10000.0 ;
       break ;
    case 5 : fmuestreo = 20000.0 ;
       break ;
    case 6 : fmuestreo = 30000.0 ;
       break ;
    case 7 : fmuestreo = 40000.0 ;
```

```
      break ;
    case 8 : fmuestreo = 50000.0 ;
      break ;
    case 9 : fmuestreo = 100000.0 ;
      break;
    }
  /* Decodificar los número de muestras */
  switch (ultimo_punto.no_muestras ) {
    case 0 : no_muestras = 256 ;
      break ;
    case 1 : no_muestras = 512 ;
      break ;
    case 2 : no_muestras = 1024 ;
      break ;
      }
  /* Verificar el canal a graficar */

  if (  ultimo_punto.no_canales == 2 ) {
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "Canal (1, 2) : " ) ;
    do {
      gotoxy_lcd ( 15, 2 ) ;
      ch = gettec () ;
      gotoxy_lcd ( 15, 2 ) ;
      writechar ( ch ) ;
      delay ( 1 ) ;
      } while ( ( ch < '1' ) || ( ch > '2' ) ) ;
    canal = ch - '0' ;

    if ( canal == 1 ) {
      for ( i = 0 ; i < no_muestras ; i++ )
  a[i] = ultimo_punto.ampy1[i] ;
      }
    else {
      for ( i = 0 ; i < no_muestras ; i++ )
  a[i] = ultimo_punto.ampy2[i] ;
      }
    }
  else {
    for ( i = 0 ; i < no_muestras ; i++ )
      a[i] = ultimo_punto.ampy1[i] ;
    }


  /* Calcular los arreglos de tiempos y de senal en unidades
de ingenieria */

  deltaf = 0.5 * fmuestreo / no_muestras ;
  for ( i = 0 ; i < no_muestras ; i++ ) f[i] = deltaf * i ;
  /* Llamar a las escalas */
  f_min_max ( no_muestras, f, &minf, &maxf ) ;
  f_min_max ( no_muestras, a, &miny, &maxy ) ;
```

```
sminf = minf ;
smaxf = maxf ;

/* Inicializa el modo gráfico del LCD */
init_graph () ;

refresh = TRUE ;
do {
  if ( refresh ) {
    /* Inicializar la ventana */
    win_init () ;

    scale_lin_x ( minf, maxf ) ;
    scale_lin_y ( miny, maxy ) ;

    /* Trazar los ejes */
    dr_lin_x_ax () ;
    dr_lin_y_ax () ;

    /* Dibuja las grids */
    dr_x_lin_grd () ;
    dr_y_lin_grd () ;

    /* Dibuja la curva */
    scatter_plot_data ( no_muestras, f, a, 1 ) ;

    /* Colocamos las escalas en los ejes */
    lab_lin_x_ax ( "%8.1f" ) ;
    lab_lin_y_ax () ;
    /* Titulos */
    title_x_ax ( "          F (Hz)" ) ;
    title_y_ax ( "mils 0-p" ) ;
    title_window ( "ESPECTRO LINEAL" ) ;
    }
  /* outtextxy  ( getmaxx()/2, getmaxy()-8, "<1> Zoom <2>
     Gráfica Normal <3> Cursor Gráfico <CR> Fin" )'; */
  refresh = FALSE ;
  ch = gettec () ;
  switch ( ch ) {
    case '1' : zoom_limits ( no_muestras, f, a, &minf,
          &maxf, "F=%7.1f" ) ;
      refresh = TRUE ;
      break ;
    case '2' : minf = sminf ;
      maxf = smaxf ;
      refresh = TRUE ;
      break ;
    case '3'  : cursor_grafico ( canal, no_muestras, f, a,
           minf, maxf, "F=%7.1f A=%6.4f" ) ;
      refresh = FALSE ;
      break ;
    }
```

```
  } while ( ch != 13 ) ;
  close_graph () ;
}

/* Incluye  el código de las funciones de operación en línea
*/
```

4.4.1.5 Definir Comentarios

```
void definir_comentarios (void)

/* Permite al usuario definir comentarios */

{  int i,j;
   int k;
   int tecla;
   boolean comdef ;
   int pagina ;
   int paginas ;
   int opcionmin ;
   int opcionmax ;
   char st[80] ;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  paginas = no_com_posibles / 10 ;
  if ( no_com_posibles % 10 > 0 ) ++paginas ;
  comdef = FALSE ;
  pagina = 1 ;
  do {
    gotoxy_lcd ( 0, 0 ) ;
    writestring ( "(0) Siguiente Pagina" ) ;
    /* Desplegar la pagina actual */
    if ( pagina != paginas ) {
      for ( j=0; j < 10 ; j++)
  {
  sprintf ( st, "(%2d) %s",10*(pagina-1)+j+1, coment[10  * (
pagina-1 ) + j] ) ;
  gotoxy_lcd ( 0, j+1);
  writestring ( st ) ;
  opcionmin = 10 * ( pagina - 1 ) + 1 ;
  opcionmax = 10 * ( pagina - 1 ) + 10 ;
  }
      }
    else {
      for ( j=0 ; j < ( no_com_posibles % 10 ) ; j++ ) {
  sprintf (  st, "(%2d) %s",10*(pagina-1)+j+1, coment[10 * (
pagina-1 ) + j] ) ;
```

```
  gotoxy_lcd ( 0, j+1);
  writestring ( st ) ;
  opcionmin = 10 * ( pagina - 1 ) + 1 ;
  opcionmax = no_com_posibles + 1 ;
  )
      }
    gotoxy_lcd ( 0, 12 ) ;
    writestring ( "Opcion : " ) ;
    do {
      gotoxy_lcd ( 9, 12 ) ;
      tecla = getint() ;
      if ( tecla == 0 ) break ;
    } while ( tecla < opcionmin || tecla > opcionmax ) ;
    if ( tecla == 0 ) {
      ++pagina ;
      if ( pagina > paginas ) pagina = 1 ;
      }
    else {
      coment_definidos [no_comentarios].indice_comentario =
      tecla - 1 ;
      no_comentarios++ ;
      comdef = TRUE ;
      }
    gotoxy_lcd ( 0, 0 ) ;
    clean_lcd ( 3840 ) ;
  } while ( !comdef ) ;
  menu_principal () ;

