ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería Eléctrica

"ADQUISICION DE DATOS Y GENERACION DE DIAGRAMAS DE TIEMPO PARA CIRCUITOS DIGITALES"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACION

Presentada por:

WELLINGTON CABRERA AREVALO

Guayaquil - Ecuador

1992

AGRADECIMIENTO

A Dios

A mis padres

A la Ing. Ludmila Gorenkova L.

Al Ing. Juan Sánchez H.

DEDICATORIA

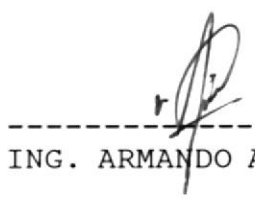
A mis padres

A mi esposa

A mi hijo

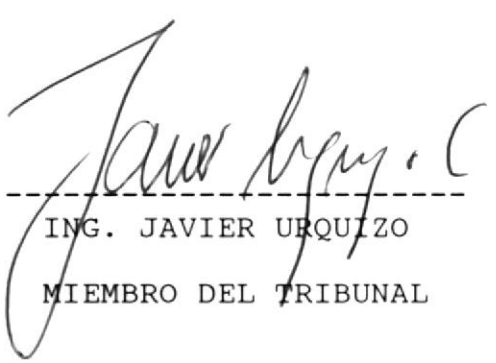
ING. LUDMILA GORENKOVA L.

DIRECTORA DE TESIS



ING. ARMANDO ALTAMIRANO

PRESIDENTE



ING. JAVIER URQUIZO

MIEMBRO DEL TRIBUNAL

ING. SIXTO GARCIA A.

MIEMBRO DEL TRIBUNAL

DECLARACION EXPRESA

"LA RESPOSABILIDAD POR LOS HECHOS, IDEAS Y DOCTRINAS EXPUETOS EN ESTA TESIS, ME CORRESPONDEN EXCLUSIVAMENTE, Y, EL PATRIMONIO INTELECTUAL DE LA MISMA, A LA ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos Profesionales de la ESPOL).

W Cabrera A

WELLINGTON MARCOS CABRERA AREVALO

RESUMEN.

En el capítulo 1 se expondrá básicamente la definición, objetivos y alcance del sistema; la lectura de este capítulo es necesaria para una mejor comprensión del propósito del proyecto; además se explicará las estrategias de solución y diseño para el mismo.

Los fundamentos teóricos para el diseño son tratados en el capitulo 2; de especial interés es la revisión que se realiza a la barra del PS2/30 o barra XT, donde se especifica la función de las señales de la barra; por otra parte se revisa también la teoría de los puertos de entrada y de la conversión analógica/digital.

Los capítulos 3,4,5 tratan del diseño del sistema. En el capitulo 3 se expone el diseño general del sistema, donde se especifica el comportamiento y las partes integrantes del mismo, asi también como la función de interfaz que efectúa el hardware y las funciones que realiza el software.

En el capítulo 4 se trata específicamente el diseño del hardware, que en sí es una interfaz para adquisición de datos en formato digital y analógico.

El diseño del software se describe en el capitulo 5; se especifica la interfaz con el usuario, los menúes y pantallas de especificación de parámetros, y el formato de salida por pantalla e impresora. En el diseño arquitectónico se presenta una vista general de los componentes del software, módulos y funciones. En el mismo capítulo, bajo el título de diseño detallado se hace una descripción de las funciones y procedimientos.

La documentación concerniente al Manual de Operación se encuentra en el capitulo 6. Se describe la instalación de la tarjeta de interfaz en el computador, el manejo del programa, el proceso de conección de un circuito digital a la interfaz construida, y como obtener los reportes y usar la pantalla de presentación de señales.

INDICE.

INTRODUCCION.

Un microcomputador puede ser utilizado como instrumento para captar señales eléctricas de un sistema digital, y brindar diferentes tipos de información partiendo de esos datos.

La presente Tesis de Grado trata de la definición, diseño e instrumentación de un sistema de computación, con componentes de hardware y software que permite leer, por medio de una interfaz construida para el efecto, señales eléctricas provenientes de un circuito digital, y presentarlas en forma de un diagrama de tiempo, tanto por pantalla como por impresora.

Con el fin de poder transferir al computador las señales del exterior, se desarrollará un circuito de adquisición de datos y un programa que representará los datos obtenidos por pantalla o por impresora. El desarrollo de este circuito es necesario por cuanto los puertos comunes no permiten efectuar un muestreo a la frecuencia necesaria para este tipo de aplicación y porque no se adaptan fácilmente al requerimiento de muestrear hasta 16 señales digitales y dos analógicas a la vez

CAPITULO I

DEFINICIÓN DEL SISTEMA DE GENERACIÓN DE DIAGRAMAS DE TIEMPO.

1.1 Definición.

En el desarrollo de sistemas digitales es siempre necesaria una herramienta que permita observar el comportamiento a través del tiempo de los circuitos que se están implementando. El sistema de Generación de Diagramas de Tiempo es una herramienta que a través de dispositivos de hardware y programas de computador permite obtener una representación visual de las señales de un circuito digital real.

El sistema a desarrollar permite obtener diagramas de tiempo para circuitos digitales por medio de un microcomputador , obteniendo las señales eléctricas de estos circuitos a través de puertos de entrada y debe permitir una fácil visualización de estas señales por la pantalla , facilitando el recorrido a través de la misma. Deberá también realizar la impresión de las señales escogidas por el usuario, en un intervalo de

tiempo definido por este. Se debe permitir también que el usuario pueda especificar la frecuencia y tiempo de muestreo.

Las señales eléctricas que se desean mostrar provienen de circuitos digitales, hasta 16 en formato digital y 2 en formato analógico. Para poder realizar la lectura de estas señales con el computador es necesario construir un circuito de interfaz , que consiste básicamente en puertos paralelos de entrada. Es necesario construir estos puertos para optimizar la velocidad de adquisición de datos; por otra parte, en cada muestreo la lectura de las 16 señales digitales y las 2 analógicas debe realizarse en forma simultanea. Este circuito será implementado en una tarjeta que será conectada en uno de las ranuras del computador .

En cada lectura, los datos son almacenados en la memoria RAM del computador, lo que permite que una vez terminado el muestreo, las señales puedan ser graficadas en la pantalla del computador, permitiendo al usuario un libre recorrido por todo el intervalo de tiempo muestreado.

En la graficación del diagrama por impresora, se permite definir un titulo para el diagrama, el

intervalo de tiempo a graficar y además escoger las señales que se desean mostrar, permitiendo la asignación de nombres para la presentación.

1.2 Objetivos.

Un objetivo importante es lograr que por medio de las facilidades de visualización y graficación de diagramas de tiempo, los estudiantes de los cursos de Sistemas Digitales comprendan que es de fundamental importancia el uso de los diagramas de tiempo en el diseño, pruebas y documentación de Sistemas Digitales.

Por ello, se buscará proveer de una herramienta que facilite las pruebas, depuración, y revisión de circuitos digitales, que permita observar el comportamiento de circuitos secuenciales a través del tiempo, por medio de la visualización de las señales provenientes de este circuito a través de la pantalla, y permitir la impresión de las señales en forma de diagramas de tiempo, que son necesarios para las pruebas y documentación de los circuitos digitales.

1.3 Ambiente de Operación.

El Sistema de Generación de Diagramas de Tiempo utilizará para su funcionamiento un computador IBM PS/2 Modelo 30, con microprocesador 8086, con 640 Kb de memoria y monitor MCGA. Este microcomputador posee tres ranuras de expansión (slots), con señales compatibles con la barra XT. El Sistema puede también operar en computadores compatibles con el IBM XT, con reloj de 8 MHz, que posean adaptador gráfico MCGA o VGA. El software correrá bajo el sistema operativo DOS . Se requiere además una impresora EPSON o compatible de 80 o 132 columnas.

1.4 Alcance.

Muestreo de hasta 18 señales, 16 en formato digital compatible con TTL o CMOS polarizado a +5V, y 2 en formato analógico. Las señales analógicas deberán estar entre los 0 y 5V.

La frecuencia de muestreo podrá ser especificada a través del programa, de hasta 30 KHz, en modo de 18 señales. La frecuencia de muestreo es controlada por software, y varía con valores discretos.

Hasta 64K de muestras, en cada uno de ellas llevando hasta 4 bytes de datos, requiriendo de un mínimo de 256 Kbytes de memoria RAM para el almacenamiento de los datos. Una muestra es la lectura de hasta 18 señales, adquiridas en forma simultánea

1.5 Estrategia de Solución.

La obtención de las señales por el computador se logra por medio de un circuito de adquisición de datos. Las señales digitales son obtenidas por este circuito a través de 2 puertos de entrada de 8 bits, que son implementados con buffers y flip-flops. Las 2 señales analógicas son convertidas a formato digital utilizando un convertidor analógico digital para cada señal, los que a su vez son vistos por el procesador como 2 puertos de entrada, por lo cual el Circuito de Adquisición de datos provee de un total de 4 puertos de entrada.

Por medio de una instrucción IN, la señal IOR.L se activa, y provoca el enclavamiento de las señales digitales en los flip-flops, mientras que a la vez provoca que arranque una nueva conversión en los convertidores analógico-digital. Posteriormente los datos de cada puerto son leídos secuencialmente y

almacenados en memoria byte a byte.

La frecuencia de muestreo es determinada por el usuario y controlada por Software. El numero de muestras es obtenido de la frecuencia y el tiempo de muestreo, que también puede ser especificado por el usuario.

CAPITULO II

FUNDAMENTOS TEÓRICOS.

2.1 La Barra del IBM PS/2 30

2.1.1 Generalidades.

Normalmente se conoce como "Barra" del microcomputador a las señales que se encuentran presentes en las ranuras o "slots" del microcomputador. El conjunto de estas señales es denominado por la documentación técnica de IBM Canal de Entrada/Salida (I/O Channel), y con este nombre lo designaremos de ahora en adelante.

El Canal de Entrada/Salida (E/S) es una extensión de la barra del microprocesador 8086, que es demultiplexado, reforzado y ampliado con la adición de líneas de interrupción y funciones de DMA (Direct Memory Access: Acceso Directo a Memoria).

El Canal de Entrada/Salida posee las siguientes señales:

- Una barra de datos bidireccional de 8 bits
- 20 líneas de direccionamiento
- Seis niveles de interrupción
- Líneas de control de lectura y escritura para memoria y E/S
- Señales de reloj
- Líneas de control para tres canales de DMA
- Línea de control de refrescamiento de memoria
- Una línea de chequeo de canal
- Líneas de poder y tierra

Cuatro niveles de voltaje son provistos para las tarjetas que se conecten al Canal de E/S:

+5 Vdc ±5%,
-5 Vdc ±10%
+12 Vdc ±5%
-12 Vdc ±10%

La línea de "I/O Channel ready" (Canal de E/S listo) está disponible en el Canal de E/S para permitir la operación con memorias o dispositivos de E/S lentos.

Los dispositivos de E/S son direccionados usando un espacio de direccionamiento de E/S mapeado. El canal esta diseñado para que mas de 64000 direcciones de dispositivos estén disponibles para tarjetas y adaptadores en el canal de E/S.

2.1.2 Distribución de señales en los Conectores.

El canal de E/S es reforzado en potencia para proveer suficiente corriente para los tres conectores de 62 pines que se encuentran disponibles, asumiendo dos cargas LS (Low Power Schottky) por cada ranura. La Figura 2.1 muestra como se encuentran distribuidas las señales en las ranuras del IBM PS/2 30.

2.1.3 Descripción de las señales.

Las señales rotuladas con "S" son salidas del canal, y las rotuladas con "E" son entradas al canal.

A0 - A19 (S): Bits de dirección desde 0 hasta 19. Estas líneas son usadas para direccionar memoria o dispositivos de entrada o salida. A0 es la señal de direccionamiento menos significativa

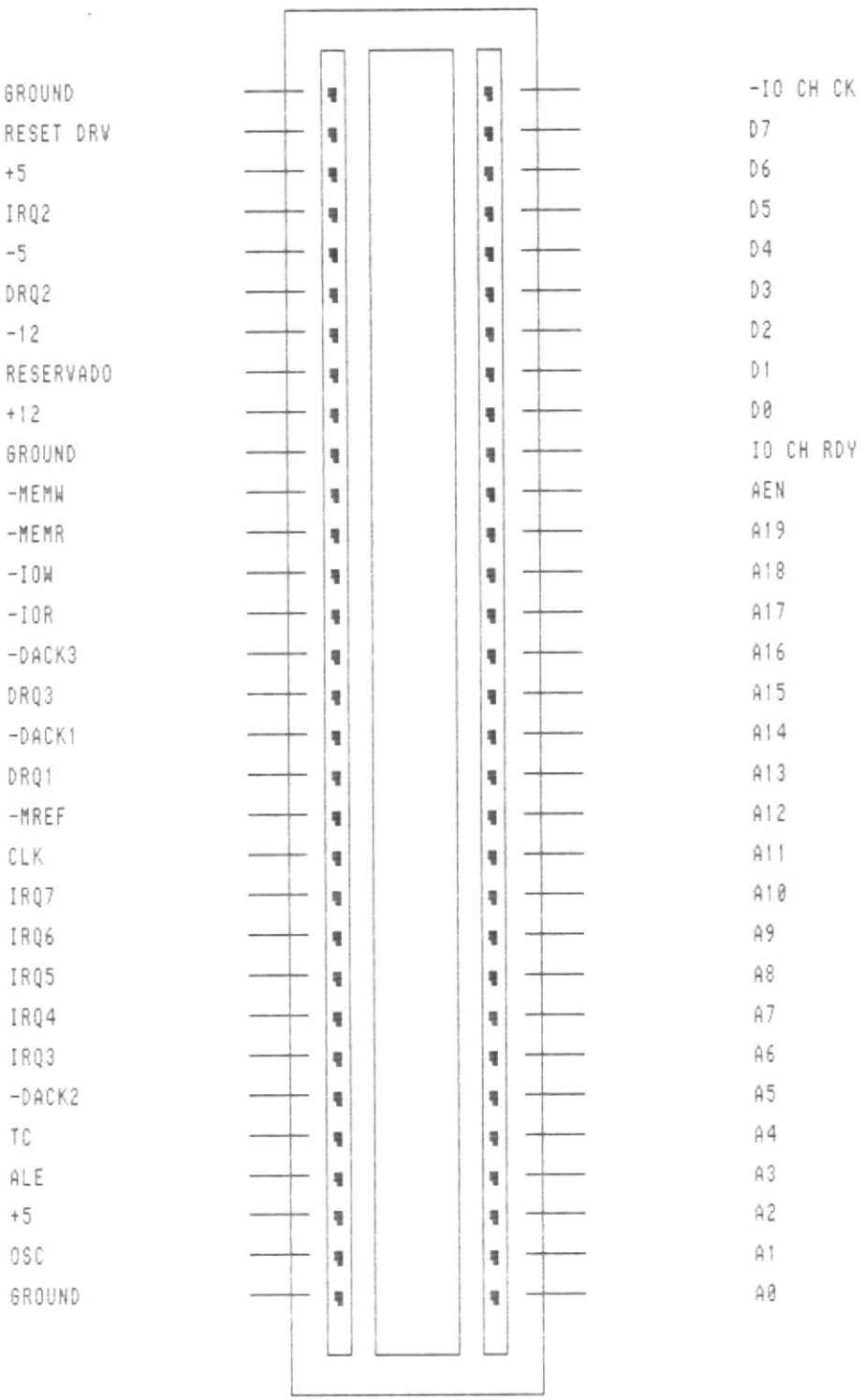


Figura 2.1

y A19 es la más significativa.Las 20 líneas de direccionamiento permite acceder hasta 1 Mb de memoria. Solo las líneas A0 hasta A15 son usadas para el direccionamiento de E/S y todas 16 deberían ser decodificadas por los dispositivos de E/S. Las señales en estas líneas son generadas por el microprocesador y el controlador de DMA.

AEN (S): Habilitación de Dirección (Address Enable). Esta línea es usada para desvincular el microprocesador y otros dispositivos del canal de E/S, para permitir que tomen lugar las transferencias de DMA. Cuando esta señal está activa, el controlador de DMA toma control del barra de direcciones, barra de datos y las señales de control de escritura y lectura; cuando está inactiva, es el microprocesador el que toma el control.

ALE (S): Habilitación de enclavamiento de dirección (Address Latch Enable). Esta señal es provista por el controlador de la barra y es usada en la tarjeta principal del computador para enclavar direcciones válidas para el microprocesador. Las direcciones son válidas en

el flanco descendente de ALE y son enclavadas en la barra mientras ALE está inactiva. Esta señal se encuentra activa durante los ciclos de DMA.

CLK (S): Reloj del Sistema. Esta línea proporciona la señal del reloj del sistema, con una frecuencia de 8 MHz y un ciclo de servicio del 33%.

D0-D7 (E/S): Corresponde a 8 líneas de datos . Estas líneas proveen los 8 bits de la barra de datos, para el microprocesador, memoria, y dispositivos de E/S.

DACK1.L - DACK3.L (S): Señales de reconocimiento de DMA 1 al 3. Estas líneas son usadas por el controlador para el reconocimiento de requerimientos de DMA. La señal DACK0 no se encuentra disponible en el Canal de E/S del Modelo 30.

DRQ1 - DRQ3 (E): Entradas de Requerimiento de DMA del 1 al 3. Estas líneas son requerimientos asincrónicos del canal.

IO CH CK.L : Revisión de Canal de E/S (I/O Chanel Check). Genera una Interrupción no Enmascarable (NMI). Es activada para indicar un error incorregible, y mantenida activa por lo menos por dos ciclos de reloj.

IO CH RDY (E): Canal de E/S Listo (I/O Channel Ready). Esta línea, normalmente activa, es puesta inactiva por una memoria o dispositivo de E/S con el fin de alargar los ciclos de memoria o de E/S, respectivamente. Esto permite a dispositivos más lentos acoplarse al canal de E/S con un mínimo de dificultad. Cualquier dispositivo lento debería desactivar esta señal inmediatamente después de detectar una dirección válida y un comando de lectura o escritura. Para cada ciclo de reloj en que esta señal es mantenida inactiva, es agregado un estado de espera. Esta línea no debería permanecer inactiva por más de 17 ciclos de reloj.

IOR.L (S): Lectura de E/S. Esta señal es una salida que indica a un dispositivo de E/S que debe ubicar sus datos en la barra de datos. Esta señal es manejada por el microprocesador o el controlador de DMA.

IOW.L (S): Escritura en E/S. Esta señal de salida indica a un dispositivo de E/S que debe tomar los datos de la barra de datos. Esta señal es manejada por el microprocesador o el controlador de DMA.

IRQ2 - IRQ7 (E): Líneas de Requerimiento de interrupción de 2 a 7. Son usadas para señalar al microprocesador que un dispositivo de E/S requiere atención. IRQ2 tiene la más alta prioridad, IRQ7 la más baja. Cuando una interrupción es generada, la línea de requerimiento de interrupción es mantenida activa, hasta que sea reconocida por el microprocesador.

MEMR.L (S): Esta señal indica a la memoria que debe ubicar sus datos en la barra de datos. Esta señal es manejada por el microprocesador o el controlador de DMA.

MEMW.L (S): Esta señal indica a la memoria que debe almacenar los datos en la barra de datos. Esta señal es manejada por el microprocesador o el controlador de DMA.

MREF.L (E/S): Refrescamiento de memoria. Esta señal indica un ciclo de refrescamiento.

OSC. (S): Señal de oscilador. Es una señal de reloj con una frecuencia de 14.3 MHz. Tiene un ciclo de trabajo del 50%.

RESET DRV(S): Esta línea es usada para inicializar la lógica del sistema en el momento del encendido o durante una baja de voltaje. Esta señal es sincronizada con el flanco descendente de CLK.

TC (S): Final de Conteo (Terminal Count). Esta línea provee un pulso cuando el conteo final para cualquier canal de DMA es alcanzado.

2.2 Puertos de Entrada.

En general, un puerto de entrada consiste en un circuito digital que, al ser direccionado por el microprocesador en una instrucción de entrada pone los datos requeridos a disposición del microprocesador en la barra de datos. El ciclo de instrucción para una instrucción de entrada-salida en el 8086 se presenta en la figura 2.2a. En una instrucción de entrada, el

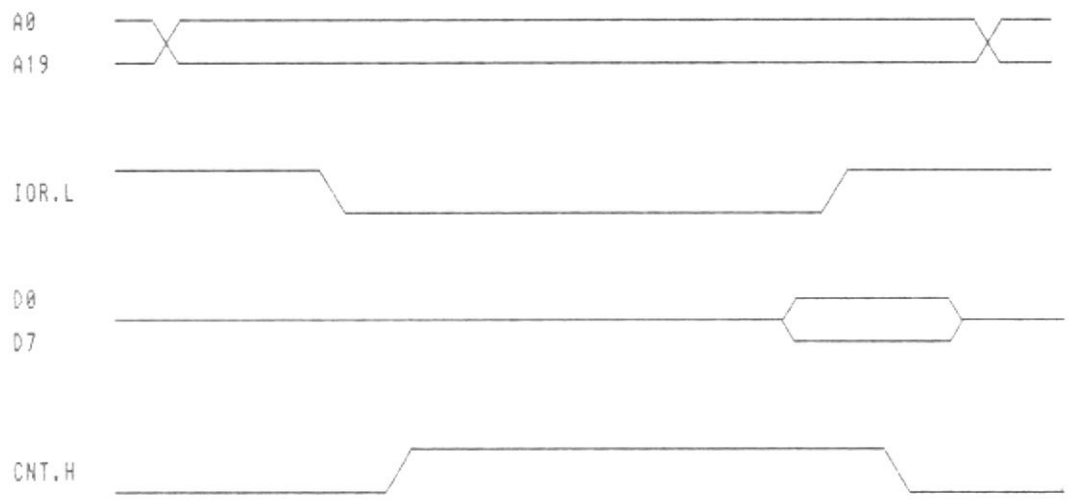


Figura 2.2 a

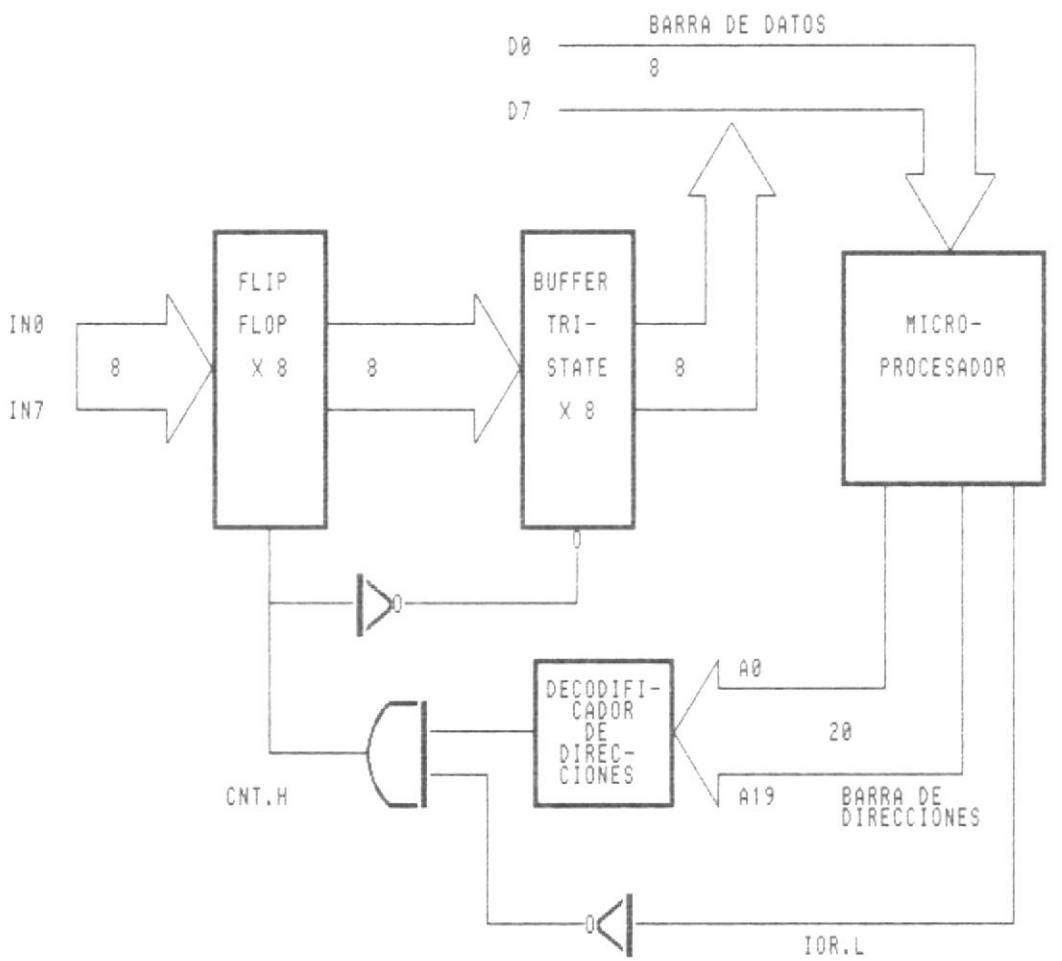


Figura 2.2 b

microprocesador toma los 8 bits que se encuentran en la barra de datos.