}
```

4.4.1.6 Programa de Comunicaciones

```
void programa_comunicaciones (void)

/* Activa el programa de comunicaciones */
{

  byte bcd;
  byte comando ;
  unsigned int status_port, status_puerto () ;
  int fin ;

  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  /* Inicializar la interfaz serie */

  inic_rs232 ( vel_trans_rs232, numero_bits_rs232,
    paridad_rs232, bit_paro_rs232 );
```

```
  /* Obtener el estado del puerto serie */

  status_port = status_puerto () ;

 verificar_estatus ( status_port ) ;
  fin = FALSE ;
  do {
    comando = leer_puerto () ;
    switch ( comando ) {
  case 1 :  solic_rec () ;
          break ;
  case 2 :  sol_fecha_hora () ;
          break;
  case 3 :  env_fecha_hora () ;
          break;
  case 4 :  modo_op () ;
          break;
  case 5 :  punto_vib ();
          break;
  case 6 :  punto_temp ();
          break;
  case 7 :  punto_man ();
          break;
  case 8 :  env_buf_vib ();
                  break;
  case 9 :  env_buf_temp_man ();
          break;
  case 10 : env_buf_com ();
          break;
  case 11 : sig_buf_vib ();
          break;
  case 12 : sig_buf_temp_man ();
          break;
  case 13 : sig_buf_com ();
          break;
  case 14 : buf_ant ();
          break;
  case 15 : canales_enlinea ();
          break;
  case 16 : solic_coment ();
          break;
  case 17 : fin_transmision () ;
          fin = TRUE ;
          break ;
  default : break ;
  }
    }
    while ( !fin ) ;
  menu_principal ();
}
```

4.4.2 Siguiente Punto

```
void sig_punto (void)

/* Despliega  los parametros  del siguiente punto en la ruta
*/

{
  apuntador_general p ;
  apuntador_vibracion pvib ;
  apuntador_temperatura ptemp ;
  apuntador_manual pman ;
  int tipo_registro ;

  tipo_registro = (int) *actual_punto ;
  switch ( tipo_registro ) {
    case 0 : pvib = ( apuntador_vibracion ) actual_punto ;
       p = pvib->siguientev ;
       break ;
    case 1 : ptemp = (  apuntador_temperatura ) actual_punto
;
       p = ptemp->siguientet ;
       break ;
    case 2 : pman = ( apuntador_manual ) actual_punto ;
       p = pman->siguientem ;
       break ;
    }
  if ( p != NULL ) {
    actual_punto = p ;
    tipo_registro = (int) *actual_punto ;
    switch ( tipo_registro ) {
      case 0 : pvib = ( apuntador_vibracion ) actual_punto ;

         desplegar_punto_vibracion ( pvib ) ;
         break ;
      case 1 : ptemp = ( apuntador_temperatura )
                       actual_punto ;
         desplegar_punto_temperatura ( ptemp ) ;
         break ;
      case 2 : pman = ( apuntador_manual ) actual_punto ;
         desplegar_punto_manual ( pman ) ;
         break ;
      }
    }
  else {
    /* Limpiar la pantalla */
    gotoxy_lcd ( 0, 0 ) ;
    clean_lcd ( 3840 ) ;
    gotoxy_lcd ( 5, 0 ) ;
    if ( ruta_cargada )
      writestring ( "Fin de la Ruta" ) ;
    else writestring ( "No hay Puntos cargados" ) ;
```

```
    sleep ( 4 ) ;
    menu_principal () ;
    }
}
```

4.4.2.1 Punto de Vibración

```
void desplegar_punto_vibracion (apuntador_vibracion pvib)

/* Despliega los parámetros de un punto de vibración */

{
  char st[80] ;
  char tecla;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 5, 0 ) ;
  writestring ( "PUNTO DE VIBRACION" ) ;
  gotoxy_lcd ( 0, 2 ) ;
  sprintf ( st, "Punto : %4d  Equipo : %s",
            pvib->identificadorv, pvib->equipov ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 3 ) ;
  writestring ( "Descripcion : " ) ;
  gotoxy_lcd ( 0, 4 ) ;
  sprintf ( st, " %s", pvib->descripcionv ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 5 ) ;
  switch ( pvib->tipo_sensor ) {
    case 0 : writestring ( "Tipo de Sensor : Desplazamiento"
                           ) ;
       break ;
    case 1 : writestring ( "Tipo de Sensor : Velocidad" ) ;
       break ;
    case 2  : writestring ( "Tipo de Sensor : Aceleracion" )
                          ;
       break ;
    }
  gotoxy_lcd ( 0, 6 ) ;
  switch ( pvib->integracion ) {
    case 0 : writestring ( "Integracion :  NO" ) ;
       break ;
    case 1 : writestring ( "Integracion : V-D" ) ;
       break ;
    case 2 : writestring ( "Integracion : V-D" ) ;
       break ;
    case 3 : writestring ( "Integracion : A-V" ) ;
       break ;
    }
```

```
  gotoxy_lcd ( 0, 7 ) ;
  writestring ( "Funcion de Adquisicion : " ) ;
  gotoxy_lcd ( 0, 8 ) ;
  switch ( pvib->funcion_adq ) {
    case 0  : writestring ( " Vpp, Vrms" ) ;
         break ;
    case 1  : writestring ( " Vpp, Vrms y Serie" ) ;
         break ;
    case 2  : writestring ( " Vpp, Vrms y Espectro" ) ;
         break ;
    case 8  : writestring ( " Vpp, Vrms, Ampl. y Fase" ) ;
         break ;
    case 9  : writestring ( " Vpp, Vrms, Serie, Ampl. y Fase"
                          ) ;
         break ;
    case 10  : writestring ( " Vpp, Vrms, Espectro, Ampl. y
                        Fase" ) ;
         break ;
    }
  gotoxy_lcd ( 0, 9 ) ;
  sprintf ( st, "Numero de Canales  : %ld", pvib->no_canales
) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 10 ) ;
  sprintf ( st, "Alerta (Vpp)  : %8.3f", pvib->alertapp ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 11 ) ;
  sprintf ( st, "Alarma (Vpp)  : %8.3f", pvib->alarmapp ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 12 ) ;
  sprintf ( st, "Alerta (Vrms) : %8.3f", pvib->alertarms ) ;

  writestring ( st ) ;
  gotoxy_lcd ( 0, 13 ) ;
  sprintf ( st, "Alarma (Vrms) : %8.3f", pvib->alarmarms ) ;

  writestring ( st ) ;
  gotoxy_lcd (0, 15);
  writestring (" <CR> Continuar...");
  do{
    tecla = gettec ();
    } while ( tecla != 13);
  menu_principal ();
}
```

4.4.2.2 Punto de Temperatura

```
void desplegar_punto_temperatura (apuntador_temperatura
     ptemp)

/* Despliega los parámetros de un punto de temperatura */

{
  char st[80] ;
  char tecla;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 5, 0 ) ;
  writestring ( "PUNTO DE TEMPERATURA" ) ;
  gotoxy_lcd ( 0, 2 ) ;
  sprintf ( st, "Punto : %4d  Equipo : %s",
                ptemp->identificadort, ptemp->equipot ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 4 ) ;
  writestring ( "Descripcion : " ) ;
  gotoxy_lcd ( 1, 5 ) ;
  writestring ( ptemp->descripciont ) ;
  gotoxy_lcd ( 0, 7 ) ;
  sprintf ( st, "Alerta : %8.3f", ptemp->alertat ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 9 ) ;
  sprintf ( st, "Alarma : %8.3f", ptemp->alarmat ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0,12);
  writestring ("<CR> Continuar...");
  do{
     tecla = gettec ();
     }while ( tecla != 13);
  menu_principal ();

}
```

4.4.2.3. Punto Manual

```
void desplegar_punto_manual (apuntador_manual pman)


/* Despliega los parametros de un punto manual */

{
  char st[80] ;
  char tecla;
```

```
  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 5, 0 ) ;
  writestring ( "PUNTO MANUAL" ) ;
  gotoxy_lcd ( 0, 2 ) ;
  sprintf ( st, "Punto : %4d  Equipo : %s",
              pman->identificadorm, pman->equipom ) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 4 ) ;
  writestring ( "Descripcion : " ) ;
  gotoxy_lcd ( 1, 5 ) ;
  writestring ( pman->descripcionm ) ;
  gotoxy_lcd ( 0, 7 ) ;
  sprintf ( st, "Alerta Inferior : %8.3f", pman->alertabajam
) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 9 ) ;
  sprintf ( st, "Alarma Inferior : %8.3f", pman->alarmabajam
) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 11 );
  sprintf ( st, "Alerta Superior : %8.3f", pman->alertaaltam
) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0, 13 ) ;
  sprintf ( st, "Alarma Superior : %8.3f", pman->alarmaaltam
) ;
  writestring ( st ) ;
  gotoxy_lcd ( 0,15);
  writestring ("<CR> Continuar...");
  do{
    tecla = gettec ();
    } while ( tecla != 13);
  menu_principal ();
}
```

4.4.3 Adquisición

```
void adquis (void)

/* Realiza la adquisición del siguiente punto */

{

  apuntador_general p;
  apuntador_vibracion pvib;
  apuntador_manual pman;
  apuntador_temperatura ptemp;
  int tipo_registro;
  tipo_registro = (int) *actual_punto;
```

```
    switch ( tipo_registro ){
      case 0 : pvib = (apuntador_vibracion) actual_punto;
         adquisv( pvib );
         break;
      case 1 : ptemp = (apuntador_temperatura) actual_punto;

         adquist ( ptemp );
         break;
      case 2 : pman = (apuntador_manual) actual_punto;
         adquism( pman );
         break;
       }

}
```

4.4.3.1 Adquisición de Punto de Vibración

```
void adquisv (apuntador_vibración pvib)

/* Realiza la adquisición de un punto de vibración */

{ float fmuestreo;
  int no_muestras;
  int i ;
  int j, k ;
  float  y[2049   /*1025*/],  y2[2049  /*1025*/],  cero[2049
/*1025*/];
  double sqrt() ;
  int *ptr;
  float *ptr1;
  char st [80] ;
  char tecla;

  /* Limpiar la variable cero */

  for ( i = 0 ; i < 2049 /* 1025 */ ; i++ ) cero[i] = 0.0 ;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  /* Asignar fecha y hora de adquisición */

  _dos_getdate ( &pvib->fecha_adq_vib ) ;
  _dos_gettime ( &pvib->hora_adq_vib ) ;

  /* Decodificar y actualizar los valores de la  frecuencia
     de muestreo */
  switch ( pvib->fmuestreo) {
    case 0 : fmuestreo = 500.0 ;
       break ;
```

```
      case 1 : fmuestreo = 1000.0 ;
         break ;
      case 2 : fmuestreo = 2000.0 ;
         break ;
      case 3 : fmuestreo = 5000.0 ;
         break ;
      case 4 : fmuestreo = 10000.0 ;
         break ;
      case 5 : fmuestreo = 20000.0 ;
         break ;
      case 6 : fmuestreo = 30000.0 ;
         break ;
      case 7 : fmuestreo = 40000.0 ;
         break ;
      case 8 : fmuestreo = 50000.0 ;
         break ;
      case 9 : fmuestreo = 100000.0 ;
         break;
      }
    /* Decodificar los número de muestras */
    switch (pvib->no_muestras ) {
      case 0 : no_muestras = 512 ;
         break ;
      case 1 : no_muestras = 1024 ;
         break ;
      case 2 : no_muestras = 2048 ;
         break ;
       }