En la figura 2.2b se presenta un puerto de entrada simple. Como se puede observar, consiste básicamente en un banco de flip-flops tipo D, buffers de triple estado, un circuito de decodificación de dirección y la lógica de selección.

Primeramente, el microprocesador pone la dirección del dispositivo en la barra de direcciones. La señal decodificada se presenta a la salida del decodificador de dirección. Al tomar la señal IOR.L un nivel bajo, se produce el enclavamiento de las señales, a la vez que es habilitado el buffer. A partir de este momento, las señales están disponibles en la barra de datos para ser leídas por el microprocesador.

2.3 Conversión Analógica-Digital.

2.3.1 Procedimiento general para la conversión analógica-digital.

El convertidor analógico-digital es un dispositivo que toma un voltaje analógico a su entrada, y después de un cierto tiempo produce

a la salida un código digital, el cual representa el valor de la señal de la entrada analógica.

En general, el proceso para convertir una señal analógica a formato digital requiere una secuencia de cuatro procesos individuales , que son: muestreo, mantenimiento, cuantificación y codificación.

Estos procesos no son necesariamente efectuados como operaciones separadas. En la práctica, el muestreo y el mantenimiento se hacen simultáneamente en un tipo de circuito denominado Circuito de Muestreo y Mantenimiento (Sample and Hold), mientras la cuantificación y la codificación se hacen simultáneamente en el convertidor A/D.

En la figura 2.3b se presenta una señal que debe ser muestreada en dos instantes, separados por el intervalo de muestreo Ts. Un circuito básico de Muestreo y Mantenimiento se muestra en la figura 2.3a. El muestreo está temporizado por la señal de control Vs, que cierra y abre el interruptor electrónico SW. Durante el tiempo Tc el interruptor está cerrado y el condensador se

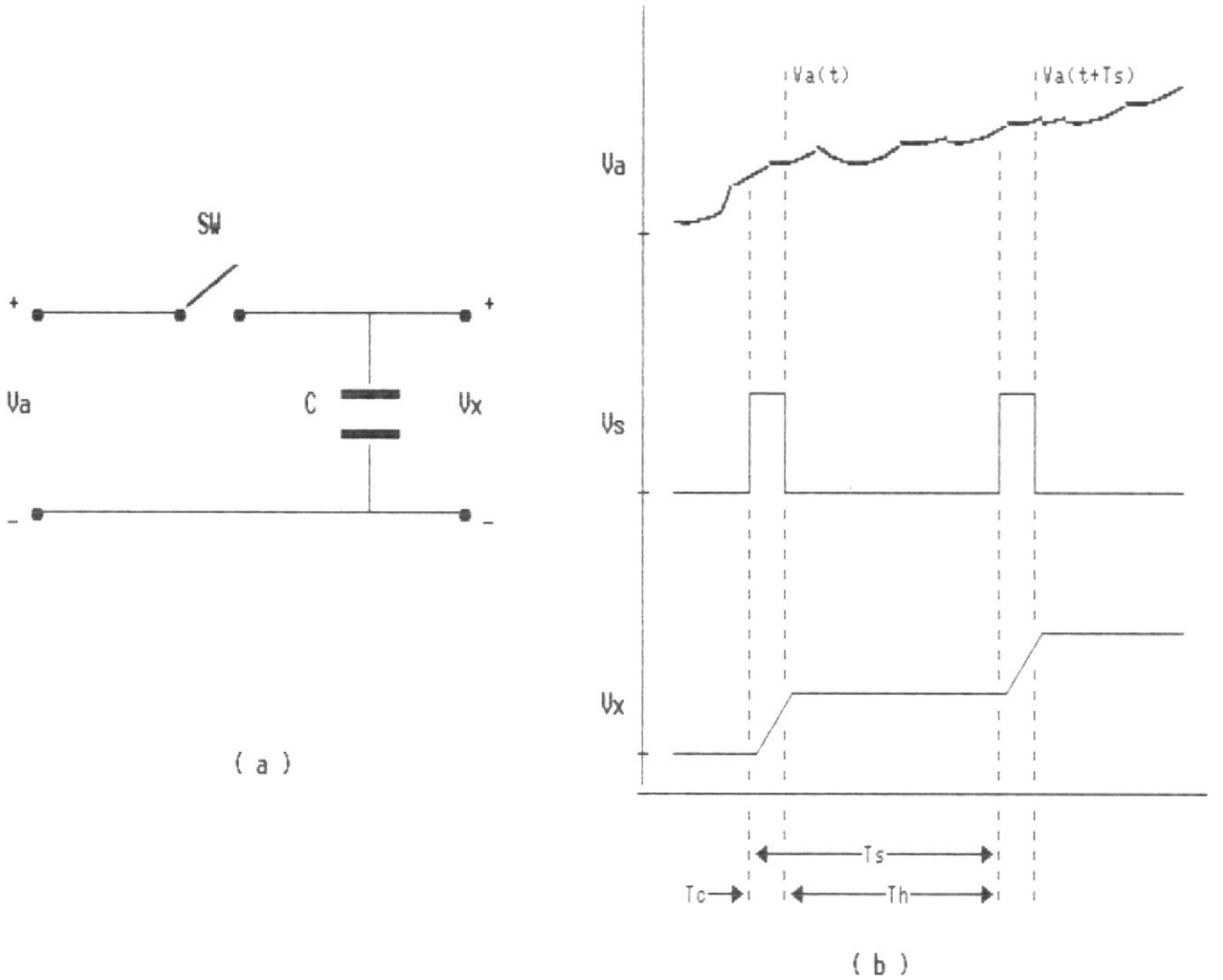


Figura 2.3

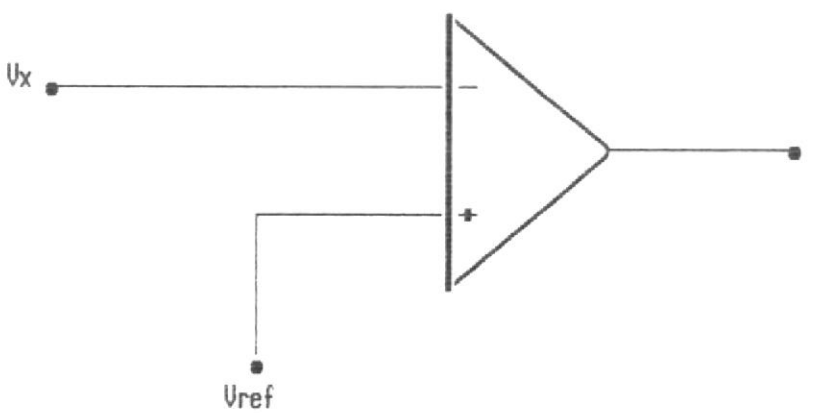


Figura 2.4

carga hasta Va.

En el tiempo restante Th = Ts -Tc el valor de la muestra se mantiene en el condensador. La forma de onda de salida del circuito de Muestreo y Mantenimiento está representada en la figura 2.3 b

El circuito de Muestreo y Mantenimiento permite tomar lectura de la señal analógica no en todos los instantes, si no solamente en los tiempos de muestreo; por lo tanto en el intervalo entre dos muestreos tenemos tiempo de convertir cada tensión de muestreo a formato digital.

El circuito básico de un convertidor analógico-digital se presenta en la figura 2.4. El voltaje desconocido de entrada Vx está conectado a una de las entradas del comparador, y un voltaje de referencia Vref está conectado a la otra entrada. Si el voltaje de entrada Vref excede el voltaje de entrada Vx, entonces el voltaje de salida será alto. Si el voltaje de entrada Vx es más grande que Vref, entonces el voltaje de salida será bajo.

La diferencia básica entre el funcionamiento de varios tipos de convertidores A/D es la estrategia que se utiliza para variar la señal de referencia Vref.

2.3.2 Convertidor A/D de doble rampa.

El diagrama de bloques del convertidor A/D de doble rampa puede observarse en la figura 2.5. El ciclo de conversión consiste en dos intervalos de integración separados. Primero, el voltaje desconocido Vx es integrado en un período de tiempo conocido T1. El valor de esta integral es comparado con el resultado de integrar un voltaje de referencia conocido durante un tiempo variable T2.

Cuando comienza el proceso de conversión, en t = 0, el interruptor S1 está conectado a la entrada Vx y la muestra mantenida en la salida del circuito S/H es aplicada al circuito integrador. La salida del integrador es:

$$Vo = -\frac{1}{RC}\int_0^t Vx\,dt + Vc(0) \approx -Vx\frac{t}{\tau}$$

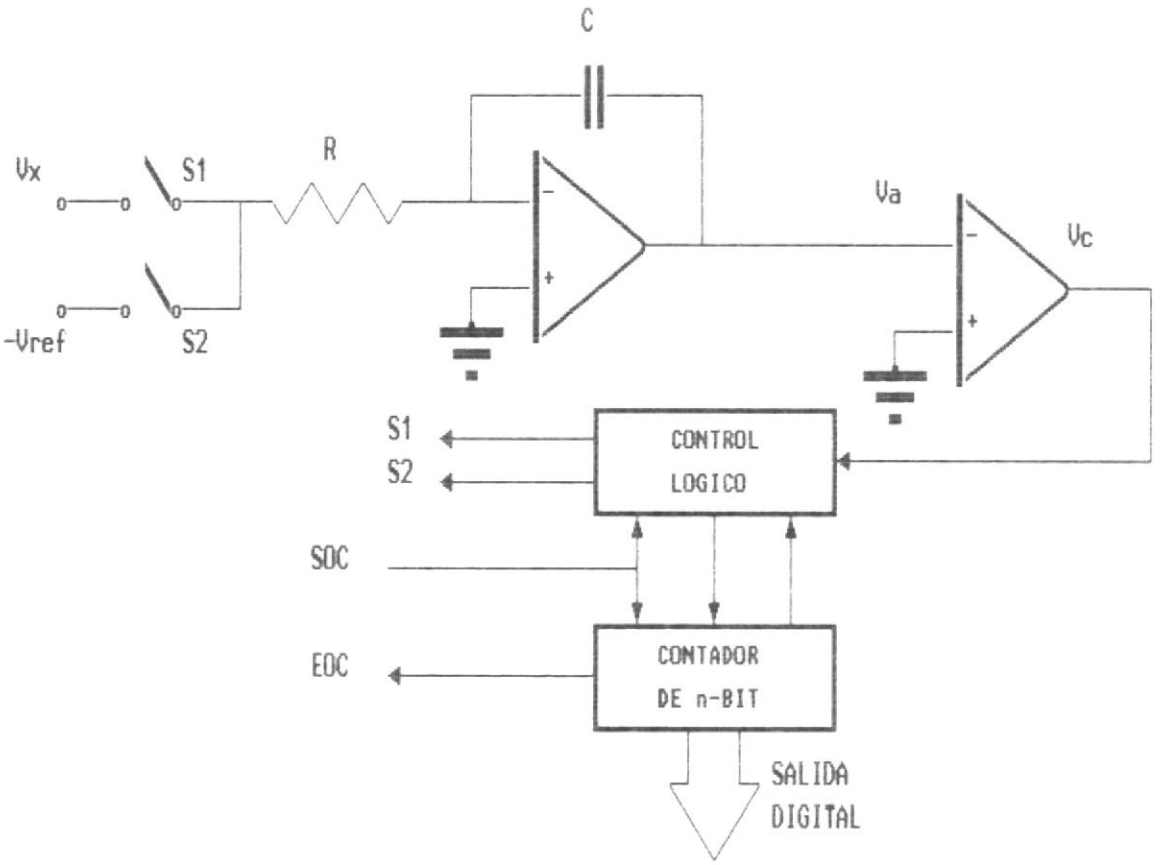


Figura 2.5

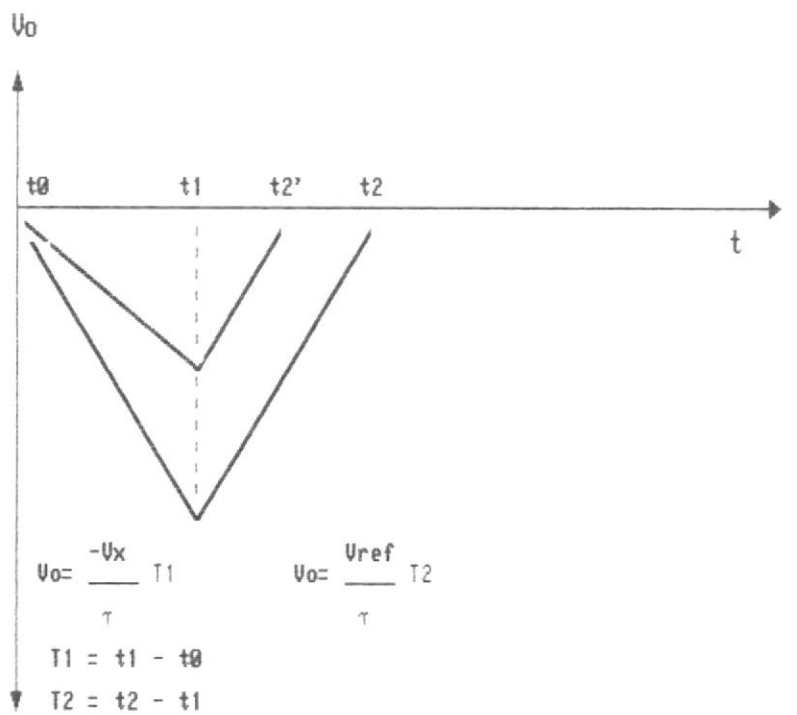


Figura 2.6

La forma de onda de Vo está presentada en la figura 2.6. En el mismo instante (t=t0) el contador empieza a contar hasta un número predeterminado (t=t1). En este momento, el circuito controlador cambia las posiciones de S1 y S2. Es decir, en la entrada del circuito integrador se conecta la señal -Vref. También en este mismo instante el circuito controlador inicializa el contador. La señal de salida del integrador empieza ahora a desplazarse en otro sentido, ya que el voltaje de referencia es negativo. El contador continua contando hasta que el voltaje de salida Vo se hace positivo.

En este instante (t=t2) la salida del comparador pasa al estado 0 y el circuito controlador hace detener el contador. La cuenta registrada a la salida del contador en este momento t2 es proporcional al voltaje desconocido Vx. El convertidor A/D de doble rampa simplemente convierte el voltaje analógico en un período de tiempo que es medido por el contador. Se debe tener presente que para que el convertidor de doble rampa trabaje correctamente, el voltaje analógico medido debe ser siempre menor que el voltaje de referencia.

2.3.3 Convertidor A/D de Cuenta Ascendente.

En la figura 2.7 se encuentra la representación en diagrama de bloques del convertidor A/D de cuenta descendente. La conversión Analógica-Digital comienza cuando un pulso de Reset vuelve a poner la salida del contador a cero. Cada pulso de reloj sucesivo incrementa la salida del contador hasta que la salida del DAC excede la entrada desconocida Vx. En este momento, la salida del comparador cambia de estado e impide que el contador siga contando. El cambio de estado a la salida del comparador indica que la conversión es completa, y en este momento el contenido del contador binario representa el valor convertido de la señal Vx.

El tiempo de conversión es variable y proporcional al voltaje desconocido Vx. El tiempo máximo de conversión para el convertidor de cuenta ascendente ocurre para una señal de entrada de plena escala y corresponde a 2n períodos del reloj.

La ventaja del convertidor A/D de cuenta ascendente (o contador-rampa) es que el requiere

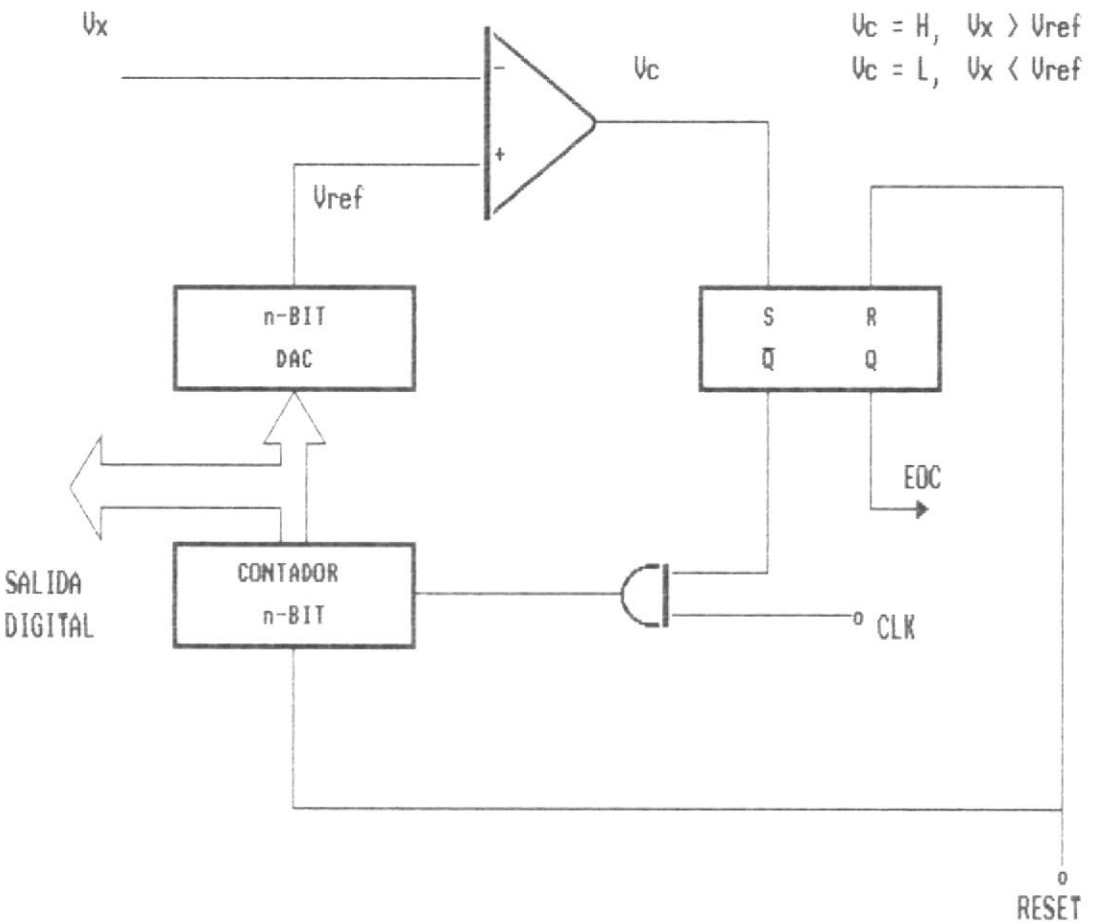

Vx
Vc = H, Vx > Vref
Vc = L, Vx < Vref
Vc
Vref
n-BIT
DAC
S
R
Q
Q
EOC
SALIDA
DIGITAL
CONTADOR
n-BIT
CLK
RESET

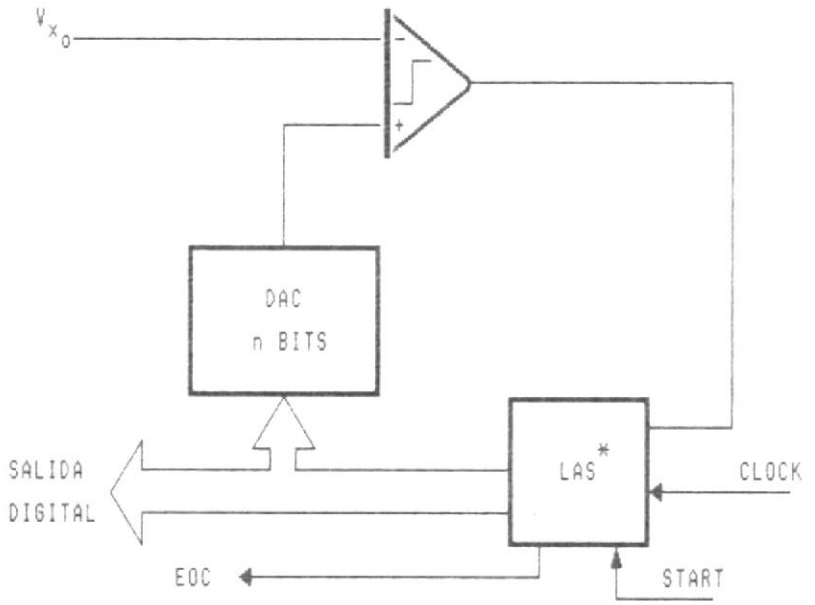
Figura 2.7

una mínima cantidad de hardware, es barato de implementar y más rápido que el de doble rampa.

2.3.4 Convertidor A/D de Aproximaciones Sucesivas.

El convertidor de aproximaciones sucesivas utiliza una técnica muy eficiente para variar la entrada de referencia del comparador, logrando una conversión que requiere solo n períodos de reloj para completar una conversión de n-bit.

Un diagrama esquemático de un convertidor de aproximaciones sucesivas de tres bits es mostrada en la figura 2.8 . Una búsqueda binaria es utilizada para encontrar la mejor aproximación a Vx. Después del reset, la lógica de aproximaciones sucesivas provoca una salida en el convertidor digital analógico de $7V_{FS}/16$ y después chequea la salida del comparador (La salida del DAC es desplazada en -0.5 LSB o $-V_{FS}/16$). En el siguiente pulso de reloj, la salida del DAC es incrementada en $V_{FS}/4$ si la salida del comparador fue 2, y decrementada por $V_{FS}/4$ si la salida del comparador fue 0. La salida del comparador es examinada nuevamente, y el próximo pulso de reloj provoca que la salida del DAC sea



*LOGICA DE APROXIMACIONES SUCESIVAS

Figura 2.8

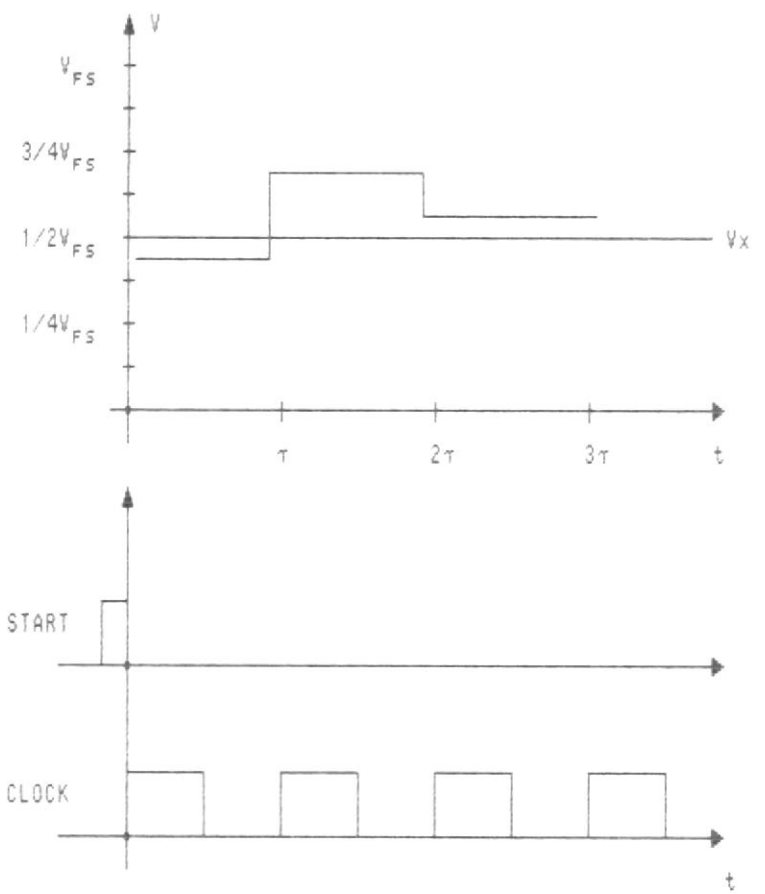


Figura 2.9

incrementada o decrementada en $V_{FS}/8$. Se hace una tercera comparación, luego de la cual la última salida binaria no es cambiada si Vx fue mayor que la última salida del DAC, o es decrementada en un LSB si Vx fue menor que la salida del ADC. La conversión se completa al final de tres períodos de reloj para un convertidor de tres bits o al final de n períodos de reloj para un convertidor de n bits.

Las posibles secuencia binarias para un DAC de tres bits y la secuencia seguida en la conversión por aproximaciones sucesivas de la figura 2.9 se muestra en la figura 2.10. En el inicio de la conversión, la entrada del DAC es inicializada en 100. Al final del primer período de reloj se encuentra que el voltaje del DAC es menor que Vx, por lo que el código del DAC se incrementa a 110. Al final del segundo período de reloj el voltaje del DAC es encontrado muy alto, y el código del DAC se decrementa a 101. Después del tercer período de reloj, el voltaje del DAC se encuentra todavía muy elevado, por lo que el código del DAC es decrementado a 100.

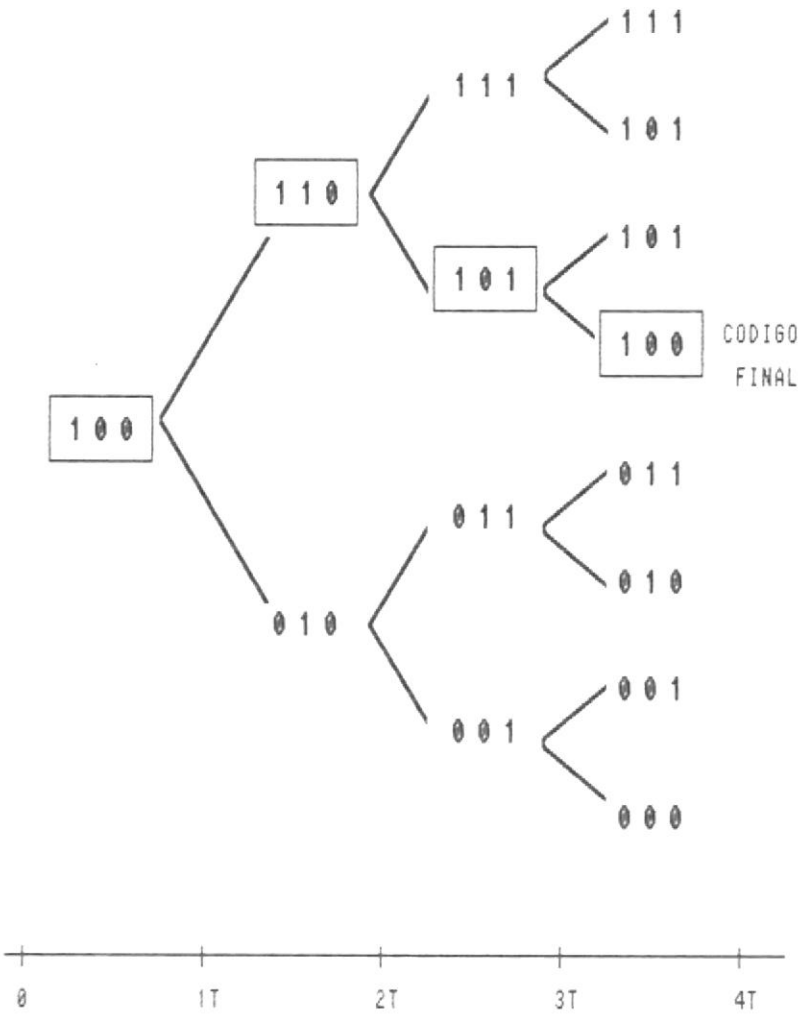

1 1 1
1 1 1
1 0 1
1 1 0
1 0 1
1 0 1
1 0 0
CODIGO
FINAL
1 0 0
0 1 1
0 1 1
0 1 0
0 1 0
0 0 1
0 0 1
0 0 0
0
1T
2T
3T
4T

Figura 2.10

Esta técnica permite altas velocidades de conversión. Por ejemplo, un convertidor de 10 bits con un reloj de 1 MHz puede completar una conversión en 10 microsegundos. La técnica de conversión de analógica digital por aproximaciones sucesivas es muy popular y es usada en muchos convertidores de 8 o 16 Bits.

Los principales factores que limitan la velocidad de este convertidor son el tiempo requerido por el DAC y el tiempo requerido para que el comparador responda a señales de entrada, las que pueden diferir en pequeñas cantidades.

Un problema en la aplicación de estos convertidores es que la entrada permanezca constante durante todo el período de conversión. Una pequeña variación en la señal de entrada es aceptable si no cambia por más de 0.5 LSB ($\frac{1}{2}V_{FS}*2^n$). La frecuencia de una entrada senoidal con una amplitud pico a pico igual al voltaje de escala completa del convertidor (V_{FS}) debe satisfacer la desigualdad:

$$fo \leq fc/n(2\pi)2^n.$$

Para una frecuencia de reloj de 1 MHz y un convertidor de 10 bits, fo debe ser menor de 16 Hz. Esto corresponde a una variación máxima de la entrada de 0.25 volts/milisegundo, para V_{FS} = 5 voltios. Si la entrada cambia a una razón más alta durante el proceso de conversión, la salida del convertidor no guardará una precisa relación con el valor del voltaje desconocido Vx. Para superar esta limitación de frecuencia se utilizan circuitos de muestreo y mantenimiento antes de los convertidores de aproximaciones sucesivas.

2.3.5 Convertidores Paralelos (Flash).

Los más rápidos convertidores utilizan circuitería adicional para realizar una conversión paralela en lugar de una en serie. El término "convertidor flash" a veces es usado para identificar los convertidores paralelos debido a la velocidad propia del dispositivo. La figura 2.11 muestra un convertidor paralelo de tres bits en el cual la entrada desconocida Vx es comparada simultáneamente con siete voltajes de referencia diferentes. La red lógica codifica la salida de los comparadores en los tres bits que representan el valor de el voltaje de entrada. Estos

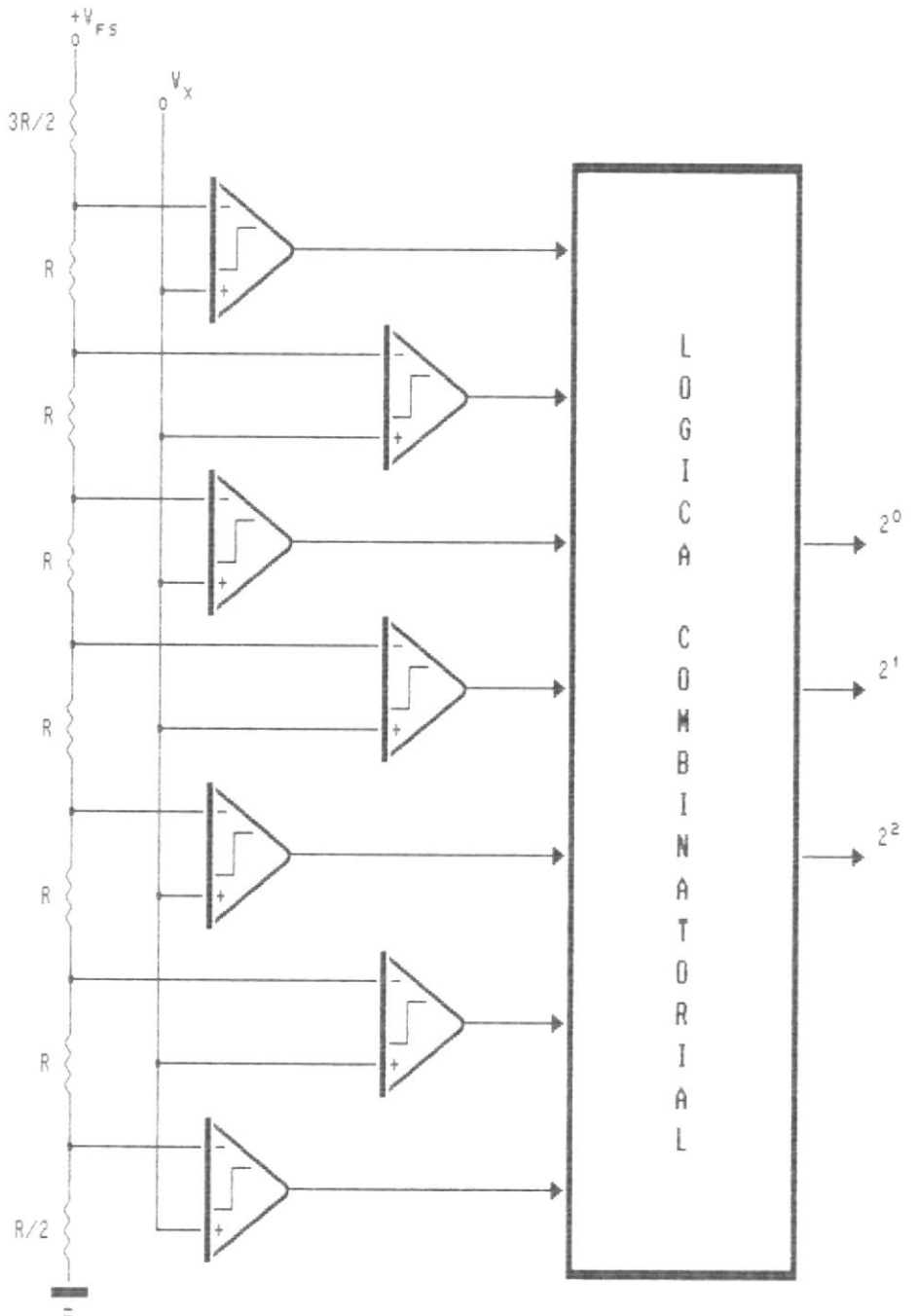


Figura 2.11

convertidores tienen una muy alta velocidad, debido a que solo están limitados por los retardos en los comparadores y en la red lógica. Además, la salida representa continuamente a la entrada, retardada por los comparadores y la red lógica.

Los convertidores analógico digitales paralelos son usados cuando se requiere la más alta velocidad y son encontrados usualmente convertidores de relativa baja resolución, debido a que se necesitan 2^n-1 comparadores y voltajes de referencia para un convertidor de n bits. Por tanto, el costo de la implementación de estos convertidores crece rápidamente con la resolución. Sin embargo, han sido realizados recientemente convertidores de 6, 8 y 10 bits, a través del uso de tecnología de circuitos integrados monolíticos. Estos convertidores alcanzan razones de conversión de 10 a 100 millones de conversiones por segundo.

En vista de que el Sistema de Generación de Diagramas de Tiempo requiere altas frecuencias de muestreo y por lo tanto un convertidor analógico-digital muy rápido, se seleccionó un convertidor

con una técnica de conversión basada en la conversión paralela. Este convertidor es el ADC0820 de National, que internamente posee dos convertidores paralelos de 4 bits.

CAPITULO III:

DISEÑO GENERAL DEL SISTEMA

DE GENERACION DE DIAGRAMAS DE TIEMPO.

3.1 Visión General del Sistema.

El sistema está conformado por dos componentes principales:

Un componente de hardware, el Circuito de Adquisición de Datos, que es el encargado de obtener las señales eléctricas mediante los puertos de entrada que dispone.

Un componente de software, que consiste en un programa que ofrece las funciones de: lectura de parámetros de funcionamiento, lectura de las señales eléctricas, y graficación a través de pantalla e impresora.

El Circuito de Adquisición de Datos provee de la circuitería necesaria para la obtención de las señales eléctricas provenientes de un circuito digital. Esta circuitería consiste básicamente en 2 puertos de entrada de 8 bits, para la lectura de las 16 señales

digitales, y dos convertidores analogico- digitales con resolución de 8 bits para la lectura de 2 señales digitales. Estos convertidores son vistos por el procesador como puertos de entrada.

El usuario deberá especificar los parámetros para el muestreo, que son el tiempo de muestreo y la frecuencia de muestreo. El programa va muestreando sucesivamente los puertos durante el tiempo que especifique el usuario y va almacenando los datos en memoria. Esto se realiza a la frecuencia de muestreo especificada.

Una vez almacenados los datos, el programa podrá graficar las señales mostradas en el monitor del computador, permitiendo recorrer las señales durante todo el tiempo de muestreo, o imprimirlas en un intervalo de tiempo especificado por el usuario.

3.2 Funciones del Hardware.

El hardware consiste en un Circuito de Adquisición de Datos, a través del cual el sistema obtiene las señales a ser graficadas. Este circuito estará conectado a la barra del computador, a través de una ranura de expansión.

El Circuito de Adquisición de Datos tiene como función tomar señales eléctricas del circuito digital que se está analizando y ponerlas en la barra, a disposición del microprocesador. Las señales digitales son leídas a través de dos puertos de entrada de 8 bits cada uno y las dos señales analógicas son digitalizadas por dos convertidores analógico-digitales de 8 bits de resolución.

En cada muestreo, las 16 señales digitales son enclavadas simultáneamente, 8 en el puerto 0340H y 8 en el puerto 0341H. Al mismo tiempo, arranca la conversión en los convertidores analogico-digitales, que son vistos como puertos de entrada en las direcciones 0342H y 0343H. Posteriormente los puertos son leídos secuencialmente.

Las direcciones y funciones de los puertos que provee el Circuito de Adquisición de Datos son:

0340H : Puerto de entrada para las señales digitales D0 a D7.

0341H : Puerto de entrada para las señales digitales D8 a D15.

0342H : El primer convertidor analogico-digital es visto por el procesador como un puerto de entrada en esta dirección.

0343H : El segundo convertidor analogico-digital es visto por el procesador como un puerto de entrada en esta dirección.

0344H : Al leer en este puerto, se enclavan simultáneamente los datos en los puertos de lectura de las señales digitales y se arranca la conversión de las señales analógicas a digitales.

3.3 Funciones del Software.

El programa permite la lectura, almacenamiento en memoria y representación visual por pantalla o por impresora de los datos obtenidos a través del Circuito de Adquisición de Datos, a más de que provee de la interfaz para el ingreso interactivo de datos tales como los parámetros del muestreo, nombres de señales y otros datos.

El programa permite escoger entre tres modos de muestreo:

8 Digitales: El muestreo se realiza leyendo solamente el primer puerto de entrada para las señales digitales. Aunque solo permite muestrear las 8 primeras señales digitales, en este modo se consiguen las más altas frecuencias de muestreo.

16 Digitales: En este modo se muestrean los dos puertos de entrada de señales digitales.

16 Digitales-2 Analog.: Se muestrean los dos puertos de entrada de señales digitales y las salidas de los 2 convertidores analogico-digitales.

La frecuencia y el tiempo de muestreo son también parámetros que deben ser especificados por el usuario.

Basándose en el valor de la frecuencia de muestreo, el programa calcula el valor del contador de retardo, y en base al tiempo y la frecuencia de muestreo, calcula el numero de muestreos a realizar.

El arranque es determinado por el Modo de arranque y la señal de arranque. En modo Manual, el arranque se realiza a partir del momento en que se presione la tecla de arranque. En modo Controlado, luego de presionar la tecla de arranque se muestrean

sucesivamente las entradas hasta el momento que la señal de arranque tome el nivel de voltaje especificado, entonces se producirá el arranque.

El programa también presenta la pantalla de visualización de señales, la cual permite un fácil recorrido a través del diagrama de tiempo, desplazándose en cualquier sentido, y permitiendo incluso especificar la posición en el tiempo a la que se desea trasladarse.

El programa también realiza la impresión de los diagramas de tiempo obtenidos, permitiendo especificar el titulo del diagrama, el intervalo de tiempo que se desea imprimir y los nombres para la identificación de señales.

CAPITULO IV

DISEÑO DEL HARDWARE

4.1 Exposición del Diseño

4.1.1 Generalidades

El hardware a desarrollar consiste en una interfaz que permite que las señales que provienen de un circuito digital puedan ser leídas por el microprocesador. Esta interfaz es denominada "Circuito de Adquisición de Datos" . Se necesitan leer 16 señales digitales en formato TTL o CMOS polarizado con 5 voltios y 2 señales en formato analógico.

Para lograr este objetivo se deben proveer los puertos de entrada que sean necesarios. El microprocesador dispondrá de 4 puertos de entrada. Dos de estos leen las señales digitales, 8 bits cada uno. Los otros dos dispositivos que son vistos como puertos son en realidad convertidores analógico-digitales que poseen una adecuada circuitería interna para la interacción con microprocesadores.

También es necesaria la realización de la circuitería que realiza la selección del puerto que requiera el microprocesador.

Los cuatro puertos deben de realizar la adquisición de datos de manera simultánea. Para el efecto, el circuito a diseñar permite que, al ejecutar el microprocesador una instrucción de lectura a la dirección de E/S 0344H, se genere una señal que provoca el enclavamiento de los datos presentes a la entrada de los puertos, y el inicio de conversión en convertidores analógico-digitales.

Los datos son obtenidos leyendo sucesivamente los dispositivos que cumplen la función de puertos de entrada. En cada lectura son leídos 8 bits. En la primera y segunda lectura son leídas las 16 señales digitales. En la tercera lectura son leídos 8 bits de la primera señal analógica digitalizada, y en la cuarta lectura son leídos los 8 bits de la segunda señal analógica.

4.1.2 El Decodificador de Direcciones.

El decodificador de direcciones tiene como propósito establecer un mapeo entre una dirección y un dispositivo. En nuestro caso, 4 direcciones mapean cada una a un dispositivo diferente para el efecto de que el microprocesador reciba los datos provenientes de ellos , pero una quinta dirección actúa como una señal para que todos los dispositivos tomen los datos que poseen en sus entradas.

El circuito presenta 5 salidas de selección, que corresponden a las direcciones de memoria de 0340H a 0344H. Las cuatro primeras localidades habilitan para lectura a los puertos 0, 1, 2 y 3. Al direccionar 0344H se activa la señal WR.L para indicar a los puertos que deben tomar los datos que tienen a la entrada.

Al ejecutar una instrucción IN, la señal IOR.L se pone en bajo, y la dirección que actuó como operador en la instrucción IN se hace presente en la barra de direcciones. En el decodificador se habilita la salida según la dirección seleccionada. Un diagrama del decodificador de

direcciones se encuentra en la figura 4.1

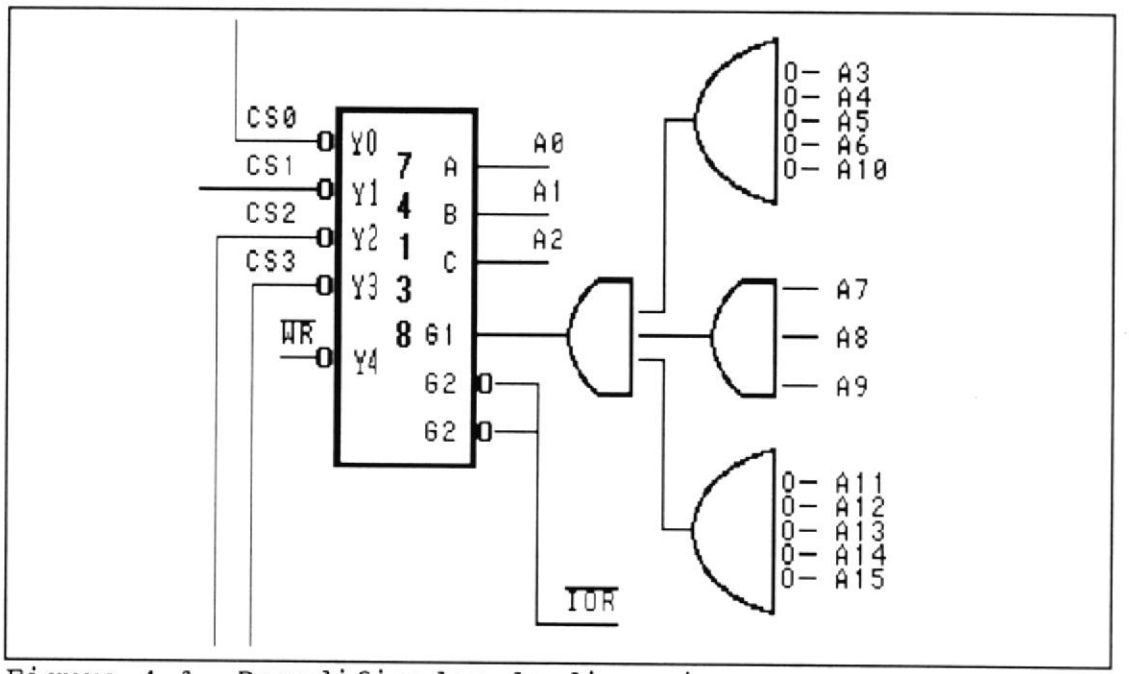


Figura 4.1. Decodificador de direcciones

4.1.3 Adquisición de las señales digitales

Las 16 señales digitales son leídas por medio de 2 puertos de entrada, cada uno de 8 bits.El puerto de entrada que efectúa la lectura de 8 señales digitales está formado por un banco de 8 Flip Flops tipo D, y un banco de 8 buffers triple estado, que se encuentra conectado a la barra de datos del sistema.

Cuando WR.L se activa, provoca el enclavamiento de los datos de la entrada de los Flip Flops. Los datos son leídos por el microprocesador si se

activa la señal de selección, la cual, conectada a la señal de habilitación del banco de buffers, permite que estos abandonen el estado de alta impedancia.

4.1.4 Adquisición de las señales en formato analógico.

Debido a que necesitamos la mayor frecuencia de muestreo posible, se ha escogido el convertidor analógico digital ADC-0820 , de National, pues posee un tiempo de conversión de 2us. Además, el ADC-0820 es un convertidor analógico-digital que brinda facilidades para la comunicación al sistema de microprocesadores, pues su salida es mediante buffers de triple estado. Posee 3 modos de operación: Modo lectura (RD), modo escritura luego lectura (WR then RD), y modo Stand-Alone.

El ADC 0820 se lo utilizará en el modo de operación escritura luego lectura (WR then RD) porque permite que la interfaz digital con las señales de la barra del computador sea sencilla, además en este modo es el que permite lograr menores tiempos de conversión (600 ns) . El dispositivo funciona en este modo cuando el pin

MODE es puesto en alto. Una conversión es iniciada cuando la señal WR.L se pone en bajo; sin embargo, existen dos opciones para leer los datos de salida, que se relacionan con la temporización de la interfaz. Si es deseado un funcionamiento a base de interrupciones, el usuario puede esperar que la señal INT.L se ponga en bajo antes de leer las salidas. Sin embargo, si se desea un tiempo de conversión más corto, el procesador no necesita esperar a INT.L y puede ejecutar una lectura después de esperar solo 600 ns después de haberse puesto WR.L en bajo. Al hacer esto, INT.L se pone en bajo inmediatamente y los datos aparecen a la salida.

Cuando se ejecuta una instrucción de lectura (IN) a la dirección de E/S 0344H, la señal WR.L, a la salida del decodificador se pone en bajo, lo que produce también que las señales de habilitación de los convertidores analógico-digitales también se pongan en bajo. De esta manera, se inicia simultáneamente la conversión en ambos dispositivos. Luego, ejecutando instrucciones IN de la dirección 0342H y 0343H las entradas RD se ponen en bajo para el primero y el segundo convertidor, respectivamente; esto permite que

los datos a la salida de los convertidores puedan ser leídos inmediatamente.

La señal de entrada llega al convertidor analógico-digital correspondiente a través de un amplificador operacional, conectado en configuración seguidor de emisor. Estos dispositivos cumplen la función de desacoplar eléctricamente el circuito, cuyas señales se están midiendo, del circuito de adquisición de datos .

4.1.5 Implementación del Circuito de Adquisición de Datos.

El Circuito de Adquisición de Datos debe ser conectado a una de las ranuras del computador en el cual se instale este sistema. Este circuito será construido a manera de circuito impreso, y se conectará a una ranura a través del conector propio de la tarjeta del circuito impreso. Las señales del circuito digital a analizar son llevadas al computador por un cable que es conectado al Circuito de Adquisición de Datos por medio de un conector de 25 pines. El diagrama de posicionamiento de la tarjeta del circuito se

puede observar en la figura 4.2.

Los componentes del Circuito de Adquisición de Datos se encuentran enumerados en la Tabla I.