    /* Actualizar variables del último punto */
    ultimo_punto.identificadorv = pvib->identificadorv ;
    ultimo_punto.sensibilidad = pvib->sensibilidad ;
    ultimo_punto.no_canales = pvib->no_canales ;
    ultimo_punto.fmuestreo = pvib->fmuestreo ;
    ultimo_punto.no_muestras = pvib->no_muestras ;
    ultimo_punto.funcion_adq = pvib->funcion_adq;

    /* Configurar el Hardware*/
    gotoxy_lcd ( 0, 2 ) ;
    writestring ( "Configurando el Hardware ..." ) ;
    cop (  fmuestreo );

    /* Programación de los integradores */
    outp ( 832, 16 ) ;  /* Descargar el capacitor de realimen-
                           tación */
    delay (1);
    switch (pvib->integracion){
      case 0 : outp ( 832, 17 ); /* Sin Integracion */
         break;
      case 1  : outp  ( 832,  66 );  /* Integracion vel-despl.
sensor 100 mv/pulg/seg */
         break;
```

```
    case 2  : outp  ( 832,  68 );  /* Integracion vel-despl.
sensor 500 mv/pulg/seg */
        break;
    case 3 : outp (  832,  72  );  /*  Integración acel-vel.
sensor 500 mv/G */
        break;
    }

  sleep (4);
  /* Adquirir la senal*/
  gotoxy_lcd ( 0, 3 ) ;
  writestring ( "Adquiriendo serie de Tiempo ..." ) ;
  senaltiempo ( no_muestras, ultimo_punto.no_canales,0,
                ultimo_punto.x ) ;

  /* Calcular vpp */
  gotoxy_lcd ( 0, 4 );
  writestring ( "Calculando el Valor pico a pico ..." ) ;
  vpp ( no_muestras, ultimo_punto.no_canales,
  ultimo_punto.sensibilidad,   ultimo_punto.x,  &pvib->vpp1,
&pvib->vpp2 );

  /* Calcular la serie de tiempo en formato real */
  senal_float ( ultimo_punto.no_canales, no_muestras,
       ultimo_punto.sensibilidad, ultimo_punto.x, y,y2);
  /* Regresa los valores de la senal en U.I. en y2 y y */

  /* Calcular el valor RMS */
  gotoxy_lcd ( 0, 5 );
  writestring ("Calculando el valor RMS...");
  /* Limpiar la variable cero */
  for ( i = 0 ; i < 2049 /* 1025 */ ; i++ ) cero[i] = 0.0 ;
  /* Calcular el espectro del primer canal*/
  gotoxy_lcd ( 0, 6 );
  writestring ( "Calculando el espectro..." ) ;
  fft ( y, cero, no_muestras) ;

  /* Calcular la amplitud y la fase del primer canal*/
  gotoxy_lcd ( 0, 7 ) ;
  writestring ( "Calculando amplitud y fase..." ) ;
  for ( i=0 ; i <= no_muestras / 2 ; i++ )
  ultimo_punto.ampy1[i] = (float) 2.0 * sqrt ( y[i] * y[i] +
cero[i] * cero[i] ) / no_muestras ;
  if ( ultimo_punto.no_canales == 2 ) {
    for ( i = 0 ; i < 2049 /*1025*/ ; i++ ) cero [i] = 0.0 ;


  /* Calcular el espectro del segundo canal*/
    gotoxy_lcd ( 0, 8 ) ;
    writestring ( "Calculando el espectro de la serie 2 ..."
) ;
    fft ( y2, cero, no_muestras ) ;
```

```
    for ( i = 0 ; i < no_muestras / 2 ; i++ ) {
      ultimo_punto.ampy2[i] =  (float) 2.0  * sqrt ( y2[i] *
y2[i] + cero[i] * cero[i] ) / no_muestras ;
    }
  }
  gotoxy_lcd ( 0, 9 ) ;
  writestring ( "Calculando el valor RMS ..." ) ;
  rms( no_muestras, ultimo_punto.ampy1, &pvib->vrms1 );
  if ( ultimo_punto.no_canales == 2 )
  rms ( no_muestras, ultimo_punto.ampy2, &pvib->vrms2 ) ;
  sleep (3);

  /* Regresa los valores RMS en &pvib->vrms1 y  pvib->vrms2
     */

  /* Desplegar los valores pico  a pico y RMS */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;
  gotoxy_lcd ( 0, 2  );
  writestring( "Vpp Canal 1 : ");
  gotoxy_lcd ( 14, 2 );
  sprintf ( st, "%7.2f", pvib->vpp1 ) ;
  writestring ( st ) ;
  if ( ultimo_punto.no_canales == 2 ) {
    gotoxy_lcd ( 0, 4 );
    writestring("Vpp Canal 2 : ");
    gotoxy_lcd ( 14, 4 );
    sprintf( st, "%7.2f",pvib->vpp2);
    writestring ( st ) ;
    }
  gotoxy_lcd ( 0, 6);
  writestring ( "Vrms Canal 1 : ");
  gotoxy_lcd ( 15, 6 );
  sprintf ( st, "%7.2f",pvib->vrms1 );
  writestring ( st ) ;
  if ( ultimo_punto.no_canales == 2) {
    gotoxy_lcd ( 0, 8 ) ;
    writestring( "Vrms Canal 2 : ");
    gotoxy_lcd ( 15, 8 );
    sprintf ( st, "%7.2f",pvib->vrms2);
    writestring ( st ) ;
    }

  /* Verificar los límites pico a pico de alerta y alarma */


  if (  pvib->alarmapp < pvib->vpp1 )
     { gotoxy_lcd ( 23, 2 );
       writestring(" Alarma");
     }
  else
    if ( pvib->alertapp < pvib->vpp1 )
```