TABLA I

COMPONENTE	NUMERO	DESCRIPCION
IC1,3	74273	8 Flip Flops tipo D
IC2,4,6,7	74244	8 Buffers de triple estado
IC5	74138	Decodificador de 3 a 8
IC8,9	741	Amplificador Operacional
IC10,11	ADC0820	Convertidor analógico-digital
IC12	7408	4 puertas AND de 2 entradas
IC13	7411	3 puertas AND de 3 entradas
IC14	74260	2 puertas NOR de 5 entradas
IC15	7414	Inversor de Histérisis
C1-4		Capacitor de 10μF, 25 V
R1,2		Resistencia de 10 MΩ

Diagrama de Posicionamiento de la Tarjeta del Circuito de Adquisicion de datos

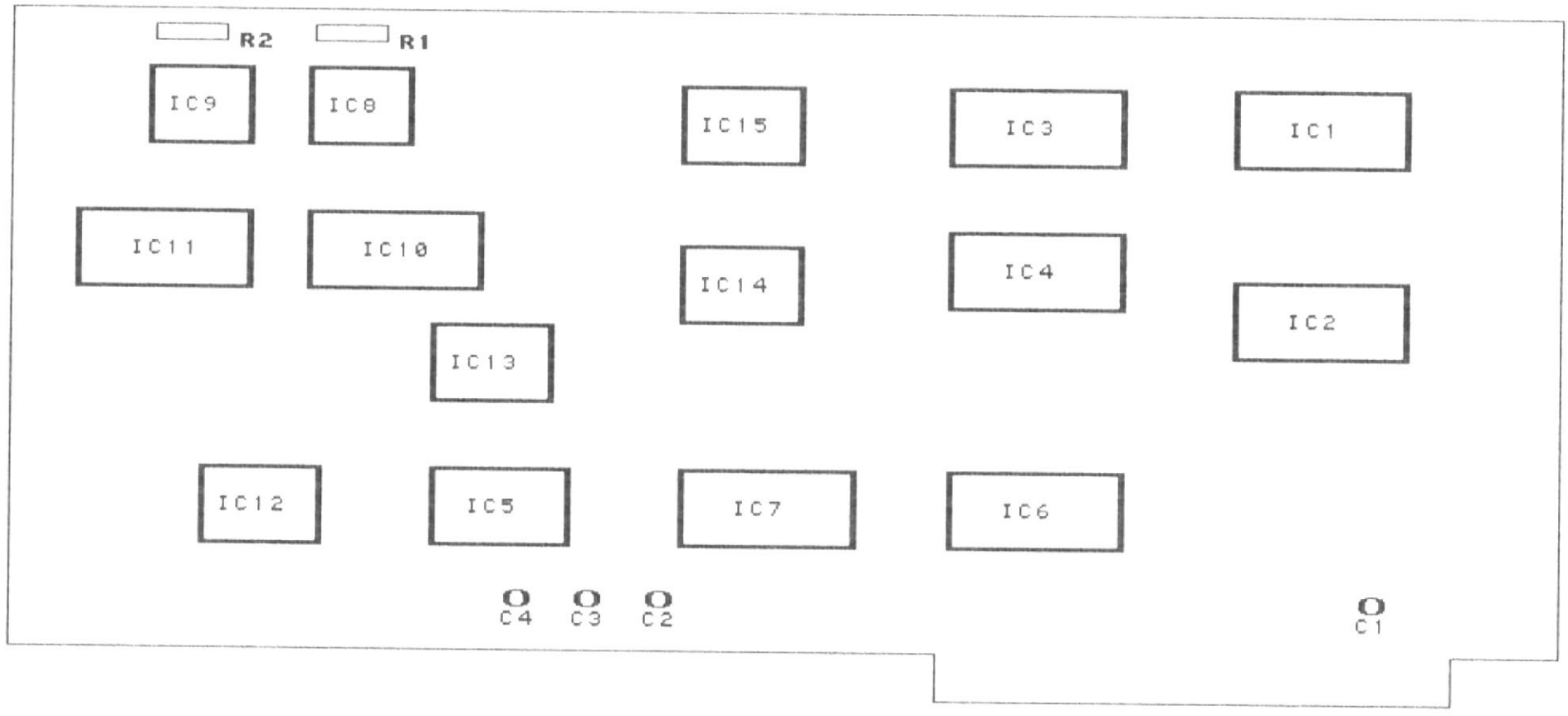


Figura 4.2

4.1.6 La Caja de Conecciones

Las señales que se van a graficar deben ser conectadas a la Caja de Conecciones, que provee de puntos para la conección de 16 señales digitales, 2 analógicas y un punto para conección de la tierra del circuito al que pertenecen estas señales. La Caja de Conecciones posee un conector de 25 pines , en el cual debe conectarse el cable que envía las señales al Circuito de Adquisición de Datos; este cable transporta las 18 señales antes mencionadas y una línea de tierra. Un gráfico de la caja de conecciones se presenta en la figura 4.3.

4.2 Diagrama Esquemático.

El diagrama esquemático del Circuito de Adquisición de Datos permite tener una visión global del diseño del mismo, y se muestra en la figura 4.4. Las señales rotuladas A1 ... A31 corresponden a las señales del lado derecho de una de las ranuras del computador y las señales rotuladas B1 ... B31 corresponden al lado izquierdo de dicha ranura.

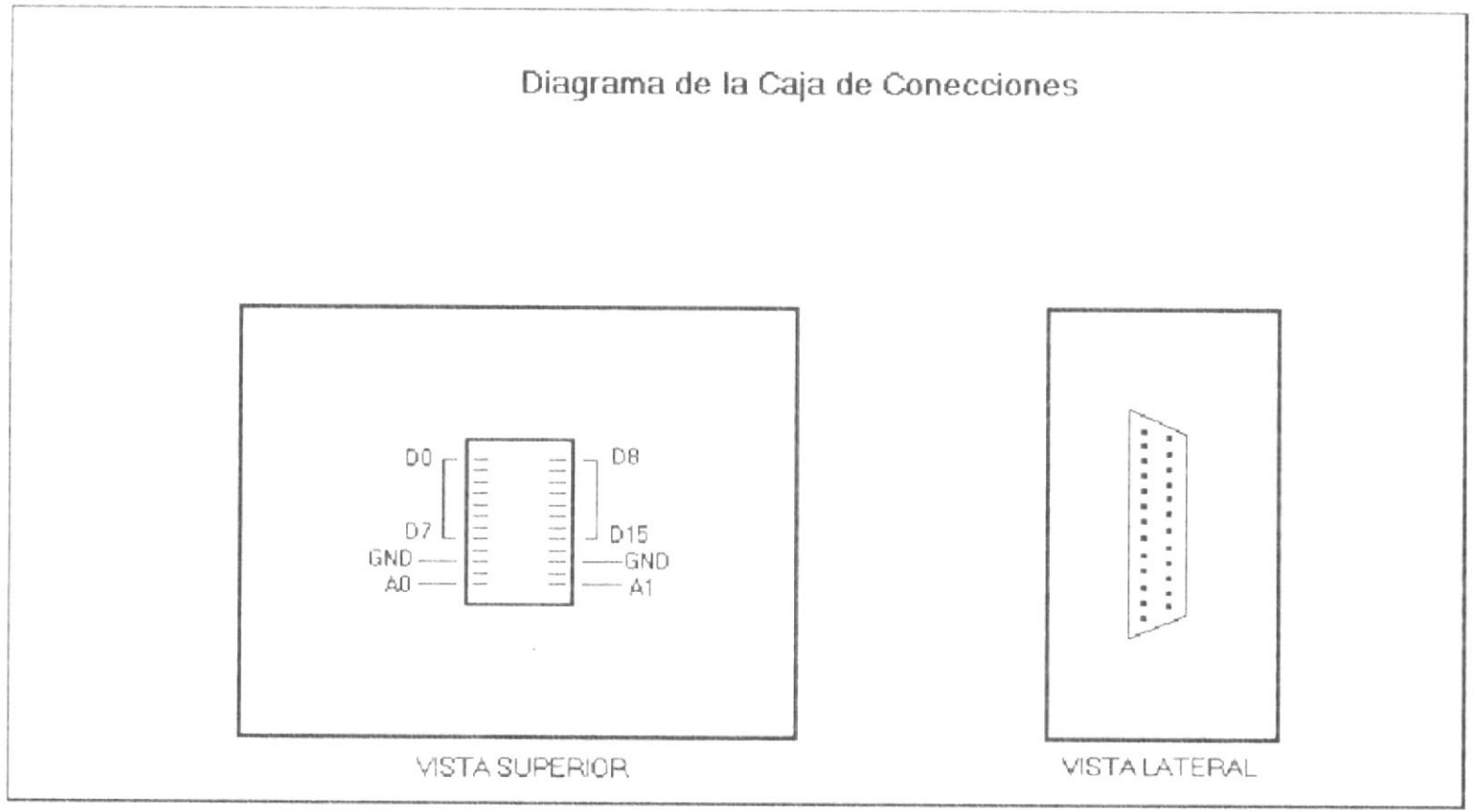


Diagrama de la Caja de Conecciones
D0
D7
GND
A0
D8
D15
GND
A1
VISTA SUPERIOR
VISTA LATERAL

Figura 4.3

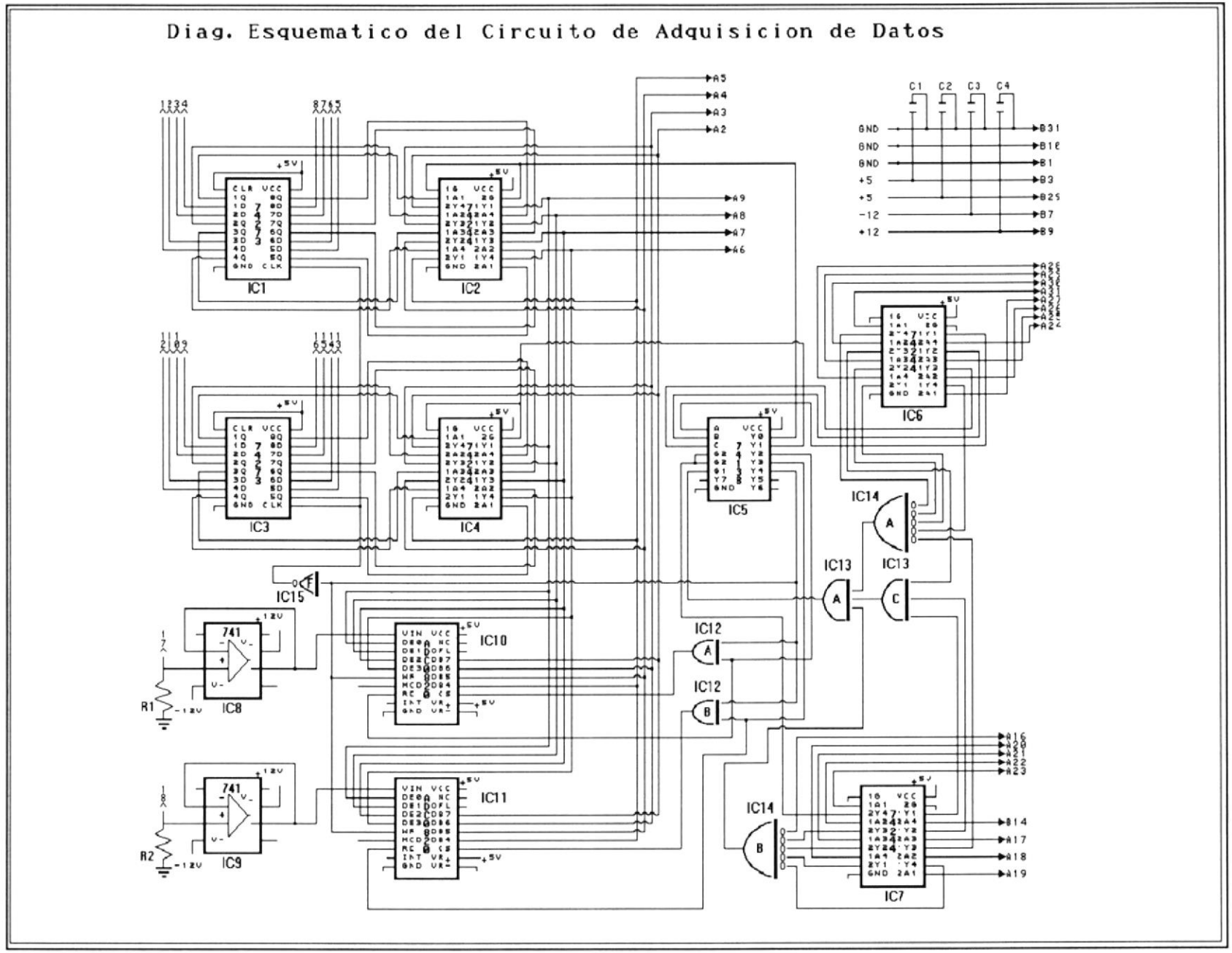

Diag. Esquematico del Circuito de Adquisicion de Datos
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
IC15
741
R1
R2
C1
C2
C3
C4
GND
+5
-12
+12
+5V
-12V
+12V

Figura 4.4

4.3 Diagramas de Tiempo.

Los diagramas de tiempo de las Figuras 4.5 y 4.6 muestran las señales de direccionamiento del microprocesador (A0 ... A15), la señal de habilitación de lectura para un dispositivo de E/S (IOR.L), la señal del circuito de adquisición de datos que indica escritura a los dispositivos y su correspondiente negada (WR.H y WR.L). También se muestran las señales de selección de los dispositivos: CS0 corresponde al primer puerto de entrada, CS1 corresponde al segundo puerto de entrada, CS2 al primer convertidor analógico-digital y CS3 al segundo convertidor analógico-digital. El nivel bajo de IOR.L, en combinación con la dirección del puerto, ponen en bajo estas señales de habilitación.

En la figura 4.5 observamos las tres primeras etapas del proceso de muestreo. IOR.L se pone en bajo con lo que WR se hace verdadero. Las señales CS2 y CS3 se ponen en bajo, poniendo a los ADC en espera de una instrucción. Con el flanco ascendente positivo de WR, los datos son enclavados en los dos bancos de flip flops tipo D, y se inicia la conversión en ambos convertidores analógico-digitales. Luego se procede a leer los datos del puerto 0 (primer puerto de entrada),

poniendo en bajo CS0. De igual manera se procede con el puerto 1 (segundo puerto de entrada) , el cual se lee con el nivel bajo de CS1.

En la figura 4.6 podemos ver las siguientes etapas del muestreo. Luego de haber leído los dos primeros puertos, ya se ha terminado la conversión en los convertidores analógico-digitales. El nivel bajo de CS2 pone al primer convertidor en estado de lectura, y los datos son transferidos al microprocesador. En la siguiente etapa CS3 se pone en bajo, con lo que los datos del segundo convertidor son leídos por el microprocesador.

Diagrama de Tiempo (a)

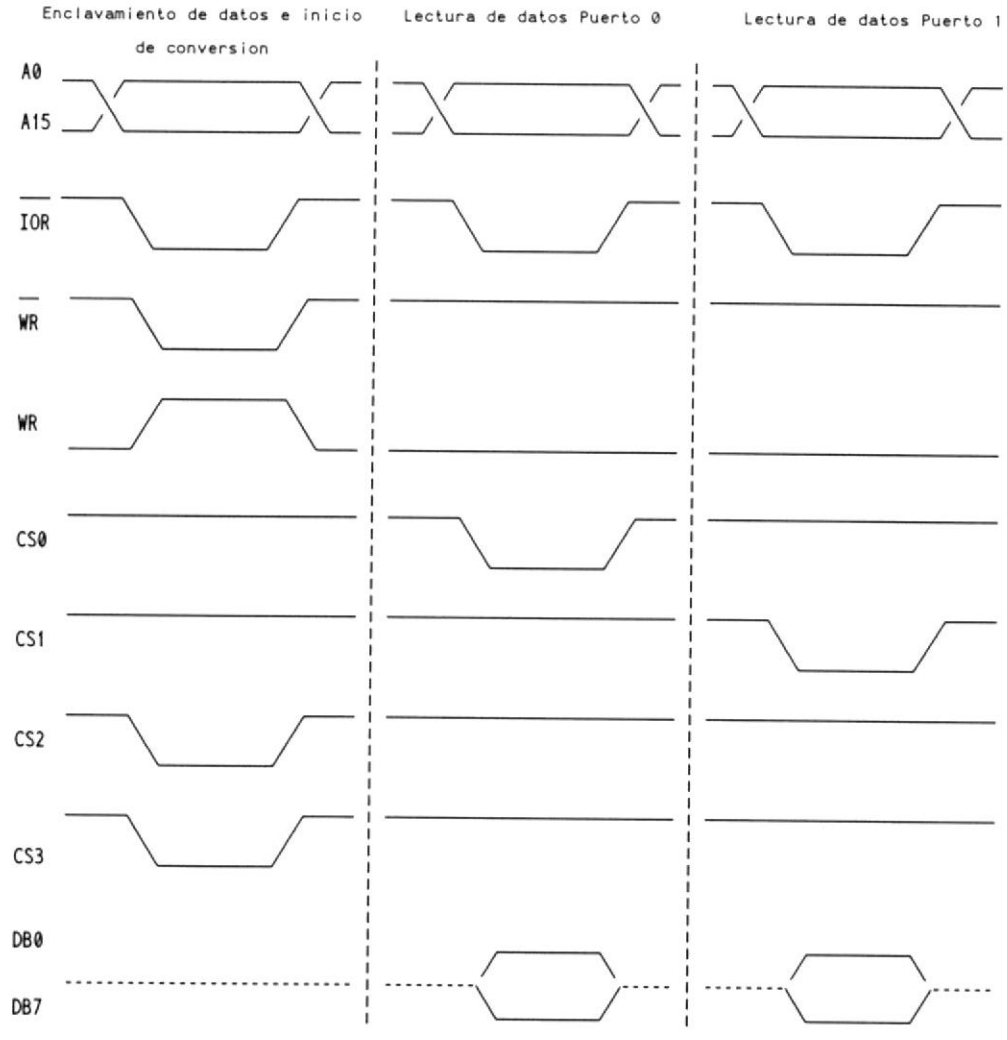


Figura 4.5

Diagrama de Tiempo (b)

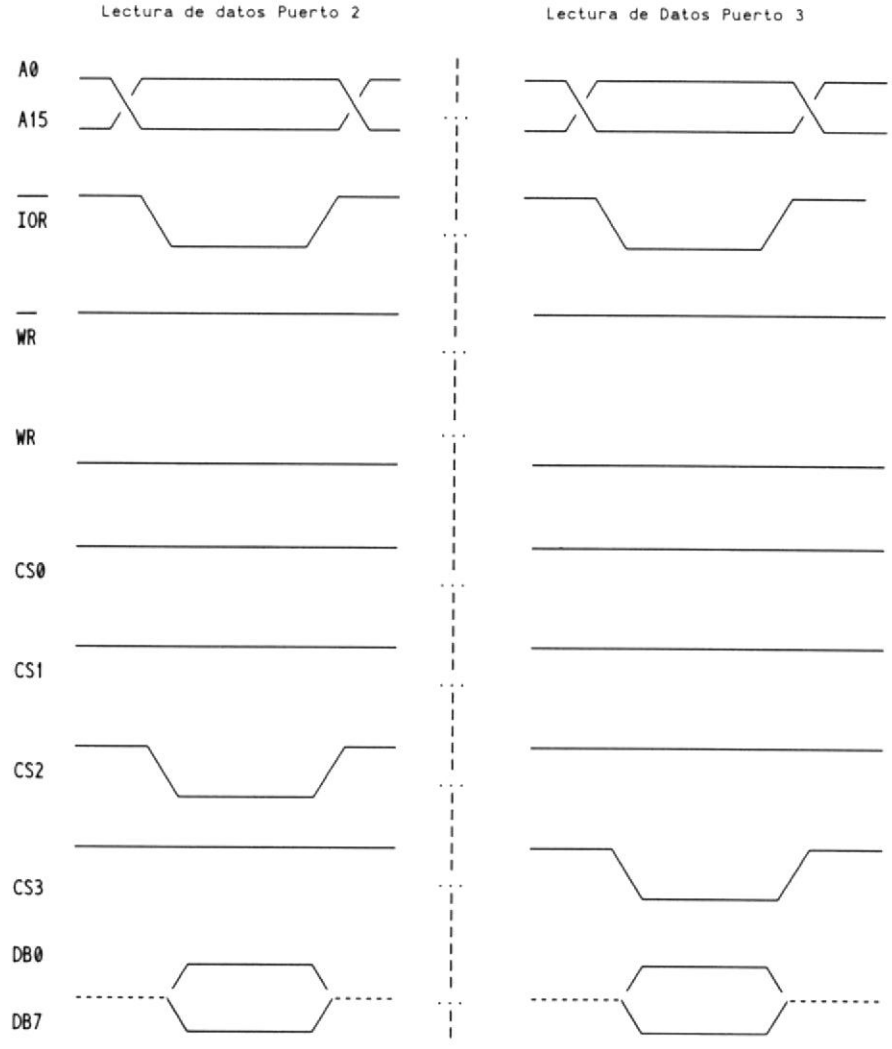


Figura 4.6

CAPITULO V

DISEÑO DEL PROGRAMA PARA EL SISTEMA DEL GENERACIÓN DE DIAGRAMAS DE TIEMPO.

5.1 Especificación de la Interfaz con el Usuario.

El sistema adquiere los datos de las señales analógicas y digitales por medio de una interfaz entre estas señales y el microprocesador, interfaz que consiste en un circuito de adquisición de datos. Por otra parte, el sistema requiere datos de control proporcionados por el usuario, como son los parámetros para el muestreo, los comandos para la observación de las señales a través de la pantalla y las especificaciones para la graficación impresa de las mismas. Todos estos datos de control son receptados por el sistema a través de la interfaz con el usuario, que consiste en pantallas de menúes, ventanas para ingreso de datos de datos de control de muestreo y de impresión de reportes.

Las diversas opciones del programa son ejecutadas al presionar teclas funcionales; para facilitar el uso del mismo las opciones con las teclas que las activan se especifican en la primera línea de la pantalla, a

manera de un menú horizontal. Por ejemplo:

F1:Ayuda F10:Salir.

Cuando sea necesario ingresar un conjunto de parámetros se presenta una ventana que permite ingresar los datos requeridos a manera de formulario.

En la ultima la línea de la pantalla se muestran mensajes de estado o de error. Por ejemplo:

Por favor, ingrese una frecuencia mas baja.

5.1.1 Pantallas de menúes, de ayuda y especificación de datos

5.1.1.1 Menú Principal

El menú principal se observa en la figura 5.1. Tiene las siguientes opciones:

F1: Ayuda: Se muestra una ventana con información sobre el uso del programa. Las teclas Page Up/Page Down proveen de las funciones de Avance/Retroceso de página. Figura 5.2

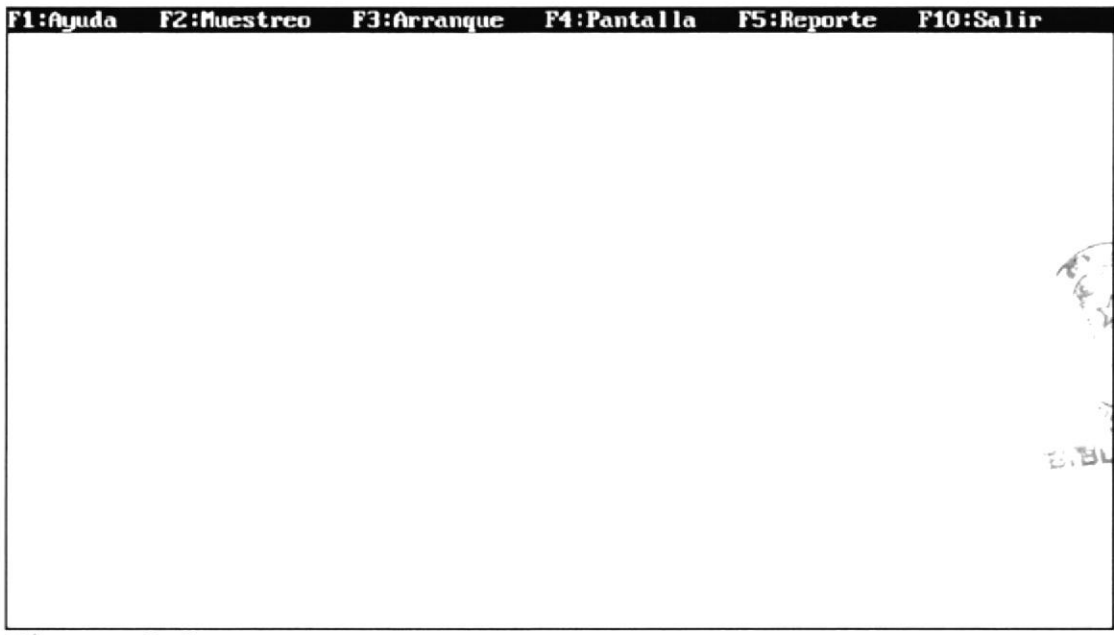


Figura 5.1

F2: Muestreo : Se muestra la ventana de Asignación de parámetros de Muestreo

F3: Arranque : En modo de muestreo manual, al presionar F3 se inicia inmediatamente el muestreo. en modo automático, el muestreo comenzará cuando la señal de arranque alcance el nivel especificado

F4: Pantalla: Se muestra la pantalla de Visualización de señales.

F5: Reporte: Se muestra la ventana para la especificación de datos para la

generación de la salida impresa.

F10 :Salir: Termina el programa y retorna al Sistema Operativo

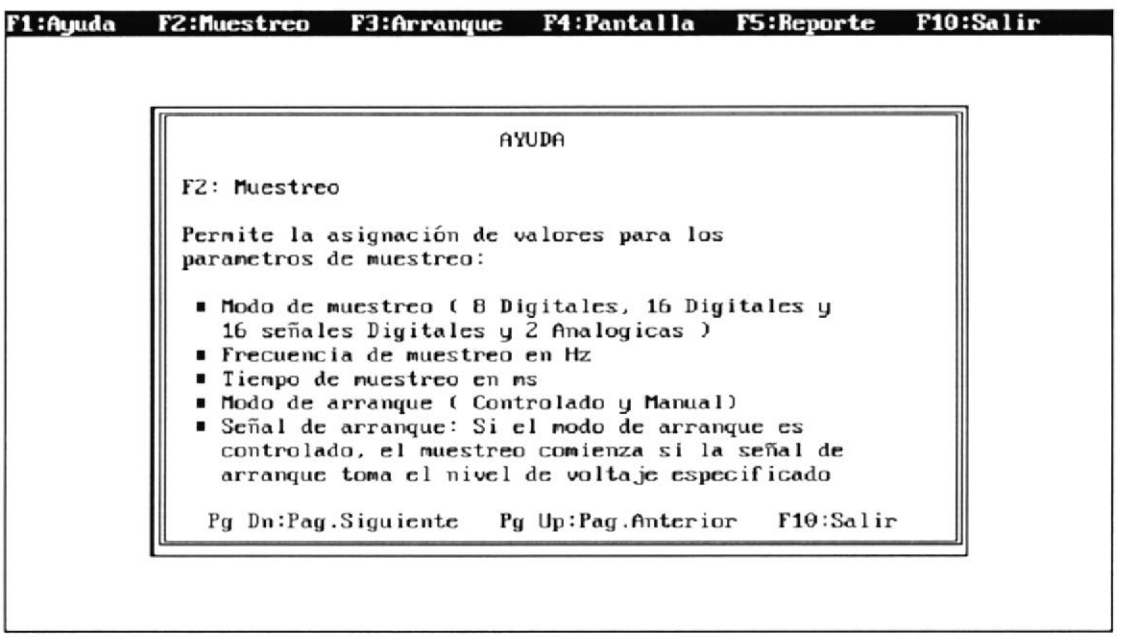


Figura 5.2

5.1.1.2 Asignación de Parámetros de muestreo.

En esta pantalla se presenta un formulario donde se definen los parámetros que requiere el programa para controlar la manera como se produce el muestreo (Figura 5.3) Estos parámetros son :

Modo de muestreo: Existen tres posibles modos de muestreo: 8 señales digitales, 16 señales digitales y 16 señales digitales mas dos analógicas.

Frecuencia de muestreo: La frecuencia en Hz a la que las señales presentes en la entrada son enclavadas para posterior lectura.

Tiempo de muestreo: El tiempo en milisegundos que durará el muestreo de las señales

Modo de Arranque: Puede ser Controlado o Manual. En modo Manual, el muestreo comienza inmediatamente después de

presionada la tecla de arranque. En modo Controlado ,después de presionar la tecla de arranque , el muestreo comenzará cuando la señal de arranque alcance el nivel especificado.

Señal y nivel de arranque: Estos parámetros deben ser especificados en caso de que el modo de arranque sea Controlado.En modo Controlado ,después de presionar la tecla de arranque , el muestreo comenzará cuando la señal de arranque alcance el nivel especificado.

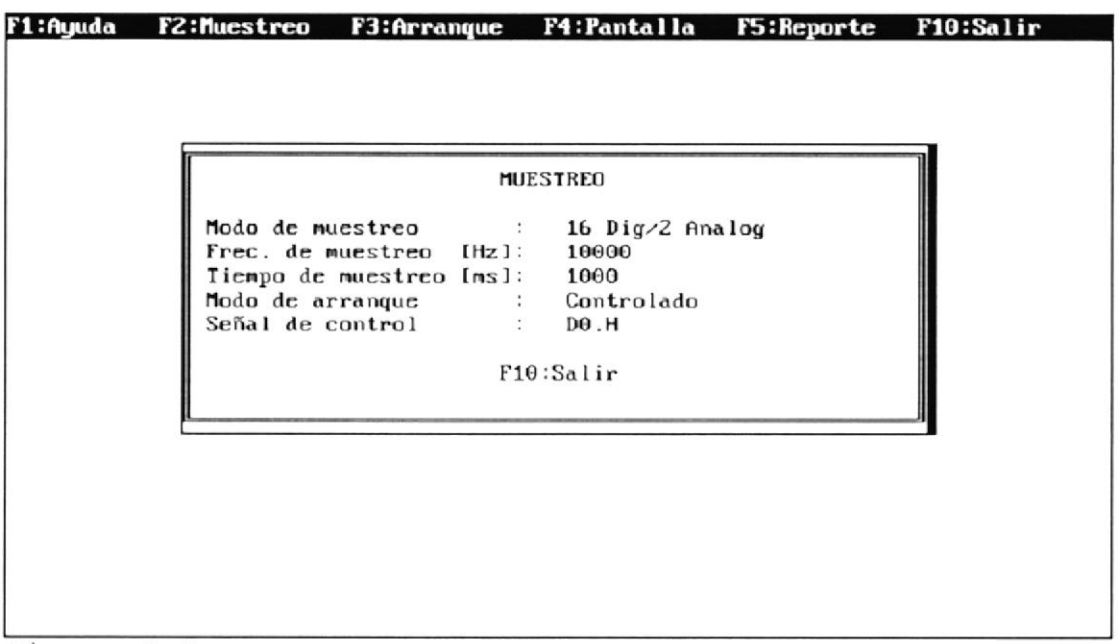


Figura 5.3

5.1.1.3 Especificación de datos para reporte.

Los datos para reporte son ingresados por el usuario a través de un formulario, donde deberá ingresar los siguientes campos.

Titulo : Una cadena de caracteres que se imprimirá en la cabecera del reporte, a manera de título.

Tiempo inicial : Indica desde que instante de tiempo serán mostradas las señales en el reporte.

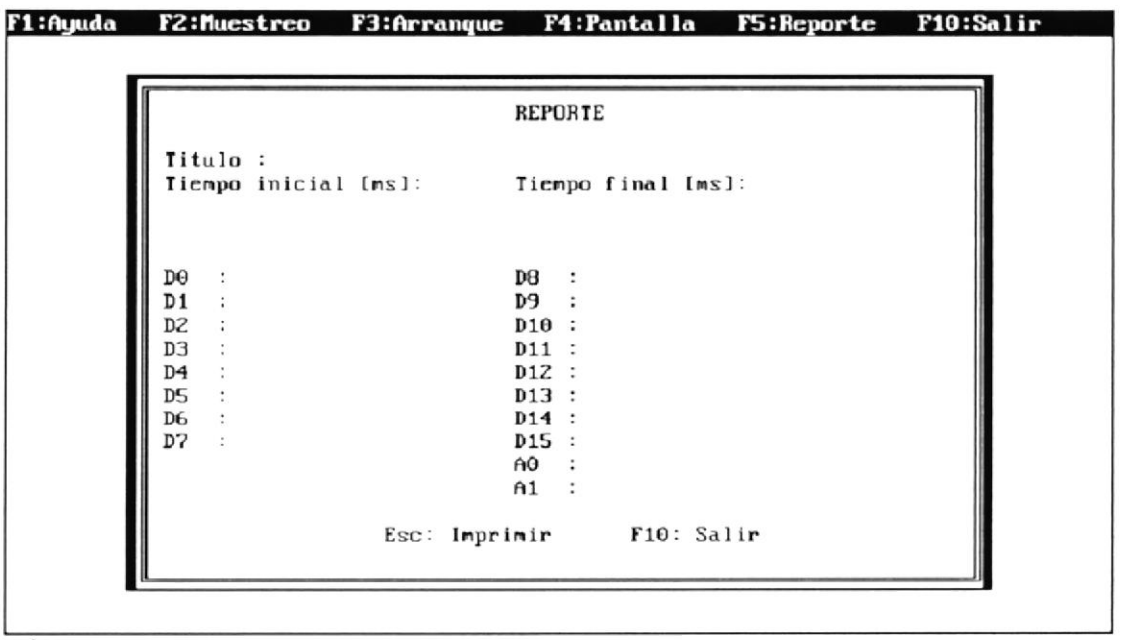


Figura 5.4

Tiempo final : Indica hasta que tiempo serán mostradas las señales en el reporte.

Nombres de las Señales : En el formulario aparecerán todas las señales muestreadas, cuya cantidad varía dependiendo del modo de muestreo. El usuario deberá asignar un nombre a cada señal que desee mostrar en el diagrama de tiempo. Si no desea mostrar una señal no deberá asignarle ningún nombre.

El usuario puede desplazarse de un campo a otro con las flechas del teclado. La tecla ESC inicia la impresión del reporte y la tecla F10 retorna al menú principal. Ver Figura 5.4

5.1.1.4 Menú de la Pantalla de Visualización de señales.

El menú de la pantalla de visualización de señales se observa en la figura 5.5. Tiene las siguientes opciones:

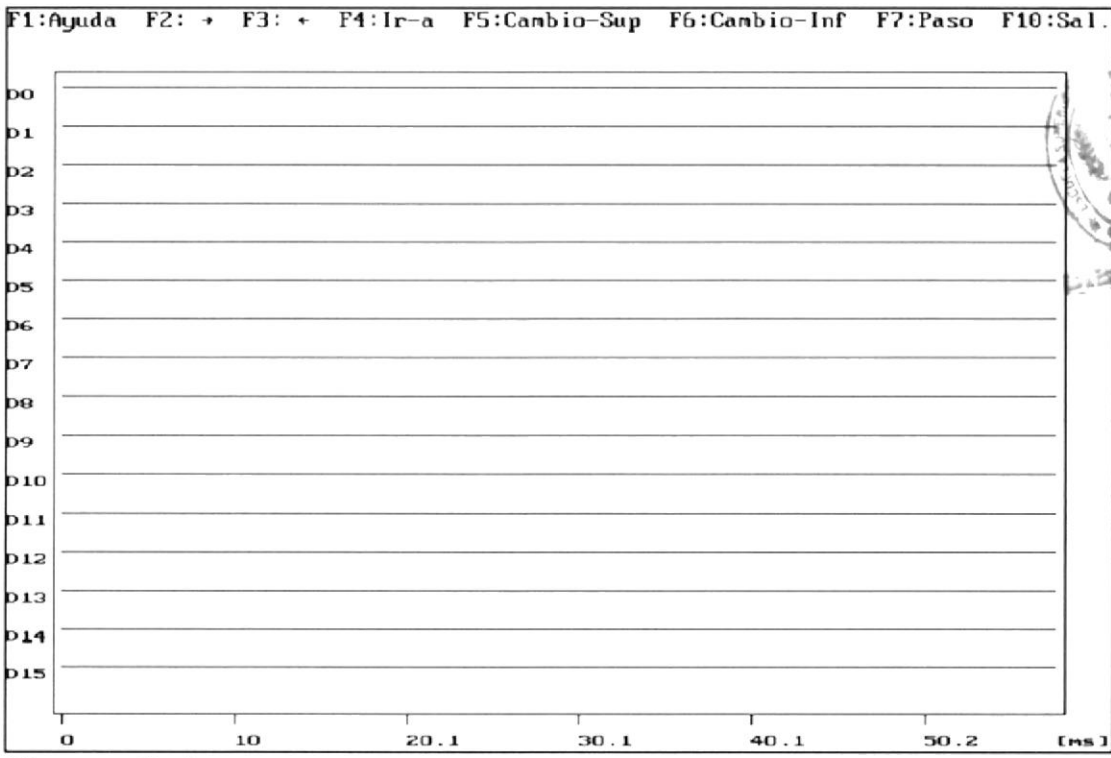


Figura 5.5

F1: Ayuda : Se presenta una ventana de ayuda sobre el uso de la pantalla de visualización de datos. Figura 5.6.

F2: → : Sirve para desplazarse horizontalmente avanzando en el tiempo.

F3: ← : Sirve para desplazarse horizontalmente retrocediendo en el tiempo.

F4: Ir-a : Permite especificar una

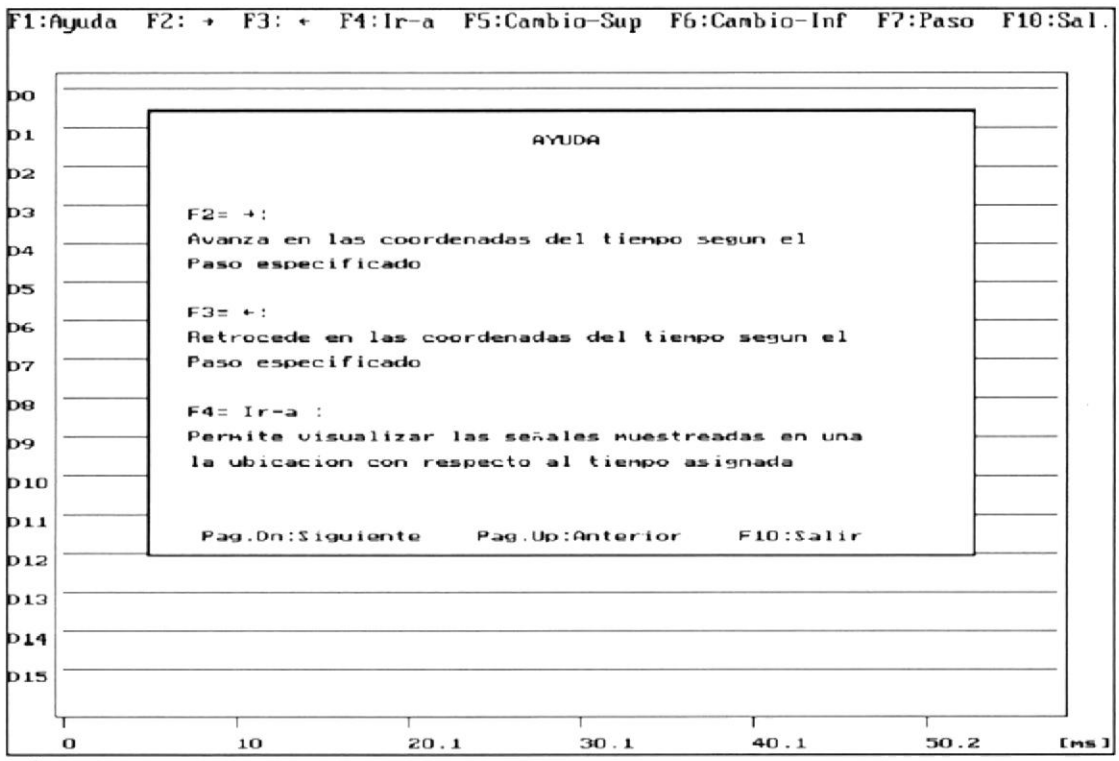


Figura 5.6

coordenada en el tiempo y luego desplaza la pantalla hasta esta coordenada.

F5: Cambio-Sup: Permite conmutar la zona superior de la pantalla con otro conjunto de señales, en el modo de 16 señales analógicas y 2 digitales.

F6: Cambio-Inf: Permite conmutar la zona inferior de la campaña con otro conjunto de señales, en el modo de 16 señales analógicas y 2 digitales.

F7: Paso : Mediante esta opción se podrá especificar otro valor para el paso, es decir la cantidad de milisegundos que avanza o retrocede la pantalla al presionar F2 o F3.

F10: Salir : Retorna al menú principal.

5.1.2 Salidas impresas y por pantalla.

5.1.2.1 Especificación del formato de impresión del diagrama de tiempo.

Una de las cualidades del reporte es que en el diagrama de tiempo solo se mostraran las señales definidas por el usuario. El programa aprovechará el espacio disponible mostrando las señales con mayor amplitud, para que el diagrama sea más fácil de leer. Ver Fig 5.7.

Cada hoja del diagrama de tiempo tiene tres partes: La cabecera, donde se presenta el título del reporte , el cuerpo,

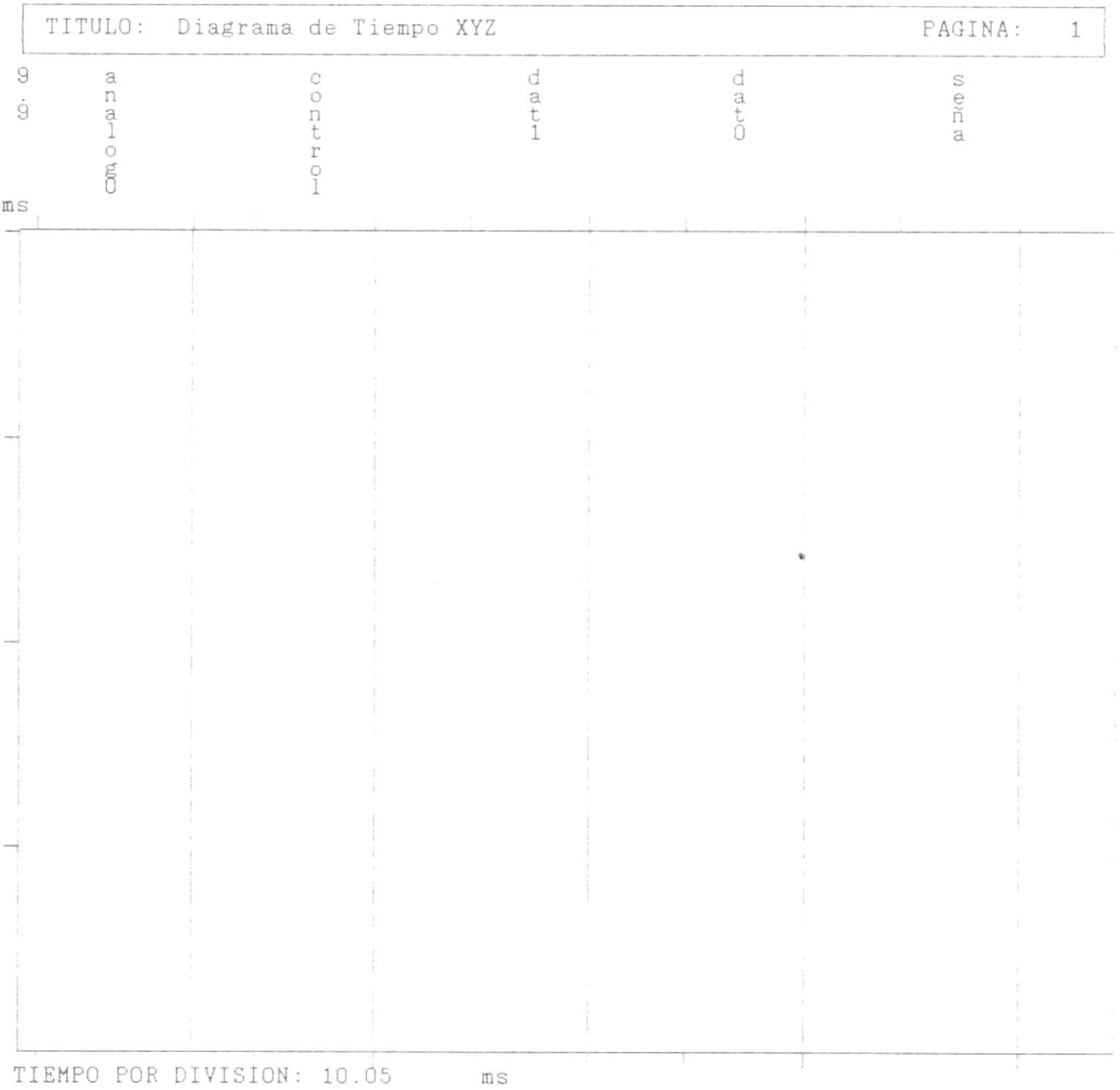

TITULO: Diagrama de Tiempo XYZ
PAGINA: 1
9
.
9
analog0
control
dat1
dat0
señ
a
ms
TIEMPO POR DIVISION: 10.05 ms

Figura 5.7

donde se grafican las señales, y el número de página actual enmarcados.

El cuerpo: En el cuerpo se grafican las señales que fueron etiquetadas por el usuario. Al inicio del cuerpo aparecen los nombres de las señales impresos en sentido vertical. También se presenta la posición en el tiempo desde el cual se grafica la presente página.

El fin de página: Al concluir el cuerpo, se imprime el fin de página, donde se presenta el tiempo en milisegundos que hay entre cada división.

5.1.2.2 Pantalla de Visualización de señales.

Una de las maneras en que el usuario podrá obtener el diagrama de tiempo es a través de la pantalla de visualización de señales, que permite observar las señales muestreadas por todo el intervalo de muestreo. En la pantalla se presenta una zona que es parte del total muestreado. Se puede avanzar o retroceder en el

tiempo desplazando la pantalla en dirección horizontal en uno u otro sentido.

Podemos observar la Pantalla de Visualización de señales en la Figura 5.5. En la parte superior de la pantalla se ubicará una línea donde se presentaran las acciones posibles a manera de menú horizontal. Estas opciones son seleccionadas mediante las teclas funcionales especificadas.

La pantalla se divide en dos zonas de presentación de datos: la superior y la inferior.

En el modo de muestreo de 16 señales digitales y 2 analógicas, en cualquiera de estas zonas pueden mostrarse: las señales digitales D0 a D7, las señales digitales D8 a D15 o las señales analógicas A0 y A1. Las opciones del menú permiten conmutar una zona de un grupo de señales a otro. La opción "Cambio Superior" conmuta la zona superior y la

opción "Cambio Inferior" la otra zona. En el modo de 16 o de 8 señales digitales no es necesaria la conmutación , pues pueden verse simultáneamente todas las señales muestreadas.

En la parte inferior de la pantalla se presentan las coordenadas del tiempo a que corresponden los datos que se están mostrando. Los valores están en milisegundos.

5.2 Diseño Arquitectónico.

5.2.1 Estructura del Software.

Para poder definir la estructura del software, debemos primeramente identificar sus funciones principales. Estas son:

1. Permitir el ingreso de los parámetros de muestreo.
2. Determinar el inicio del muestreo.
3. Realizar el muestreo.
4. Presentar una pantalla para la visualización de señales.

5. Permitir el ingreso de los datos para reporte e imprimir el diagrama de tiempo.
6. Proporcionar ayuda al usuario.
7. Presentar el menú principal que invoca la ejecución de la funciones anteriores.

A estás debemos agregar una función interna del programa, la de calcular parámetros del programa en base a los parámetros de muestreo especificados por el usuario.

Cada una de estas funciones principales se implementará a través de varias funciones de programación. Un grupo de funciones de programación que son usadas para realizar una misma función del sistema se agrupan en un módulo. Los puntos 2 y 3 de la lista anterior se han unificado en un mismo módulo por ser similares. Los módulos del programa con las funciones que los componen se especifican en la Tabla II.

TABLA II

NOMBRE	OBJETIVO	FUNCIONES
princip	Muestra el menú principal y llama a las demas funciones del sistema.	main acepta_op
ayuda	Muestra una ventana de ayuda.	ayuda disp_help
leepar	Muestra la ventana de asignación de parámetros de muestreo y permite el ingreso de estos datos.	lee_datmuestreo abre_winbord
calcpar	Cálculo de los parámetros internos.	cal_param tmuestreo tretardo t2muestreo
muestreo	Determina el inicio del muestreo (arranque) y efectúa el muestreo.	muestreo arranque
pantalla	Presenta la pantalla de visualización de señales.	ver_pantalla muestra_gr marcos escalas procesa map_gr map_analog etiquetas gnum_str ayudap gdisp_help

NOMBRE	OBJETIVO	FUNCIONES
reporte	muestra la ventana de especificación de datos de muestreo y realiza la impresión de diagramas de tiempo	rp_ventana rp_calparam valida imp_diagrama inputf imp_nombre_sen imp_cabecera imp_marco imp_cuerpo imp_pie limpia_pr_línea pr_línea imprime imp_escala imp_analog imp_digit envia

En el diagrama de la figura 5.8 (a y b) se presenta la estructura arquitectónica del software, donde es posible identificar las funciones con su relación jerárquica.