```
      {  gotoxy_lcd ( 23, 2 );
writestring ("Alerta");
    }
  else { gotoxy_lcd ( 23, 2 );
 writestring("Normal");
 }
if ( ultimo_punto.no_canales == 2 ) {
  /* Segundo canal */
  if ( pvib->alarmapp < pvib->vpp2 )
    { gotoxy_lcd ( 23, 4 );
writestring ("Alarma");
    }
  else
    if ( pvib->alertapp < pvib->vpp2 )
{ gotoxy_lcd ( 23, 4 );
  writestring("Alerta");
 }
    else{ gotoxy_lcd ( 23, 4 );
    writestring ("Normal");
  }
  }
/* Verificar los valores RMS de alerta y alarma */
if (  pvib->alarmarms < pvib->vrms1 )
   { gotoxy_lcd ( 24, 6 );
     writestring(" Alarma");
   }
else
  if ( pvib->alertarms < pvib->vrms1 )
     {  gotoxy_lcd ( 24, 6 );
writestring("Alerta");
    }
  else { gotoxy_lcd ( 24, 6 );
 writestring("Normal");
 }
if ( ultimo_punto.no_canales == 2 ) {
  if ( pvib->alarmarms < pvib->vrms2 )
    { gotoxy_lcd ( 24, 8 );
writestring ("Alarma");
    }
  else
    if ( pvib->alertarms < pvib->vrms2 )
{ gotoxy_lcd ( 24, 8 );
  writestring("Alerta");
 }
    else{ gotoxy_lcd ( 24, 8 );
    writestring ("Normal");
  }
 }

  gotoxy_lcd ( 2, 12);
  writestring ("<CR> Continuar....");
```

```
 /*Almacenar la señal en el tiempo si la función de adquisi-
ción lo requiere*/
  if ( ultimo_punto.funcion_adq == 1 ||
                ultimo_punto.funcion_adq == 9){
     ptr = (int*) _fmalloc ((no_muestras *
                              ultimo_punto.no_canales) * 2);
     for (i=0; i <= (no_muestras * ultimo_punto.no_canales);
i++)
   *(ptr+i) = ultimo_punto.x [i];
     pvib->buffer_vib = (apuntador_general) ptr;
     }

 /*Almacenar  el  espectro  si  la función de adquisición lo
requiere*/
  if ( ultimo_punto.funcion_adq == 2 ||
               ultimo_punto.funcion_adq == 10){
     ptr1 =(float*) _fmalloc ((no_muestras / 2) * 4);
     for (i=0;  i <=  (no_muestras * ultimo_punto.no_canales
/2); i++)
   *(ptr1+i) = ultimo_punto.ampy1 [i];
     pvib->buffer_vib = (apuntador_general) ptr1;

     if (ultimo_punto.no_canales == 2){
     ptr1 =(float*) _fmalloc ((no_muestras / 2) * 4);
     for (i=0; i <= (no_muestras ); i++)
   *(ptr1+i+(no_muestras/2)) = ultimo_punto.ampy2 [i];
     pvib->buffer_vib = (apuntador_general) ptr1;
     }
   }
  if ( pvib->identificadorv % 10 == 0 )
                     ++no_punt_vib_medidos ;
  else ++no_punt_vib_noprogs ;
  do{
     tecla = gettec ();
     } while ( tecla != 13);
  menu_principal ();
}


a) void vpp (int no_muestras, byte no_canales, float
            sensibilidad, int  *buffer, float *vpp1, float
          *vpp2)

/* Calcula el valor pico a pico */

{
  int valor_pp;
  register  int i ;
  int min1, max1 ;
  int min2, max2 ;

   /* inicializamos valores min y max */
```

```
    min1 = 4095 ;
    max1 = 0 ;
    min2 = 4095 ;
    max2 = 0 ;

    /* Cálculo de los valores max y min (enteros) */

    if ( no_canales == 1 ) {
      for (i=0; i < no_muestras ; i++)
        {
        if ( min1 > buffer[i] )  min1 = buffer[i] ;
        if ( max1 < buffer[i] )  max1 = buffer[i] ;
        }
      valor_pp = max1 - min1 ;
      *vpp1 = ( 0.0024414 * valor_pp ) / sensibilidad ;
      *vpp2 = 0.0 ;
      }
    else {
      for ( i = 0 ; i < no_canales * no_muestras ; i++ ) {
        if ( i % 2 == 0 ) {
    /* Primer canal */
    if ( min1 > buffer[i] ) min1 = buffer[i] ;
    if ( max1 < buffer[i] ) max1 = buffer[i] ;
    }
        else {
    /* Segundo canal */
    if ( min2 > buffer[i] ) min2 = buffer [i] ;
    if ( max2 < buffer[i] ) max2 = buffer [i] ;
    }
        }
      valor_pp = max1 - min1 ;
      *vpp1 = ( 0.0024414 * valor_pp ) / sensibilidad ;
      valor_pp = max2 - min2 ;
      *vpp2 = ( 0.0024414 * valor_pp ) / sensibilidad ;
      }
}


b) void señal_float (byte no_canales, int no_muestras, float
                     sensibilidad, int  *buffer, float *y,
                   float *y2)

/* Calcula los valores de las  muestras de  una señal  en el
tiempo en formato de punto flotante */

{ register int i, j, k ;
  if ( no_canales == 1 )
    {
      for ( i=0 ; i < no_muestras ; i++ )
      y [i] = ( 0.0024414 * buffer[i] ) / sensibilidad ;
    }
  else {   /*primer canal */
```

```
    j = 0 ;
    k = 0 ;
    for ( i=0 ; i < ( no_muestras * no_canales ) ; i++ )
  { if  ( i % 2  == 0 ) {
      y [j] = ( 0.0024414 * buffer [i] ) / sensibilidad ;
      ++j ;
      }
    else { /* Segundo canal */
      y2[k] = ( 0.0024414 * buffer [i] ) / sensibilidad ;
      ++k ;
      }
   }
      }
}  /* senal_float */


c) void rms (int no_muestras, float *amps, float.*vrms)