5.2.2 Estructura de datos.

En cada muestreo son leídos los cuatro puertos de entrada de 8 bits cada uno. Los puertos que leen las señales digitales (0x0340 y 0x0341) en cada

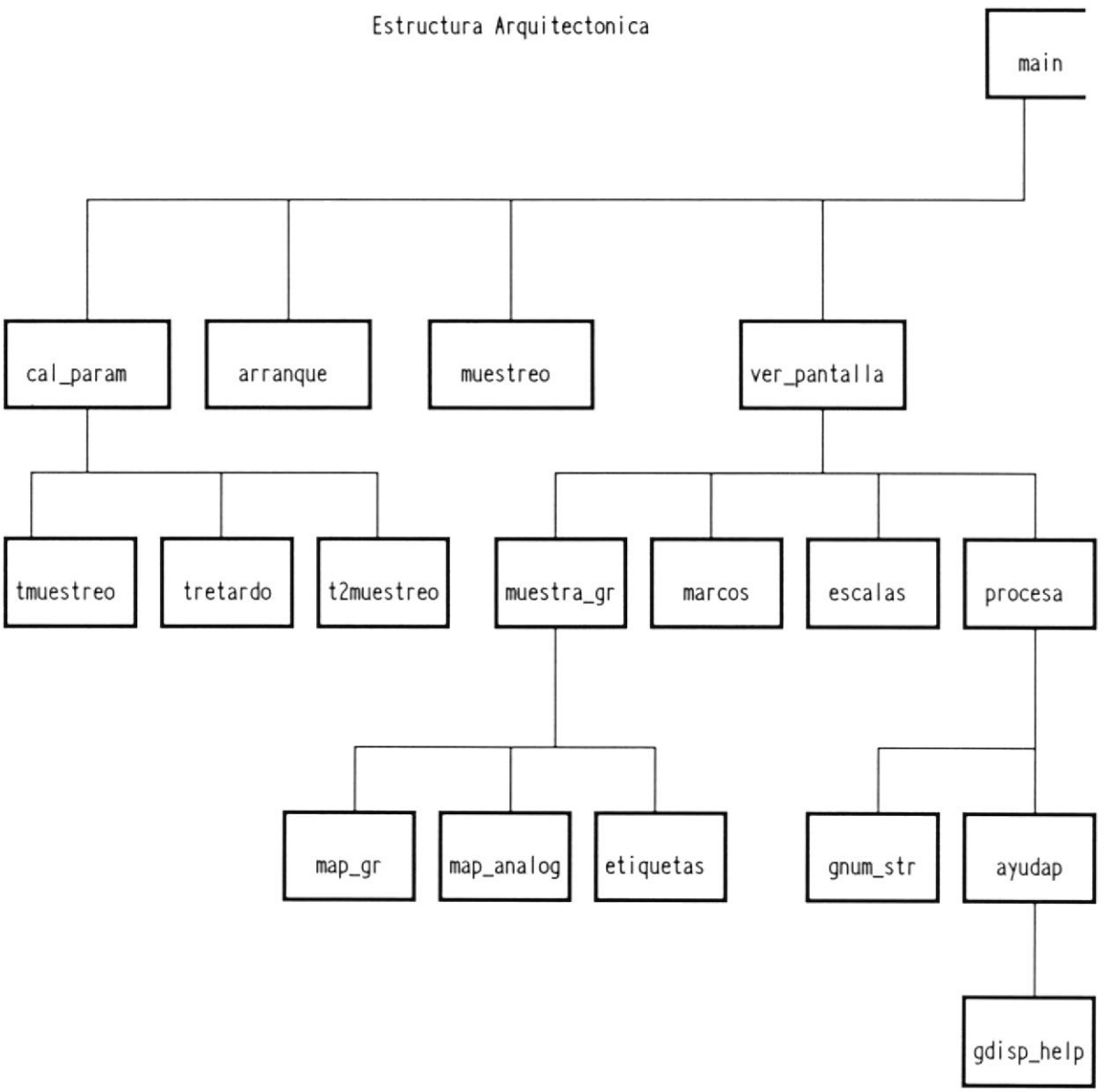

Estructura Arquitectonica
main
cal_param
arranque
muestreo
ver_pantalla
tmuestreo
tretardo
t2muestreo
muestra_gr
marcos
escalas
procesa
map_gr
map_analog
etiquetas
gnum_str
ayudap
gdisp_help

Figura 5.8 (a)

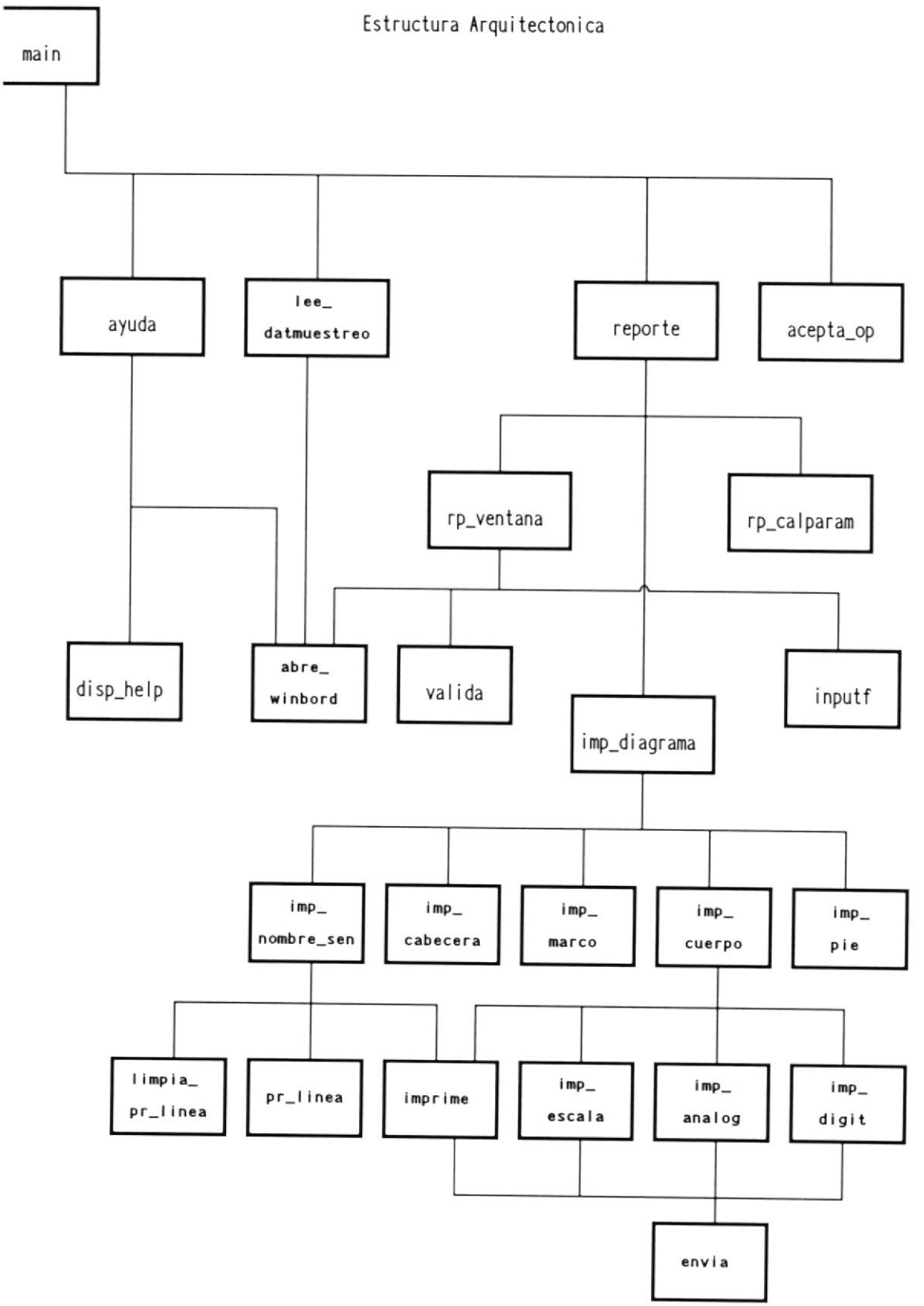

Estructura Arquitectonica
main
ayuda
lee_
datmuestreo
reporte
acepta_op
rp_ventana
rp_calparam
disp_help
abre_
winbord
valida
imp_diagrama
inputf
imp_
nombre_sen
imp_
cabecera
imp_
marco
imp_
cuerpo
imp_
pie
limpia_
pr_linea
pr_linea
imprime
imp_
escala
imp_
analog
imp_
digit
envia

Figura 5.8 (b)

bit tienen el nivel de voltaje de una de las señales en el instante en que se efectuó el muestreo. Por ejemplo.

```
1 0 0 1 0 1 1 0
↑             ↑
D7           D0
```

Representa que :

D0: bajo

D1: alto

D2: alto

D3: bajo

D4: alto

D5: bajo

D6: bajo

D7: alto

En los puertos de lectura de las señales analógicas (0x0342 y 0x0343), el valor leído representa un número que multiplicada por la resolución del convertidor analógico digital da como resultado el nivel de voltaje presente a la entrada del convertidor.

El contenido de cada puerto es almacenado tal como es leído en la memoria del computador, lo que ofrece dos ventajas importantes: No se pierde

tiempo en procesar los datos para ser almacenados, lo cual ayuda a tener muestreos de frecuencias más altas , pues los datos son escritos en memoria tal como son leídos. En segundo lugar, el almacenamiento es compacto, pues cada bit representa un nivel de una señal, así pues en cada byte tendremos el nivel lógico correspondiente a ocho señales.

Los datos correspondientes a cada puerto se almacenan en una estructura de datos semejante a un arreglo de datos de un byte.

Conceptualmente, tenemos 4 arreglos, uno para cada puerto , con 65536 (10000H) elementos tipo byte cada uno. En seudocódigo:

```
A  ARREGLO[0..65535] DE BYTES      Datos del puerto 0340H
B  ARREGLO[0..65535] DE BYTES      Datos del puerto 0341H
C  ARREGLO[0..65535] DE BYTES      Datos del puerto 0342H
D  ARREGLO[0..65535] DE BYTES      Datos del puerto 0343H
```

Esta estructura datos se instrumenta asignando a cada arreglo un segmento propio en la memoria (Un segmento es una porción contigua de memoria de 64 Kb). Por tanto, en cada arreglo se tiene espacio disponible para

64K muestras, es decir 256K (64 x 8) señales.

A Desde 5000:0000 hasta 5000:FFFF

B Desde 6000:0000 hasta 6000:FFFF

C Desde 7000:0000 hasta 7000:FFFF

D Desde 8000:0000 hasta 8000:FFFF

5.3 Diseño Detallado.

5.3.1 Consideraciones del Video.

5.3.1.1 Modos de Video.

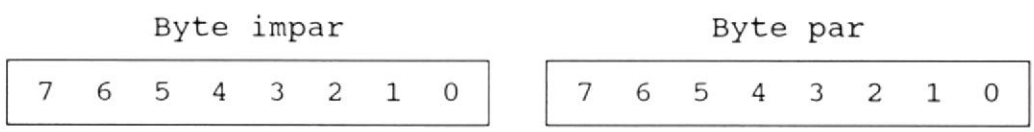
El microcomputador IBM PS/2 Modelo 30 dispone de adaptador gráfico MCGA (Multi Color Graphic Array). Los modos de video con sus características se presentan en la tabla III.

Tabla III

Modo de Video	Formato de Despliegue
Modo 0,1 40 Columnas Alfanumérico	40 columnas x 25 filas caja de caracter de 8x16 320 por 400 16 de 256K colores Display buffer desde B8000 Buffer de video de 2000 bytes
Modo 2,3 80 Columnas Alfanumérico	80 columnas x 25 filas caja de caracter de 8x16 640 por 400 16 de 256K colores Display buffer desde B8000 buffer de video de 4000 bytes
Modo 4,5 320 por 200 Gráfico	caja de caracter de 8x8 Doble llenado 320 por 200 4 de 256K colores Display buffer desde B8000 Buffer de video de 16000 bytes
Modo 6 640 por 200 Gráfico	caja de caracter de 8x8 Doble llenado 640 por 200 2 de 256K colores Display buffer desde B8000 Buffer de video de 16000 bytes
Modo 11 640 por 480 Gráfico	caja de caracter de 8x16 640 por 480 2 de 256K colores Display buffer desde A0000 Buffer de video de 38400 bytes Direccionamiento lineal
Modo 13 320 por 200 Gráfico	caja de caracter de 8x8 Doble llenado 320 por 200 256 de 256K colores Display buffer desde A0000 Buffer de video de 64000 bytes Direccionamiento lineal

5.3.1.2 Formatos de Almacenamiento.

En los modos alfanuméricos (de texto), dos bytes definen cada caracter en la pantalla. El byte par accesa el generador de caracteres para crear los puntos a dibujar. El byte impar define el color de los puntos. Están disponibles 16 colores de caracteres, y ocho colores de fondo cuando el parpadeo (blink) está habilitado. El parpadeo es controlado en el CGA Mode Control register, 03D8H.



Atributo Caracter

Byte de Atributo:

Bits 7-4 : Color de fondo (background)
Bits 3-0 : Color del caracter (foreground)

En los modos 4 y 5, el par de bits C1 y C0 seleccionan uno de 4 colores para cada punto.

En los modos 6 y 11, un bit define cada

Bit	Definición del punto	
7,6	C1, C0	Primer punto
5,4	C1, C0	
3,2	C1, C0	
1,0	C1, C0	Ultimo punto

punto, definiendo el bit más significativo el primer punto.

Bit	Definición del punto	
7	C0	Primer punto
6	C0	
5	C0	
4	C0	
3	C0	
2	C0	
1	C0	
0	C0	Ultimo punto

En modo 13, un punto está definido por todo un byte. Esto permite escoger entre 256 colores para cada punto.

5.3.1.3 Selección del Modo de Video.

Para la graficación de las señales por pantalla necesitamos tener la mayor resolución posible, la cual es ofrecida por el modo de video 11. Además, este modo de video también existe en los

sistemas de video VGA, por lo cual es factible que el programa corra con este adaptador gráfico. La desventaja de este modo es que solo se pueden tener dos colores en pantalla. El mapa de memoria para el almacenamiento del video se presenta en la figura 5.9, donde cada byte define 8 puntos:

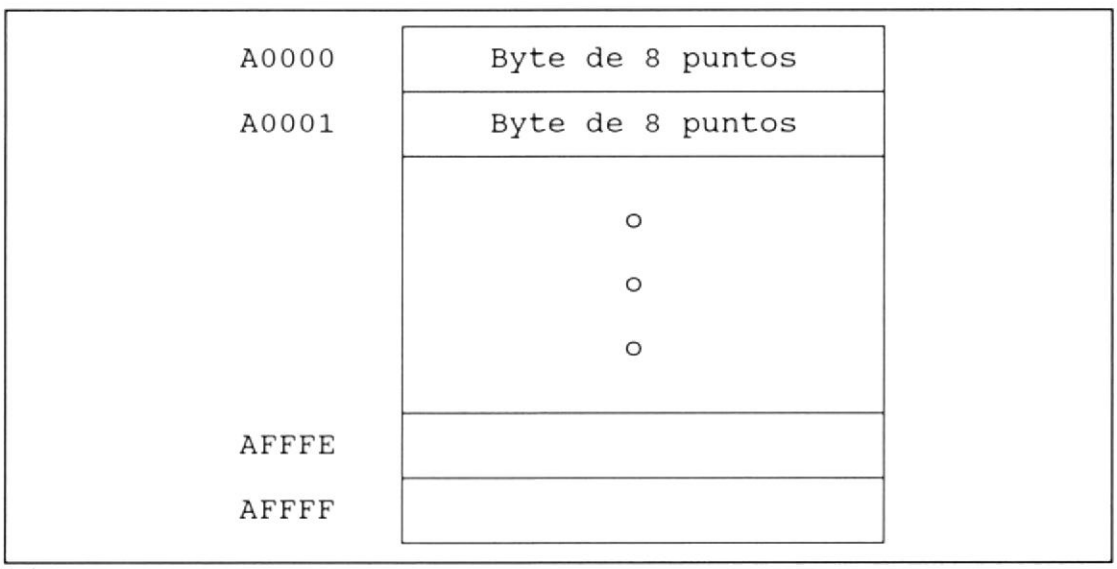


Figura 5.9

5.3.2 Consideraciones del Compilador.

5.3.2.1 Pseudovariables y Ensamblador en línea.

Debido a los requerimientos de velocidad

del software, parte de la programación se la realizará en Ensamblador, por lo tanto es necesario presentar las técnicas provistas por Turbo C++ que serán usadas en la codificación.

El compilador Turbo C++ ofrece una manera sencilla de accesar a los registros del CPU, a través de las pseudovariables. Una pseudovariable es simplemente un identificador que corresponde a un determinado registro. Este puede usarse como si fuera una variable de tipo entero sin signo (unsigned int) o caracter sin signo (unsigned char), según si es un registro de 16 o de 8 bits.

Sin embargo, estas pseudovariables deben de usarse con cuidado. Debido a que el compilador está generando constantemente código que usa los registros, no se tiene una total garantía de que los valores que se almacenen en una pseudovariable se mantengan intactos por un tiempo indefinido. Por ejemplo, no se puede esperar que los valores de

pseudovariables permanezcan sin alterar a través de una llamada a función; no todos los registros son preservados durante una llamada a función. Los únicos registros que tendrán el mismo valor antes y después de una llamada a función son _DS, _BP, _SI y _DI.

El Ensamblador en Línea es una facilidad que brinda Turbo C++ , que consiste en permitir escribir código de Ensamblador dentro del programa fuente en C++. Para introducir código de Ensamblador , solamente se requiere usar la palabra asm antes de la instrucción en lenguaje Ensamblador. La sintaxis es:

---

asm mnemónico operandos; (o nueva línea)

donde

mnemónico es un instrucción válida del 8086

operandos contiene el/los operandos permitidos para el mnemónico, y puede referenciar a variables, constantes y etiquetas de lenguaje C++

; (o nueva línea) es un punto y coma o una nueva línea, cualquiera de las cuales determina el fin del enunciado asm

---

Es posible incluir varias instrucciones de Ensamblador utilizando llaves.

```
asm {
      pop ax; pop ds
      iret
      }
```

5.3.3 Especificación de Algoritmos.

5.3.3.1 Algoritmo del arranque.

La función de arranque tiene la finalidad de esperarar que la señal de arranque alcance el nivel especificado si el modo es automático, pero si el modo es manual la función llega inmediatamente a su fin. Esta función tiene tres parámetros:

MODO : indica si el arranque es manual (0) o automático (1)

BSTART : Es una variable de un byte, donde el enésimo bit es 1 si la enésima señal es la señal de arranque.

NIVEL : Indica si el arraque se produce con la señal en alto (1) o en bajo (0)

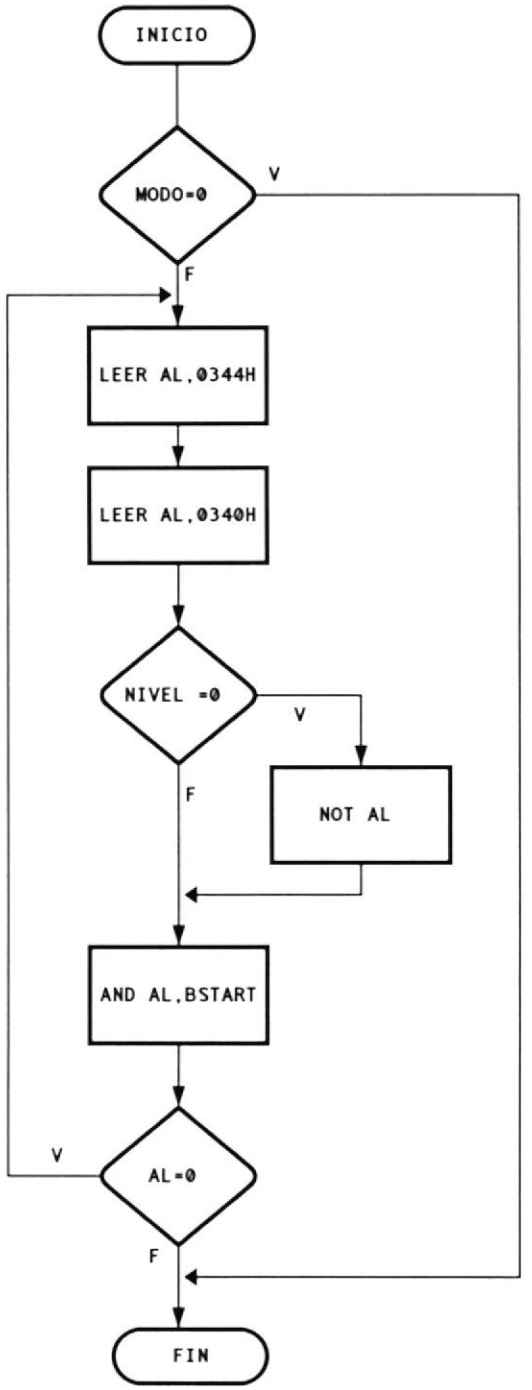

INICIO
MODO=0
V
F
LEER AL,0344H
LEER AL,0340H
NIVEL =0
V
F
NOT AL
AND AL,BSTART
V
AL=0
F
FIN

Figura 5.10

Se lee el byte de las señales en el puerto 0340H, cuando el nivel es 0 se niega el contenido del mismo. Este byte y el byte de arranque son operandos de una instrucción AND.

Ejemplo:

```
MODO    = 1                 BYTE LEIDO = 0100 1010
NIVEL   = 1                              0100 1010
                                    AND
BSTART  = 0000 0010                      0000 0010
                                        -----------
                                         0000 0010
```

El resultado de la operación AND es diferente a 0, por tanto saldrá en este momento del lazo, e inmediatamente comenzará el muestreo. Ver figura 5.10

5.3.3.2 Algoritmo del Muestreo.

El muestreo consiste en enclavar los datos en los puertos de entrada y luego leerlos. La función que realiza el muestreo tiene los siguientes parámetros:

MODO : Si modo es 2, lee los puertos 0340H, 0341H, 0342H y 0343H. Si es 1 , los dos primeros. Si es 0, solo el primero.

RETARDO : Determina el número de iteraciones en el lazo de retardo.

MUESTRAS : Número de muestreos que se producen.

En el diagrama de la figura 5.11 se presenta el algoritmo del muestreo, asumiendo que se están leyendo los cuatro puertos (modo 2). La primera lectura, al puerto 0344H en realidad efectúa el enclavamiento de los datos en los flip flops tipo "D" e inicia la conversión en los convertidores analógico-digitales.

5.3.3.3 Algoritmo del cálculo de parámetros de muestreo.

El objetivo de esta función es calcular el número de muestras y el tiempo de separación entre muestras, en base al tiempo y a la frecuencia de muestreo, además de calcular el valor del retardo (el parámetro retardo de la función de muestreo).

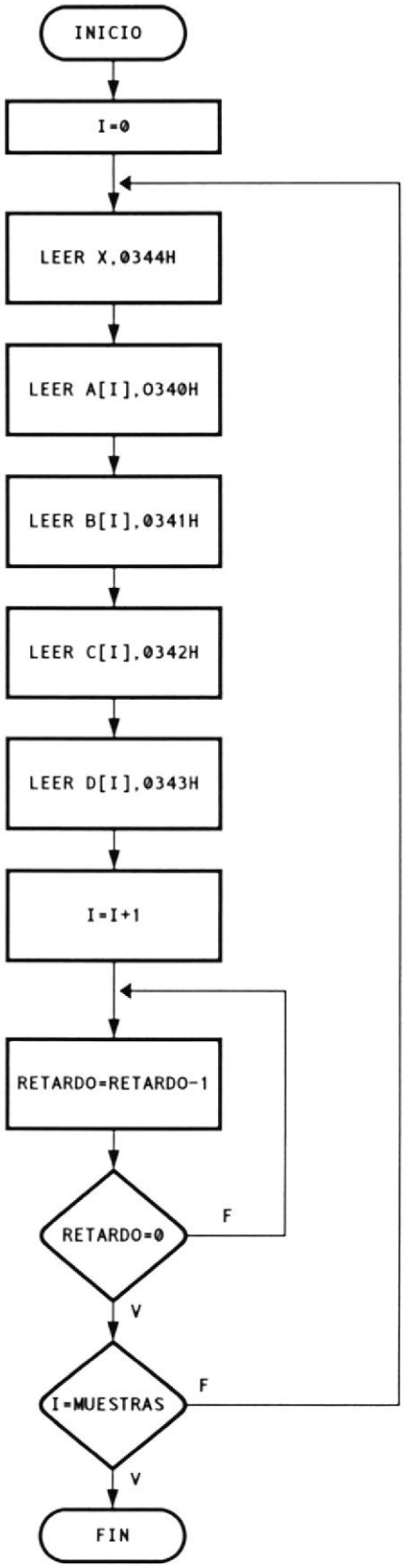

INICIO
I=0
LEER X,0344H
LEER A[I],0340H
LEER B[I],0341H
LEER C[I],0342H
LEER D[I],0343H
I=I+1
RETARDO=RETARDO-1
RETARDO=0
F
V
I=MUESTRAS
F
V
FIN

Figura 5.11

Para calcular el retardo, partimos de la fórmula de la frecuencia de muestreo. El tiempo total para una iteración del lazo de muestreo puede representarse

tm + (tr)(xr)

Donde tr es el tiempo que toma en una iteración en el lazo de retardo y xr es el valor del contador del retardo, y tm es el tiempo que toma una iteración en el lazo de muestreo, pero excluyendo el lazo de retardo. Por esta razón se codificó una función idéntica a la función de muestreo, pero donde se ha suprimido el lazo de retardo, esta función es tmuestreo. Para calcular el tiempo de una iteración en el lazo de retardo, se codificó la función tretardo, que ejecuta la cantidad de veces especificada el lazo de retardo; posteriormente se deberá dividir para este número de veces.