/* Calcula el valor RMS de un espectro contenido en AMPS */

{
  register int i ;
  float sum ;
  sum = 0.0 ;
  for ( i = 0 ; i < no_muestras; i++ ) sum = sum + amps[i] *
amps[i] ;
  *vrms = sqrt ( sum / 2.0 ) ;
}


d) void señaltiempo (int n, byte nocanales, byte  nocanal,
                     int *buffer)

{
   register int i ;
   register int j ;
   register int k ;
   int nobloques ;
   byte stat ;

     outp ( 838, 0 ) ;
/* Inicializa el convertidor mientras captura*/

     if ( nocanales == 1 ){
/*habilitar un solo canal*/
       if( nocanal ==1 ) outp (835,0x2);
/*habilitar canal 0 */
       else outp (835,0x1);
/* habilitar el canal 1*/
      }
     else  outp  ( 835, 0x3 ) ;    /*habilitar 2 canales*/
```

```
/* Determinar el numero de bloques a leer */
     nobloques = ( n * nocanales ) /16 ;
/* Limpiar el buffer del convertidor Analógico a Digital */
     for ( j = 1 ; j <= 32 ; j++ ) {
       inp  (834);              /*parte baja de la lectura */
       inp  (832);              /*parte alta de la lectura*/
       }
     outp (838, 192 );          /*programa el convertidor
                                pra tomar 16 lecturas*/
/* Iniciar la conversión */
      _disable () ;
     for ( i = 1 ; i <= nobloques ; i++ ) {

/*  Esperar  la  interrupción  del  Convertidor  Analógico a
Digital */

       k =  16 * ( i - 1 ) ;
       do {
    } while ( ( inp ( 838 ) & 128 ) == 0 ) ;

       /* Leer siguiente bloque de 16 muestras */

       for ( j = 1 ; j <= 16 ; j++ ) {
   buffer [k+j-1] = (int) ( inp(834) + (  ( inp(832)  & 15 )
<< 8 )) - 2048 ;
   }
       }
      _enable () ;
      outp (835,0x0);
/* Inhabilitaal convertidor para leer*/
}
void swap ( float *s1, float *s2 ) ;


e) void  ampyfase ( float *xreal, float *yimag, int numdat,
                    float *amp, float *fase )


/* Calcula  la  amplitud  y  la  fase  de  un  espectro. Los
parámetros de entrada  son NUMDAT el número de puntos. XREAL
y YIMAG contienen la  parte real  y la  parte imaginaria del
espectro en  formato de punto flotante. AMP y  FASE regresan
la amplitud y la fase */

{
  register int i ;
  double sqrt(), atan() ;

  for ( i = 0 ; i <= numdat / 2 ; i++ ) {
     amp [i] = (float)  2.0 *  sqrt (  xreal[i] *  xreal[i] +
yimag[i] * yimag[i] ) / numdat ;
     fase [i] = (float) atan ( yimag[i] / xreal[i] ) ;
```

```
    )
}

f) void  frecuencias (float fmuestreo, int numdat, float
                      *float)


/* Determina el arreglo  de  frecuencias  para  un espectro.
FMUESTREO es la
   frecuencia de muestreo, NUMDAT es el numero de puntos
   que se  capturaron en la serie de tiempo y FREC contendra
el arreglo de
   frecuencias */

{
  register int i ;
  float deltaf ;

  deltaf = fmuestreo / numdat ;
  for ( i = 0 ; i <= numdat ; i++ ) frec[i] = deltaf * ( i -
1 ) ;
}

 void fft ( float *xreal, float *yimag, int numdat )

/* Calcula la transformada rapida de Fourier. Los parámetros
de entrada son
   NUMDAT el número de puntos y XREAL y  YIMAG tienen  en la
entrada  los  puntos  de  la  serie  de  tiempo  (en formato
flotante) y regresan con la parte real y la parte imaginaria
del espectro */

{
  int dc, maxpower, arg, q, cntr, pnt0, pnt1, i, j, a, b, k,
m ;
  float prodreal, prodimag, harm, x, y ;
  float cosary[MAXPUNTOS], sinary[MAXPUNTOS] ;
  double sin(), cos() ;

  j = 0 ;
  for ( i = 0 ; i <= numdat - 2 ; i++ ) {
    if ( i < j ) {
      swap ( &xreal[i], &xreal[j] ) ;
      swap ( &yimag[i], &yimag[j] ) ;
      }
    k = numdat / 2 ;
    while ( k <= j ) {
      j -= k ;
      k /= 2 ;
      }
    j += k ;
    }
```

```
  maxpower = 0 ;
  i = numdat ;
  while ( i != 1 ) {
    maxpower += 1 ;
    i /= 2 ;
    }
  harm = 6.2831853 / numdat;
  for ( i=0; i <= numdat - 1 ; i++ ) {
    sinary [i] = (float) sin ( harm * i ) ;
    cosary [i] = (float) cos ( harm * i ) ;
    }
  a = 2;
  b = 1;
  for ( cntr = 1 ; cntr <= maxpower ; cntr++ ) {
    pnt0 = numdat / a ;
    pnt1 = 0 ;
    for ( k = 0 ; k <= b-1 ; k++ ) {
      i = k ;
      while ( i < numdat ) {
  arg = i + b ;
  if ( k == 0 ) {
    prodreal = xreal [arg] ;
    prodimag = yimag [arg] ;
    }

  else {
    prodreal = xreal [arg]  * cosary  [pnt1] -  yimag[arg] *
sinary[pnt1] ;
    prodimag =  xreal [arg]  * sinary  [pnt1] + yimag[arg] *
cosary[pnt1] ;
    }
  xreal [arg] = xreal [i] - prodreal ;
  yimag [arg] = yimag [i] - prodimag ;
  xreal [i] = xreal [i] + prodreal ;
  yimag [i] = yimag [i] + prodimag ;
  i += a ;
  }
      pnt1 += pnt0 ;
      }
    a *= 2 ;
    b *= 2 ;
    }
}
```

4.4.3.2 Adquisición de punto de Temperatura