Para poder realizar las mediciones de tiempo, todos estos eventos se han realizado FFFFH veces. Por lo tanto si

$$f1 = FFFFH/frec$$

Entonces

$$f1 = xr.tr + tm$$

y

$$xr = (f1 - tm)/ tr$$

Luego de esto, como ya se tiene el valor del contador de retardo, es posible ejecutar una función similar a la del muestreo, t2muestreo, con una cantidad de muestras tentativa, para encontrar la separación entre muestras, según se observa en la figura 5.12. El cálculo de los tiempos de ejecución de las funciones se lo muestra como una función para facilitar la lectura del diagrama.

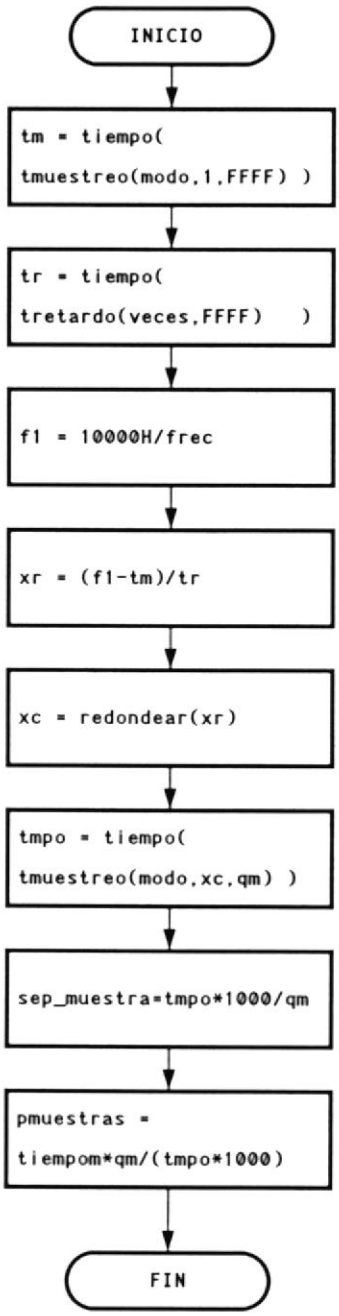

INICIO
tm = tiempo(
tmuestreo(modo,1,FFFF))
tr = tiempo(
tretardo(veces,FFFF))
f1 = 10000H/frec
xr = (f1-tm)/tr
xc = redondear(xr)
tmpo = tiempo(
tmuestreo(modo,xc,qm))
sep_muestra=tmpo*1000/qm
pmuestras =
tiempom*qm/(tmpo*1000)
FIN

Figura 5.12

5.3.4 Especificación de Funciones y Parámetros.

La especificación de las funciones identificadas en el diseño es la siguiente:

```
main()
/*-------------------------------------------------------------------
  Función principal del programa. Llama a la función que muestra el menú
  del programa y que espera por una tecla funcional. Llama a la función
  que corresponde a la opción seleccionada.
-------------------------------------------------------------------*/

acepta_op()
/*-------------------------------------------------------------------
  Muestra el menú principal y espera que se presione una tecla de doble
  código.
  Retorna el segundo código de la tecla presionada.
-------------------------------------------------------------------*/

ayuda()
/*-------------------------------------------------------------------
  Muestra la ventana de ayuda, detecta la tecla presionada por el usuario
  según la cual realiza el  avance o retroceso de página y la salida de
  la ayuda
-------------------------------------------------------------------*/
```

```
disp_help( ysup, lini, lfin)
int ysup, lini, lfin;
/*-------------------------------------------------------------------------
  Muestra el contenido del arreglo de cadenas texto, desde el elemento
  lini hasta el elemento lfin
  PARAMETROS:
  ysup  :   Fila de la ventana actual desde la cual se empiezan a mostrar
            los datos
  lini  :   Elemento inicial a mostrar del arreglo
  lfin  :   Ultimo elemento a mostrar del arreglo
-------------------------------------------------------------------------*/

llena_str(p,c,n)
unsigned char *p,c;
int n;
/*-------------------------------------------------------------------------
  Crea una cadena formada por n veces el caracter c
  PARAMETROS:
  p : Una expresión que se evalúe en la dirección de la cadena
  c : Caracter que se repetir en la cadena
  n : Numero de repeticiones del caracter anterior
-------------------------------------------------------------------------*/
```

```
abre_winbord( xis, yis, xdi, ydi )
int xis, yis, xdi, ydi;
/*-----------------------------------------------------------------------
  Abre una ventana con marco
  PARAMETROS:
  xis   : coordenada x de la esquina superior izquierda
  yis   : coordenada y de la esquina superior izquierda
  xdi   : coordenada x de la esquina inferior derecha
  xyi   : coordenada y de la esquina inferior derecha
-----------------------------------------------------------------------*/

lee_datmuestreo(pmodo,parranque,psenal, plogica, pfrec, ptiempo)
int *pmodo;
unsigned long *pfrec, *ptiempo;
unsigned char *psenal,*parranque,*plogica;
/* ----------------------------------------------------------------------
  Muestra la ventana de Asignación de parámetros de muestreo, y hace el
  requerimiento al usuario de estos parámetros
  PARAMETROS:
  pmodo     : modo de muestreo
  parranque : modo de arranque
  psenal    : señal de arranque
  plogica   : nivel logico de la señal de arranque
  pfrec     : frecuencia de muestreo
  ptiempo   : tiempo de muestreo
-----------------------------------------------------------------------*/
```

```
clrfield(x,y,xclr)
int x,y,xclr;
/*-------------------------------------------------------------------------
  Limpia parte de una línea.
  PARAMEIROS:
  x          : columna inicial de la línea
  y          : numero de la línea
  xclr       : columna final de la línea
-------------------------------------------------------------------------*/

cal_param(modo,frec,tiempo,pmuestras,sep_muestra,falla)
unsigned long frec,tiempo;
unsigned int *pmuestras;
double *sep_muestra;
char *falla;
/*-------------------------------------------------------------------------
  Tiene como objetivo obtener el valor inicial del contador de retardo,
  el numero de muestras que corresponde al tiempo especificado y el
  tiempo en milisegundos de separación entre dos muestras sucesivas.
  PARAMEIROS:
  modo       : En modo 2 el calculo debe realizarse efectuando la lectura
               de 4 puertos, en modo 1 con la lectura de 2 puertos y en
               modo 0 efectuando la lectura de un solo puerto
  frec       : Frecuencia de muestreo
  tiempo     : Tiempo de muestreo
  pmuestras  : Retorna el numero de muestreos que hay que efectuar
```

```
  sep_muestra : Retorna el tiempo de separación entre muestras en miliseg.
  falla       : Retorna 0 si la frecuencia es demasiado baja, 1 si es dema-
                siado alta
 Retorna el valor del contador para el retardo en la función de muestreo
-------------------------------------------------------------------------*/

tmuestreo(modo,retardo,muestras)
unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Función que realiza el muestreo, pero sin el lazo de retardo.
  PARAMETROS:
  modo        : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
                primeros y en modo 0 solo el puerto 0x0340
  retardo     : Es usado para que esta función guarde similitud con la fun-
                ción muestreo
  muestras    : Un numero de 2 bytes que especifica el numero de lecturas
                que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/

tretardo(veces,retardo)
unsigned int veces,retardo;
/*-------------------------------------------------------------------------
  Función que se usa en el calculo de parámetros de muestreo. Representa el
  tiempo que toma ejecutar repetidas veces el lazo de retardo.
  PARAMETROS:
  veces       : Numero de veces que se ejecuta un lazo exterior
```

```
  retardo    : Número de veces que se ejecuta el lazo de retardo
-------------------------------------------------------------------------*/

t2muestreo(modo,retardo,muestras)
unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Función que se usa en el calculo de parámetros de muestreo. Representa
  el tiempo que toma realizar un determinado numero de muestras con un
  retardo especificado por la variable retardo.
  PARAMETROS:
  modo       : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
               primeros y en modo 0 solo el puerto 0x0340
  retardo    : Un numero de 2 bytes. Se usa con un lazo, para provocar un
               retardo entre muestras sucesivas
  muestras   : Un numero de 2 bytes que especifica el numero de lecturas
               que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/

muestreo(modo,retardo,muestras)
unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Realiza el muestreo de las señales que se leen por los puertos
  0x0340, 0x0341, 0x0342, 0x0342
  PARAMETROS:
  modo       : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
               primeros y en modo 0 solo el puerto 0x0340
```

```
  retardo    : Un numero de 2 bytes. Se usa con un lazo, para provocar un
               retardo entre muestras sucesivas
  muestras   : Un numero de 2 bytes que especifica el numero de lecturas
               que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/

arranque(modo,bstart,nivel)
unsigned char modo,bstart,nivel;
/*-------------------------------------------------------------------------
  Determina el momento en que se debe comenzar el muestreo, dependiendo de
  sus parámetros.
  PARAMETROS:
  modo    : Si es 0, arranca inmediatamente. Si es 1, arranca cuando la
            señal de arranque tiene el nivel especificado.
  bstart  : Byte de arranque. El bit n se encuentra seteado si la señal n
            es la señal de arranque.
  nivel   : Nivel de la señal de arranque. 1 = alto, 0 = bajo.
-------------------------------------------------------------------------*/

ver_pantalla(separ,v_modo, muestras)
double separ;
int v_modo;
unsigned int muestras;
/*-------------------------------------------------------------------------
  Maneja la pantalla de visualización de señales. Presenta el menú de esta
  pantalla y espera por una tecla funcional. Llama a las funciones de
```

```
  graficación
  PARAMETROS:
  separ      : tiempo de separación entre muestras sucesivas en miliseg.
  v_modo     : modo de muestreo (8, 16 o 18 señales)
  muestras   : Numero de muestreos realizados
-------------------------------------------------------------------------*/

marcos()
/*-------------------------------------------------------------------------
 dibuja el marco de la pantalla
-------------------------------------------------------------------------*/

escalas(separ)
double separ;
/*-------------------------------------------------------------------------
  Presenta las coordenadas de tiempo en la pantalla de visualización de
  señales
  PARAMETROS:
  separ    : tiempo de separación entre muestras en milisegundos
-------------------------------------------------------------------------*/

etiquetas(t,p)
int t,p ;
/*-------------------------------------------------------------------------
  Escribe los nombres de las señales que se presentan en pantalla
  PARAMETROS:
```

```
  t    : 1 si se presentan los nombres de las señales D0 a D7
         2 si se presentan los nombres de las señales D7 a D15
         3 si se presentan los nombres de las señales A0 y A1
  p    : controla la posición vertical a partir de la cual se muestran los
         nombres de las señales
-------------------------------------------------------------------------*/

muestragr(h,l)
unsigned int h,l;
/*-------------------------------------------------------------------------
  Muestra las señales leídas en la pantalla de visualización
  PARAMEIROS
  h    : Determina que bloque de señales se presenta en la zona superior de
         la pantalla
  l    : Determina que bloque de señales se presenta en la zona inferior de
         la pantalla
-------------------------------------------------------------------------*/

procesa(c1,c2,separ)
unsigned char c1,c2;
double separ;
/*-------------------------------------------------------------------------
  Ejecuta una de las opciones del menú de la pantalla de visualización de
  señales, según la tecla funcional presionada
  PARAMEIROS:
  c1   : Primer código de la tecla presionada
```

```
  c2    : Segundo código de la tecla presionada
  RETORNA:
  Si se presiona F10 retorna 0, de otra modo retorna 1
-------------------------------------------------------------------------*/

mapanalog(p,lin0)
unsigned char huge *p;
unsigned int lin0;
/*-------------------------------------------------------------------------
  Grafica una señal analógica
  PARAMETROS
  p     : Puntero a la dirección de memoria del inicio de los datos para la
          porción a graficar
  lin0 : Determina la posición vertical en la pantalla a partir de la cual
          se grafica
-------------------------------------------------------------------------*/

mapgr( p, l)
unsigned char huge *p;
int l;
/*-------------------------------------------------------------------------
  Grafica un bloque de 8 señales digitales.
  PARAMETROS:
  p     : Puntero a la dirección de memoria del inicio de los datos para la
          porción a graficar
  l     : determina la posición en la pantalla a partir de la cual se
```

```
        grafica
-------------------------------------------------------------------------*/

ayudap()
/*-------------------------------------------------------------------------
  Muestra una ventana de ayuda para la pantalla de visualización de
  señales. Efectúa el avance/retroceso de pantalla
-------------------------------------------------------------------------*/

gdisp_help( ysup, lini, lfin)
int ysup, lini, lfin;
/*-------------------------------------------------------------------------
  muestra en modo gráfico el contenido del arreglo de cadenas texto, desde
  el elemento  lini hasta el elemento lfin
  PARAMETROS:
  ysup      : línea de posición vertical inicial
  lini      : elemento inicial a mostrar
  lifi      : elemento final a mostrar
-------------------------------------------------------------------------*/

reporte(modo,paso,dmarco)
int modo,dmarco;
double paso;
/*-------------------------------------------------------------------------
  Llama a las funciones de generación de reporte: ventana de asignación de
  datos para reporte, calculo de parametros e impresión de reporte
```

```
  PARAMETROS:
  modo       : modo de muestreo:0=8 señales,1=16 señales,2=18 señales
  paso       : separación en milisegundos entre dos muestras sucesivas
  dmarco     : posición del marco derecho (en puntos de impresión )
-------------------------------------------------------------------------*/

rp_ventana(modo,tini,tfin,tit)
int modo;
double *tini, *tfin;
char *tit;
/*-------------------------------------------------------------------------
  Abre la ventana de asignación de datos de muestreo, controla el ingreso
  de los campos de acuerdo al modo especificado
  Retorna el código de la tecla ESCAPE o de F10
  PARAMETROS:
  modo  : modo de muestreo, como en la función reporte
  tini  : retorna el tiempo a partir del cual se imprimirá el reporte
  tfin  : retorna el tiempo hasta el cual se imprimirá el reporte
  tit   : retorna el titulo del reporte
  también manipula los objetos globales existe_sen e id_sen
-------------------------------------------------------------------------*/

valida(i)
int i;
/*-------------------------------------------------------------------------
Retorna 1 si datos[i] es una cadena numérica o nulo, de otra modo retorna
```

```
0
PARAMETROS:
i  : índice que corresponde al elemento del arreglo datos que se validará
-------------------------------------------------------------------------*/

inputf(i)
int i;
/*-------------------------------------------------------------------------
  Lee una tecla del teclado y retorna su código. Si la tecla es ENTER,
  lee el campo i del formulario de datos para impresión del reporte.
  Retorna el código de la tecla presionada antes de ingresar el campo.
  PARAMETROS:
  i  : Especifica el campo que se leerá
-------------------------------------------------------------------------*/

rp_calparam( imarco, dmarco, a_num, d_num, a_amp, d_amp)
int imarco, dmarco, *a_num, *d_num, *a_amp, *d_amp ;
/*-------------------------------------------------------------------------
  Obtiene el numero de señales analógicas y de señales digitales a
  imprimir, la amplitud con que serán impresas las señales analógicas y
  digitales en el reporte. También manipula los objetos globales existe_sen
  y org_sen.
  PARAMETROS:
  imarco     : distancia del marco izquierdo, en puntos de impresión
  dmarco     : distancia del marco derecho, en puntos de impresión
  anum       : numero de señales analógicas
```

```
  dnum      : numero de señales digitales
  a_amp     : amplitud de impresión de las señales analógicas
  d_amp     : amplitud de impresión de las señales analógicas
-------------------------------------------------------------------------*/

imp_diagrama(tini,tfin,pini,pfin,paso,a_amp, d_amp,rp_titulo)
double tini,tfin,paso;
unsigned int pini,pfin;
int a_amp, d_amp;
char *rp_titulo;
/*-------------------------------------------------------------------------
  Función que imprime el diagrama de tiempo según lo especifican los para-
  metros
  PARAMETROS:
  tini      : tiempo inicial en milisegundos
  tfin      : tiempo final en milisegundos
  pini      : numero de la muestra inicial a imprimir
  pfin      : numero de la muestra final a imprimir
  paso      : separación entre dos muestras sucesivas, en milisegundos
  a_amp     : amplitud máxima de las señales analógicas
  d_amp     : amplitud máxima de las señales digitales
  rp_titulo : titulo del reporte
-------------------------------------------------------------------------*/
```

```
imp_cabecera(pag,rp_titulo)
unsigned int pag;
char *rp_titulo;
/*-------------------------------------------------------------------------
  Imprime la cabecera de página, que contiene el titulo y numero de página
  del reporte.
  PARAMETROS:
  pag       : numero de página actual
  rp_titulo : titulo del diagrama de tiempo
-------------------------------------------------------------------------*/

imp_marco(tipo,a_amp,d_amp)
unsigned char tipo;
int a_amp, d_amp;
/*-------------------------------------------------------------------------
  Imprime el marco superior o inferior de una página del reporte
  PARAMETROS:
  tipo      : Es el patrón gráfico a imprimir
  a_amp     : amplitud máxima de las señales analógicas
  d_amp     : amplitud máxima de las señales digitales
-------------------------------------------------------------------------*/

imp_nombre_sen(s_tini,a_amp,d_amp)
char *s_tini;
int a_amp, d_amp;
```

```
/*-------------------------------------------------------------------------
  Imprime las etiquetas que identifican a las señales que se van a presen-
  tar en el diagrama de tiempo
  PARAMETROS:
  s_tini     : Una cadena que contiene la representación en caracteres del
               tiempo inicial
  a_amp      : amplitud máxima de las señales analógicas
  d_amp      : amplitud máxima de las señales digitales
-------------------------------------------------------------------------*/

imp_pie(paso)
double paso;
/*-------------------------------------------------------------------------
  Imprime los pies de página del diagrama de tiempo
  PARAMETROS:
  paso       : separación entre dos muestra sucesivas
-------------------------------------------------------------------------*/

imp_cuerpo(linxpag, pactual, pfin, a_amp, d_amp)
unsigned int linxpag, *pactual, pfin;
int  a_amp, d_amp;
/*-------------------------------------------------------------------------
  Imprime el cuerpo del reporte para cada página del diagrama de tiempo.
  Retorna 1 si llega al fin del reporte, de otro modo retorna 0.
  PARAMETROS:
  linxpag    : Numero de línea gráficas por página
```

```
  pactual    : Numero de la muestra actual
  pfin       : Numero de la muestra final
  a_amp      : amplitud máxima de las señales analógicas
  d_amp      : amplitud máxima de las señales digitales
-------------------------------------------------------------------------*/

imp_escala(lin)
/*-------------------------------------------------------------------------
  Imprime una división o indicador de coordenadas del tiempo para el cuerpo
  del diagrama de tiempo.
  PARAMETROS:
  lin              : línea gráfica actual
-------------------------------------------------------------------------*/

imp_digit(prel,lin,d_amp)
unsigned int prel, lin, d_amp;
/*-------------------------------------------------------------------------
  Escribe las señales digitales
  PARAMETROS:
  prel       : Numero de la muestra actual a imprimir
  lin        : línea gráfica actual
  d_amp      : amplitud máxima de las señales digitales
-------------------------------------------------------------------------*/
```

```
imp_analog(prel,lin,a_amp,frac)
unsigned int prel, lin, a_amp, frac;
/*-----------------------------------------------------------------------
  Escribe las señales analógicas
  PARAMETROS:
  prel      : Numero de la muestra actual a imprimir
  lin       : línea gráfica actual
  a_amp     : amplitud máxima de las señales analógicas
  frac      : fracción de la amplitud original
-----------------------------------------------------------------------*/

envia(col,lin,dat)
unsigned int col, lin;
unsigned char dat;
/*-----------------------------------------------------------------------
  Escribe en el arreglo prlinea, que es mandado a la impresora por la
  función imprime
  PARAMETROS:
  col       : columna de impresión en el modo gráfico
  lin       : línea gráfica actual
  dat       : dato a escribir
-----------------------------------------------------------------------*/
```

```
imprime()
/*-------------------------------------------------------------------------
  Recorre el arreglo prlinea, que es un arreglo de bytes, y va poniendo en
  la impresora cada elemento
-------------------------------------------------------------------------*/
```

CAPÍTULO VI

MANUAL DE OPERACIÓN.

6.1 Instalación del Sistema de Generación de Diagramas de Tiempo.

Antes de iniciar la operación del sistema es necesario instalar los dispositivos que son utilizados para la recepción de las señales eléctricas. Los componentes que se deben poseer son: la tarjeta del circuito de adquisición de datos, la Caja de Conecciones y el cable que conecta la Caja de Conecciones a la tarjeta. Para instalar estos dispositivos, deberá seguir estas instrucciones:

- Retire la cubierta del computador (ver manual de operaciones del computador)
- Inserte la tarjeta del circuito de adquisición de datos en una de las ranuras del computador que esté libre. Para mayor información sobre como instalar una tarjeta, ver manual de operaciones del computador.
- Conectar el cable a la tarjeta.

- Conectar la Caja de Conecciones con el otro lado del cable.

Por otra parte, el programa puede copiarse en cualquier directorio del disco duro, pero en este directorio deberá tambien copiarse el archivo CGA.BGI que se encuentra junto con el código ejecutable del programa en el mismo disco removible (diskette).

6.2 Explicación de Menúes y Opciones.

6.2.1 El Menú Principal

El menu pricipal se observa en la figura 6.1. Tiene las siguientes opciones:

F1: Ayuda: Se muestra una ventana con información sobre el uso del programa. Las teclas Page-Up/Page-Down proveen de las funciones de Avance/Retroceso de página. Ver la Figura 6.2

F2: Muestreo : Se muestra la ventana de Asignación de parámetros de Muestreo. Se explica con detalle en la sección 6.2.2

F3: Arranque : En modo de muestreo manual, al

Figura 6.1

presionar la tecla F3 se inicia inmediatamente el muestreo. en modo automático, el muestreo comenzará cuando la señal de arranque alcance el nivel especificado

F4: Pantalla: Se muestra la pantalla de Visualización de señales.

F5: Reporte: Se muestra la ventana para la especificación de datos para la generación de la salida impresa.

F10:Salir: Termina el programa y retorna al Sistema Operativo

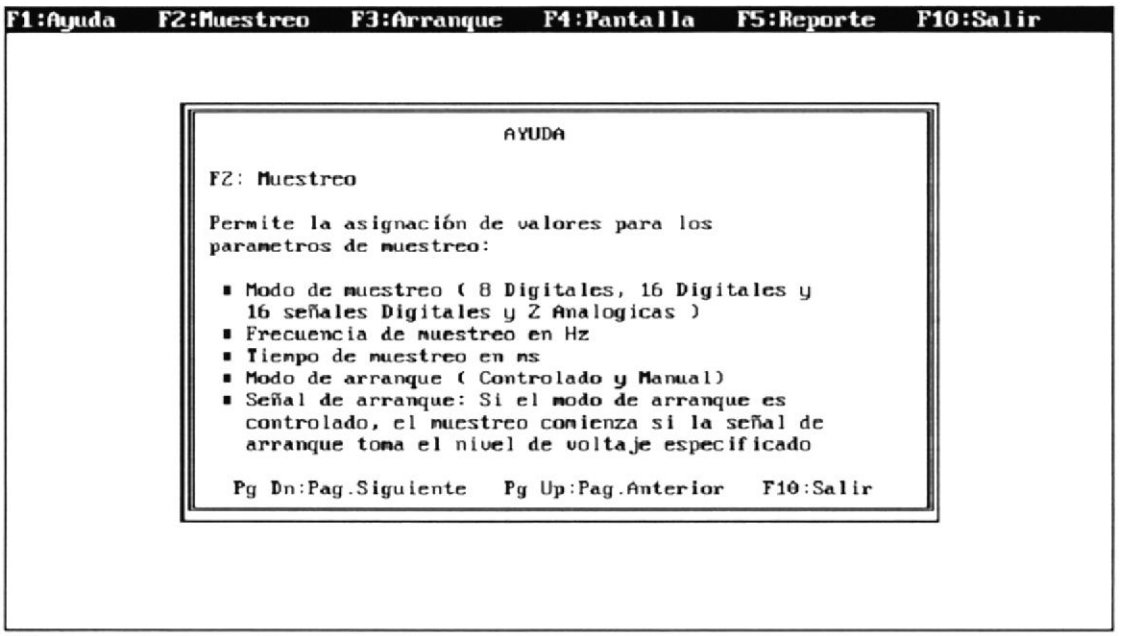


Figura 6.2

6.2.2 Asignacion de Parámetros de muestreo.

Por medio de esta opción se definen los datos que requiere el programa para controlar la manera como se produce el muestreo (Figura 6.3) Estos datos son :

Modo de muestreo: Presionando la tecla Enter, el modo de muestreo se conmuta entre los tres posibles modos que son: 8 señales digitales, 16 señales digitales y 16 señales digitales mas dos analógicas.