```
void adquist (apuntador_temperatura ptemp)

/* Adquiere un punto de temperatura */

{
 int j, i;
 byte bylo, byhi;
 float  mueslast, mues;
 int muestra ;
 float dif;
 double fabs () ;
 char ch;
 char st [80] ;
 char tecla;

  /* Limpiar la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  /* Asignar fecha y hora de adquisicion */

  _dos_getdate ( &ptemp->fecha_adq_temp ) ;
  _dos_gettime ( &ptemp->hora_adq_temp ) ;
  gotoxy_lcd ( 5, 0 );
  writestring ("Adquisicion de Temperatura");
  gotoxy_lcd ( 0, 2 );
  writestring ("Coloque el sensor de temperatura...");
  gotoxy_lcd ( 0, 4 );
  writestring ("Oprima la tecla ADQUIS" ) ;
  gotoxy_lcd ( 0, 5 ) ;
  writestring ( "para capturar el valor : ");

  /*Programar el convertidor de temperatura */
  outp ( 838,192 );

    /* Limpia el puerto*/
     for ( j = 1 ; j <= 32 ; j++ ) {
       inp  (834); /*parte alta de la lectura*/
       inp  (832); /*parte baja de la lectura*/
       }

       /* inicialización de variables para aproximación  */
  dif=0;
  mueslast = 0.0;

  do{
    ch = gettec ();
  } while ( ch != 'A');

  do{
```

```
    outp ( 838, 128);
/*Indica al convertidor que no debe guardar la lectura en el
buffer*/
    outp ( 0x350,0x02);
/*Da el  pulso de muestreo para tomar la lectura de tempera-
tura*/
    outp ( 0x350,0);
/*regresa al estado bajo el reloj del muestreo del converti-
dor para esperar otra señal de captura*/


    bylo = inp (834) ; /*parte baja de la lectura*/
    byhi = inp (832) ; /*parte alta de la lectura*/

    muestra   =   (int) (  ( (byhi  & 0x0f ) << 8)  + bylo -
2048 ) ;
    ptemp->valort = (float)  (muestra * 0.0024414 / 0.01) ;
    mues  = ptemp->valort;
    gotoxy_lcd ( 0, 7 );
    writestring ("Temperatura : ");
    gotoxy_lcd ( 14, 7 );
    sprintf ( st, "%7.2f", ptemp->valort ) ;
    writestring ( st ) ;
    dif = (float) fabs ( mues - mueslast ) ;
    mueslast = mues ;
    delay ( 500 ) ;
  } while ( dif > 0.5 ) ;
    /* Verificar los límites de alerta y alarma */

  gotoxy_lcd ( 22, 7 ) ;

  if (  ptemp->alarmat < ptemp->valort )
        writestring("Alarma");
  else
    if ( ptemp->alertat < ptemp->valort )
        writestring("Alerta");
    else writestring ("Normal");

  if ( ptemp->identificadort % 10 == 0 )
       ++no_punt_temp_medidos ;
  else ++no_punt_temp_noprogs ;
  gotoxy_lcd ( 0,10);
  writestring ("<CR> Continuar...");

  do {
      tecla = gettec ();
      } while ( tecla != 13 );
  menu_principal ();
}
```

4.4.3.3 Adquisición de punto manual

```
void adquism (apuntador_manual pman)

/* Adquiere un punto manual */


{
  /* Limpia la pantalla */
  gotoxy_lcd ( 0, 0 ) ;
  clean_lcd ( 3840 ) ;

  /* Asignar la fecha y hora de adquisición */

  _dos_getdate ( &pman->fecha_adq_man ) ;
  _dos_gettime ( &pman->hora_adq_man ) ;

  gotoxy_lcd ( 5, 0 );
  writestring ( "Punto de Captura Manual" ) ;
  gotoxy_lcd ( 0, 2 );
  writestring ("Valor : ");
  gotoxy_lcd ( 8, 2 );
  pman->valorm = getfloat ();
  if ( pman->identificadorm % 10 == 0 )
                ++no_punt_man_medidos ;
  else ++no_punt_man_noprogs ;
  menu_principal ();
}
```

La importancia de crear tecnología de origen nacional para sufragar los problemas propios de un país en vias de desarrollo, como lo es México, se hace patente en la lucha por la independencia económica; la importación de maquinaria y equipo sofisticado que facilite la industrialización de productos y servicios, se torna en una seria reducción en los recursos económicos de nuestro país.

La falta de herramientas, equipo de desarrollo e información, representan las principales limitantes para enfrentar los retos que encierra la creación de tecnología propia.

El Equipo de Adquisición de datos cuyo programa de control ha sido descrito en el presente trabajo de tesis, fué realizado en las instalaciones y bajo la supervisión del personal que labora en el Instituto de Investigaciones Eléctricas; centro de desarrollo de tecnología nacional.

Como se mencionó anteriormente, la falta de información técnica, específicamente en lo concerniente a los dispositivos electrónicos y paquetes de programacion constituyeron el principal obstáculo en la elaboración del proyecto.

La naturaleza del tema, al igual que todo desarrollo en materia de Electrónica Digital, demanda un adecuado manejo de las herramientas de ésta índole.

La programación de los dispositivos electrónicos requiere

así mismo de la información correspondiente por parte de los proveedores.

Tanto la tarjeta principal como los circuitos integrados fueron adquiridos en los Estados Unidos, la pantalla de cristal líquido (LCD) de la firma Hitachi INC., Japon, fué adquirida mediante su representante, en la ciudad de México. Lo árido de la información proporcionada y el tiempo de entrega de la misma, constituyeron factores importantes en la prolongación del período de pruebas con tales dispositivos.

Específicamente la programación del LCD y la tarjeta para desarrollo de prototipos, presentaron la problemática de tratarse de productos de reciente ingreso en el mercado. Las especificaciones y los datos técnicos que fueron proporcionados no se apegaban fielmente a la versión que se estaba recibiendo, de estos componentes.

El conjunto de instrucciones de programación para el LCD referia su texto a lo concerniente al controlador HD61830 de manera insuficiente. Una serie de iteraciones en los valores de la palabra de control para cada una de las instrucciones, se hizo necesaria para generalizar el funcionamiento de cada una de ellas.

El filtro programable requiere de un programa especialmente diseñado para definir los parámetros que determinan la naturaleza de la banda de filtrado que genera. Este programa es proporcionado por el fabricante, no así el algoritmo para

la obtención de dichos valores. Una tabla para los posibles argumentos de programación del dispositivo, según la operación del equipo, fue calculada e incluída en la codificación a partir del archivo del proveedor.

Por otra parte, cabe mencionar los aspectos relacionados con los obstáculos que se presentaron durante las pruebas con el convertidor analógico/digital y la forma como fueron superados.

El circuito AD1334 fué programado para realizar bloques de 16 conversiones; sin embargo, se advirtió que algunas muestras se estaban perdiendo. Este hecho repercutía directamente en el cálculo de los valores y la calidad de graficación.

El tiempo utilizado por el microprocesador de la tarjeta para efectuar las interrupciones de la memoria del sistema (ROMBIOS), que se realizan aún durante la ejecución del programa, es suficiente para afectar la captura de las muestras entregadas por el convertidor.

Mediante un adecuado uso de las instrucciones _disable y _enable dichas interrupciones fueron deshabilitadas durante éste proceso.

En lo relativo a la programación del circuito integrado transductor de temperatura AD594, no se presentaron problemas serios en su operación, no obstante, el tiempo de estabilización de la captura introdujo un error considerable en la lectura proporcionada. Este error fué reducido

sensiblemente mediante la repetición de la captura tantas veces como la diferencia entre la lectura anterior y la actual, fuera mayor a 0.5 grados Centígrados.

Quizá uno de los temas mas interesantes dentro de la Electrónica Digital es el concerniente a las comunicaciones. La comunicación serial entre dos computadores, es sin duda una herramienta invaluable para la transmisión de datos de manera versatil, sencilla y confiable.

El protocolo de comunicación es el medio por el cual se transmite la información, desde y hacia el equipo de adquisición de datos; los códigos se estandarizaron de acuerdo al precesamiento que se aplica a la información y los comandos determinan el tipo de datos y el momento propicio para transmitirlos.

La primera etapa de pruebas con el protocolo de comunicación se realizó mediante dos microcomputadoras; la primera de ellas de marca Televideo y la segunda, Olivetti.

Los resultados que se obtuvieron, no fueron satisfactorios debido a la manera particular del manejo de la bandera "Data Terminal Ready" (Terminal habilitada para Recibir) por parte de la computadora Olivetti, asignando un tiempo determinado para la recepción del dato, después del cual automaticamente se deshabilita dicha bandera. Debido a lo anteriór, el programa de control fue adaptado para mantener habilitada dicha bandera durante el tiempo en el cual el usuario establezca la comunicacion; tal efecto se logró mediante la

asignación del valor cero al puerto correspondiente (3fc h). La naturaleza de este trabajo de tesis, implicó el auxilio de una de las disciplinas mas fuertemente vinculadas con la Electrónica Digital y de Comunicaciones como lo es la Programación. Debido a la versatilidad del Lenguaje "C" para la codificación de sistemas de control de procesos, de comunicación y demás aplicaciones de ésta y otras índoles, se hizo necesario el conocimiento de dicho lenguaje.

El compilador para lenguaje "C" de programación que se utilizó para la codificacion, fué el paquete TURBO C de la firma Borland Inc., debido a la simplicidad que ofrece para el manejo de la lógica binaria y la comunicación serial.

Los problemas que se presentaron durante el proceso de compilación derivados de la dificultad de alterar el tamaño del "stack" llevaron a la decisión de utilizar el compilador "Quick C" de la firma Microsoft para efectuar dicha operación. Este último paquete contempla la posibilidad de modificar las condiciones de compilación de manera sencilla y externa al editor.

Los conocimientos adquiridos en la elaboración de este trabajo aunados al campo de aplicaciones que ofrece el Equipo Portatil de Adquisición de Datos, son el resultado del apoyo que gobierno federal otorga a las instituciones encaminadas a la investigación y el esfuerzo de mexicanos por lograr que en nuestro país, se constituyan las base para un desarrollo tecnológico propio.

B I B L I O G R A F I A

[1] Curso de Instrumentación, Análisis de Vibraciones y Métodos de Balanceo
J. Aguirre R. y E. Murphy A.
IIE
Agosto de 1985

[2] Microsoft Quick C Compiler Programmer's guide
Microsoft Corporation.

[3] The IBM Personal Computer from Inside out
Murray Sargent III
Richard L. Shoemaker
Addison-Wesley Publishing Company Inc.

[4] Introducción al estudio de las vibraciones mecánicas
Robert F. Steidel Jr.
CECSA

[5] Programmers problem's solver for the IBM PC
Robert Jourdain
Communications Company Inc.

[6] El lenguaje de programación C
Brian W. Kernighan
Dennis M. Ritchie
Bell Laboratories
Marray Hill, N.J.

[7] Ingenieria de Software
Fairlay
Mc Graw - Hill

[8] Estructura de Datos y diseño de Programas
Kruse
Prentice Hall

[9] Manual de Usuario del Sistema PORTATIL V2.0
Ing. Edmundo Ríos M., Ing. José Manuel Franco,
Ing. Juan José Rivera G.
Instituto de Investigaciones Eléctricas