Frecuencia de muestreo: La frecuencia en Hz a la

que las señales presentes en la entrada son enclavadas para posterior lectura.

Tiempo de muestreo: El tiempo en milisegundos que durará el mustreo de las señales

Modo de Arranque: Presionando Enter, el modo de arranque se conmuta entre Controlado o Manual. En modo Manual, el muestreo comienza inmediatamente después de presionada la tecla de arranque. En modo Controlado ,después de presionar la tecla de arranque , el muestreo comenzará cuando la señal de arranque alcance el nivel especificado.

Señal y nivel de arranque: Presionando la tecla Enter, permite seleccionar la señal y el nivel de arranque deseados.Estos datos deben ser especificados en caso de que el modo de arranque sea Controlado.

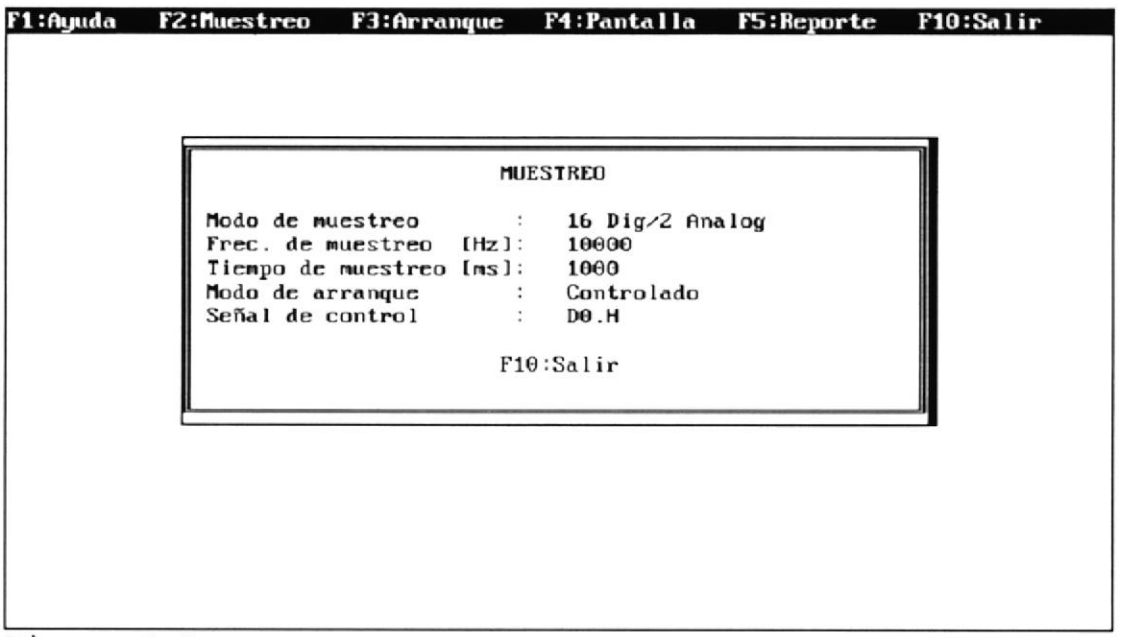


Figura 6.3

La tecla F10 finaliza el ingreso de los datos de muestreo; en ese momento se muestra el mensaje "Calculando parámetros de muestreo", que indica que el computador está procesando estos datos para poder obtener parámetros internos que son necesarios para el muestreo.

6.2.3 Menú de la Pantalla de Visualización de señales.

El menú de la pantalla de visualizacion de señales se observa en la figura 6.4. Tiene las siguientes opciones:

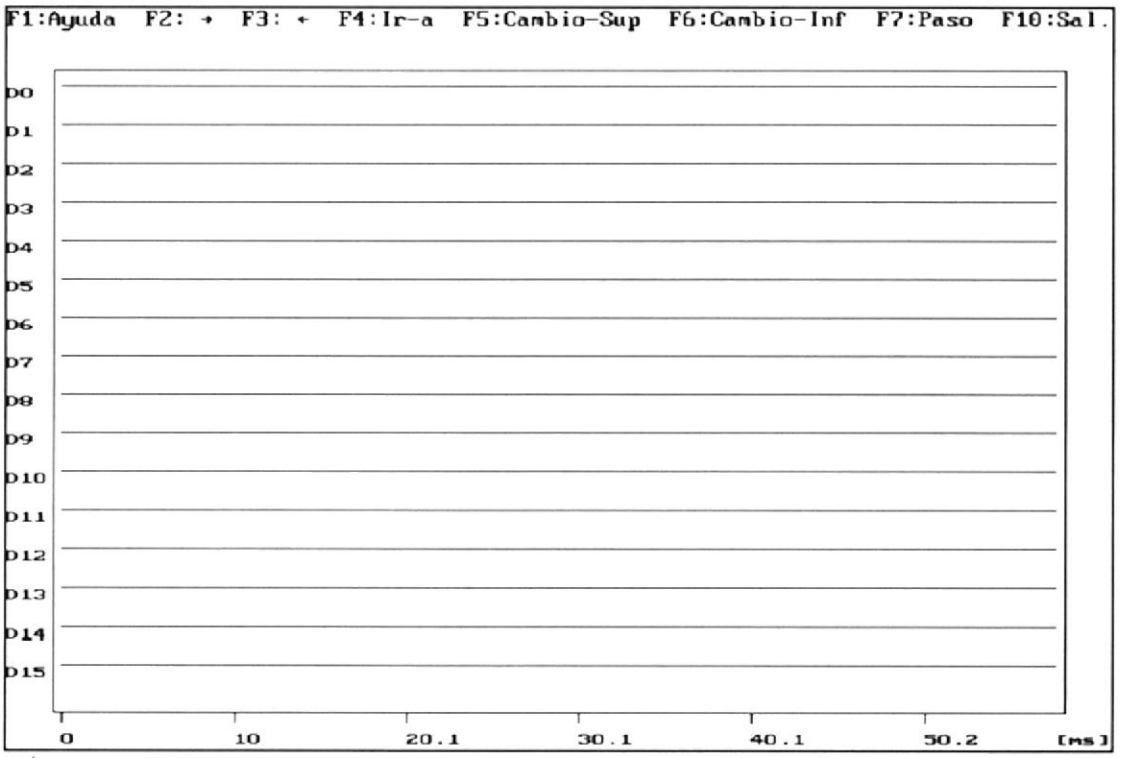


Figura 6.4

F1: Ayuda : Se presenta una ventana de ayuda sobre el uso de la pantalla de visualizacion de datos. Las teclas Pg.Up y Pg.Down permiten hacer desplazamiento hacia arriba o hacia abajo. La tecla F10 cierra la ventana de ayuda. Ver figura 6.5.

F2: → : Sirve para desplazarse horizontalmente avanzando en el tiempo.

F3: ← : Sirve para desplazarse horizontalmente retroce-diendo en el tiempo.

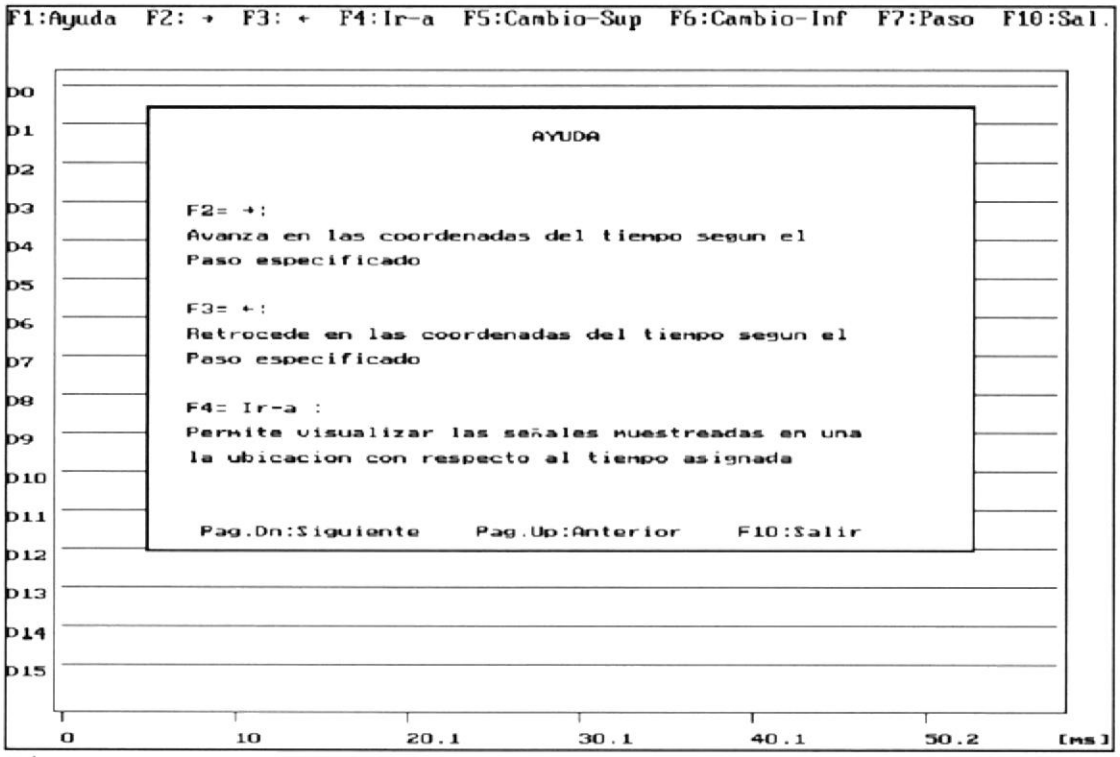


Figura 6.5

F4: Ir-a : Permite especificar una coordenada en el tiempo y luego desplaza la pantalla hasta esta coordenada.

F5: Cambio-Sup: Permite conmutar la zona superior de la campaña con otro conjunto de señales, en el modo de 16 señales analógicas y 2 digitales.

F6: Cambio-Inf: Permite conmutar la zona inferior de la campaña con otro conjunto de señales, en el modo de 16 señales analógicas y 2 digitales.

F7: Paso : Mediante esta opción se podrá

especificar otro valor para el paso, es decir la cantidad de milisegundos que avanza o retrocede la pantalla al presionar F2 o F3.

F10: Salir : Retorna al menú principal.

6.3 Conección de un Circuito Digital al Circuito de Adquisición de Datos

Previamente a la conección de un circuito digital, debe asegurarse que:

Se encuentre conectada la tarjeta del Circuito de Adquisición de Datos.
La Caja de Conecciones se encuentre conectada a la tarjeta a través del cable.

La conección de las señales se realiza a través de la Caja de Conecciones. Esta ofrece puntos para la conección de 16 señales digitales y de 2 señales analógicas con voltajes entre 0 y +5V. Además debe conectarse la tierra del circuito que se analizará a uno de los puntos de tierra de la Caja de Conecciones.

Advertencia: Los puntos que no estén conectados pueden recibir influencia de ruido del ambiente, por tanto en la

gráfica se notará esta influencia.

6.4 Uso de la Pantalla de Visualización de Señales.

La pantalla de visualizacion de señales permite observar las señales muestreadas por todo el intervalo de muestreo. En la pantalla se presenta una zona que es parte del total muestreado. Se puede avanzar o retroceder en el tiempo desplazando la pantalla en dirección horizontal en uno u otro sentido.

Podemos observar la Pantalla de Visualización de señales en la Figura 6.4. En la parte superior de la pantalla se ubicará una línea donde se presentaran las acciones posibles a manera de menú horizontal. Estas opciones son seleccionadas mediante las teclas funcionales especificadas.

La pantalla se divide en dos zonas de presentación de datos: la superior y la inferior.

En el modo de muestreo de 16 señales digitales y 2 analógicas, en cualquiera de estas zonas pueden mostrarse: las señales digitales D0 a D7, las señales digitales D8 a D15 o las señales analógicas A0 y A1. Las opciones del menú permiten conmutar una zona de un

grupo de señales a otro. La opción "Cambio Superior" conmuta la zona superior y la opción "Cambio Inferior" la otra zona. En el modo de 16 o de 8 señales digitales no es necesaria la conmutación , pues pueden verse simultáneamente todas las señales muestreadas.

En la parte inferior de la pantalla se presentan las coordenadas del tiempo a que corresponden los datos que se están mostrando. Los valores están en milisegundos. Se puede avanzar o retroceder en el tiempo mediante la opción Avanzar (F2) o Retroceder (F3) respectivamente. El paso del avance o retroceso puede modificarse usando la opción Paso (F7), que pedirá un nuevo valor de paso en milisegundos. La opción Ir-a (F4) permite que el usuario digite una ubicación en el tiempo (en milisegundos) , y luego desplaza la pantalla hasta esta posición.

6.5 Obtención de las Salidas Impresas.

El sistema permite imprimir los diagramas de tiempo obtenidos brindando la facilidad de definir el intervalo de muestreo y qué señales se van a imprimir (Fig. 6.6). Más detalladamente, los datos que se requieren para imprimir el reporte son:

Titulo : Una cadena de caracteres que se imprimirá en la cabecera del reporte, a manra de título.

Tiempo inicial : Indica desde que instante de tiempo serán mostradas las señales en el reporte.

Tiempo final : Indica hasta que tiempo serán mostradas las señales en el reporte.

Nombres de las Señales : En el formulario aparecerán todas las señales muestreadas, cuya cantidad varía dependiendo del modo de muestreo. El usuario deberá asignar un nombre a cada señal que desee mostrar en el diagrama de tiempo. Si no desea mostrar una señal no deberá asignarle ningún nombre.

El usuario puede desplazarse de un campo a otro con las flechas del teclado. La tecla ESC inicia la impresión

del reporte y la tecla F10 retorna al menú principal. Ver Figura 6.6.

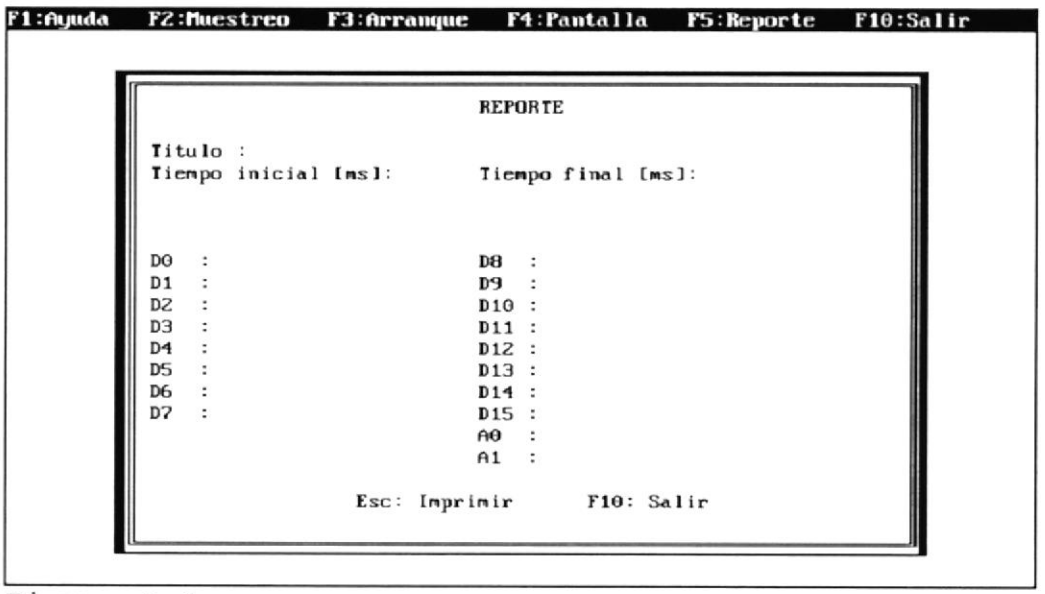


Figura 6.6

CONCLUSIONES Y RECOMENDACIONES.

Mientras se puedan alcanzar más altas frecuencias de muestreo, se logrará un mayor exito al usar un computador para graficar señales eléctricas. La presente tesis tuvo como una de sus limitaciones el que fue desarrollada para un computador muy lento; computadores de bajo precio facilmente son seis u ocho veces más rápidos, ni que decir de los computadores personales más veloces, como los de 33 o 50 MHz, con los que no habría mucha dificultad en llegar a frecuencias de muestreo de más de de 1 MHz.

Por otra parte, el convertidor analógico digital usado no permite operar a frecuencias mayores a 500 KHz, por lo cual si se desea aumentar la frecuencia más allá de este límite se tendría que escoger otro convertidor.

Podemos decir que sistemas como este pueden ser una alternativa poco costosa cuando existan necesidades como monitoreo de señales o prueba de circuitos y se disponga de una computadora.

En general, se debería incentivar el desarrollo de más proyectos que busquen encontrar aplicaciones industriales al computador, porque en muchos casos el uso de un computador es la alternativa más económica.

APENDICE A

CODIGO FUENTE DE LOS MODULOS DEL SISTEMA DE GENERACION DE DIAGRAMAS DE TIEMPO

```
/**********************
MODULO:princip
**********************/

#define LINEA_BLANCO  "                                                    \
      "
#include <conio.h>;


main()
/*-------------------------------------------------------------------------
  Funcion principal del programa. Llama a la funcion que muestra el menu
  del programa y que espera por una tecla funcional. Llama a la funcion
   que corresponde a la opcion seleccionada.
-------------------------------------------------------------------------*/
{
int op, modo=2;
unsigned long frec=10000,tiempo=1000;
unsigned int muestras,retardo;
unsigned char marranque=1, barranque,logica=1,senal=0;
char falla;
double separ;
/*
  printf( "CS: %x  DS: %x  SS: %x  ES: %x \n",_CS,_DS,_SS,_ES);
  getch();
*/
  textmode(C80);
  textattr(0x07);
  clrscr();
  textattr(0x71);
  abre_winbord(20,8,60,17);
  gotoxy(10,3);
  cprintf("SISTEMA DE GENERACION DE\n");
  gotoxy(10,4);
  cprintf("  DIAGRAMAS DE TIEMPO");
  gotoxy(9,6);
  cprintf("Wellington Cabrera Arévalo");
  gotoxy(16,8);
  cprintf("ESPOL, 1992");
  textmode(C80);
  do
    {
     op = acepta_op();
     gotoxy(5,25);
     textmode(C80);
     textattr(0x07);
     cprintf(LINEA_BLANCO);
      gotoxy(5,24);
      switch (op)
      {
         case 1:
                textattr(0x71);
```

```
                ayuda();
                textmode(C80);
                textattr(0x07);
                break;
        case 2:
                lee_datmuestreo(&modo,&marranque,&senal,&logica,&frec,&tiempo);
                textmode(C80);
                textattr(0x07);
                gotoxy(5,25);
                cprintf("Calculando parametros de muestreo ...");
                retardo = cal_param(modo,frec,tiempo,&muestras,&separ,&falla);
                gotoxy(5,25);
                if (retardo==0)
                   if (falla==0)
                     cprintf("La frecuencia seleccionada es demasiado baja");
                   else
                     cprintf("Por favor, seleccione una frecuencia mas baja");
                gotoxy(5,25);
                break;
        case 3:
                barranque = 1;
                barranque = barranque << senal;
                arranque(marranque,barranque,logica);
                muestreo(modo,retardo, muestras);
                break;
        case 4:
                ver_pantalla(separ,modo,muestras);
                break;
        case 5:
                reporte(modo,separ,950);
                textmode(C80);
                textattr(0x07);
                break;
     }
   }
   while (op != 10 );
}


acepta_op()
/*-------------------------------------------------------------------------
  Muestra el menu principal y espera que se presione una tecla de doble
  codigo.
  Retorna el segundo codigo de la tecla presionada.
-------------------------------------------------------------------------*/
{
 unsigned char c1,c2;
 int  c;

 gotoxy(1,1);
 textattr(0x71);
 cprintf(
```

```
"F1:Ayuda   F2:Muestreo   F3:Arranque   F4:Pantalla   F5:Reporte   F10:Salir     ");
do
  c1 = getch();
while (c1);
c2 = getch();
c = c1 + c2 - 58;
return(c);
}
```

```
/***********************
MODULO: ayuda
***********************/

#include <conio.h>

char texto[35][52] =
  { "F2: Muestreo",
    " ",
    "Permite la asignación de valores para los",
    "parametros de muestreo: ",
    " ",
    " ▪ Modo de muestreo ( 8 Digitales, 16 Digitales y " ,
    "   16 señales Digitales y 2 Analogicas )",
    " ▪ Frecuencia de muestreo en Hz",
    " ▪ Tiempo de muestreo en ms",
    " ▪ Modo de arranque ( Controlado y Manual)",
    " ▪ Señal de arranque: Si el modo de arranque es",
    "   controlado, el muestreo comienza si la señal de",
    "   arranque toma el nivel de voltaje especificado",
    " ",
    "F3: Arranque",
    " ",
    "Al seleccionar esta opcion arranca el muestreo de",
    "las señales. Si el modo es Manual, el muestreo se",
    "realiza inmediatamente.  En modo Controlado , el",
    "muestreo arranca cuando la señal de control alcanza",
    "el nivel de voltaje especificado",
    " ",
    "F4: Pantalla",
    " ",
    "Opcion para la visualizacion de las señales por",
    "pantalla",
    " ", " ",
    "F5: Reporte",
    " ",
    "Opcion que realiza la impresion de las senales",
    "muestreadas entre dos tiempos especificados"
    } ;

ayuda()
/*-------------------------------------------------------------------------
  Muestra la ventana de ayuda, detecta la tecla presionada por el usuario,
  segun la cual realiza el  avance o retroceso de pagina y la salida de
  la ayuda
-------------------------------------------------------------------------*/
{
int ysup=4, lini=0, linxwin=13;
char c;

 abre_winbord(10,4,70,4+18);
 gotoxy(27,2);
```

```
 cprintf("AYUDA");
 gotoxy(5,18);
 cprintf("Pg Dn:Pag.Siguiente   Pg Up:Pag.Anterior   F10:Salir");
 do
   {
    ysup=4;
    disp_help(ysup,lini,lini+linxwin);
    c=getch();
    if (!c) c=getch();
    switch (c)
      {
       case 'Q': lini += linxwin+1;
                 if (lini>34)  lini =0;
              break;
       case 'I': lini -= linxwin+1 ;
                 if (lini <0) lini =0;
              break;
      }
   }
 while (c!=68);
}

disp_help( ysup, lini, lfin)
int ysup, lini, lfin;
/*-------------------------------------------------------------------------
  Muestra el contenido del arreglo de cadenas texto, desde el elemento
  lini hasta el elemento lfin
  PARAMETROS:
  ysup  : Fila de la ventana actual desde la cual se empiezan a mostrar
          los datos
  lini  : Elemento inicial a mostrar del arreglo
  lfin  : Ultimo elemento a mostrar del arreglo
-------------------------------------------------------------------------*/
{
int y,i;

 y = ysup;
 for (i=lini; i<=lfin; i++)
   {
    gotoxy(3,y);
    if (i>34)
       cprintf("%-52.52s\n", " ");
    else
       cprintf("%-52.52s\n",texto[i]);
    y++;
   }
}
```

```
/************************
MODULO: leepar
************************/
#include <conio.h>;
#include <stdlib.h>;


llena_str(p,c,n)
unsigned char *p,c;
int n;
/*---------------------------------------------------------------------------
Crea una cadena formada por n veces el caracter c
PARAMETROS:
p : Una expresion que se evalue en la direccion de la cadena
c : Caracter que se repetir en la cadena
n : Numero de repeticiones del caracter anterior
---------------------------------------------------------------------------*/
{
int i;

 for (i=1; i<=n; i++)
   {
   *p = c;  p++;
   }
 *p = '\0';
}

abre_winbord( xis, yis, xdi, ydi )
int xis, yis, xdi, ydi;
/*---------------------------------------------------------------------------
  Abre una ventana con marco
  PARAMETROS:
  xis   : coordenada x de la esquina superior izquierda
  yis   : coordenada y de la esquina superior izquierda
  xdi   : coordenada x de la esquina inferior derecha
  xyi   : coordenada y de la esquina inferior derecha
---------------------------------------------------------------------------*/
{
unsigned char borde[80],pinta[80];
int anchow, altow, i;
extern int _wscroll;

 window(xis, yis, xdi, ydi );
 anchow = xdi - xis + 1;
 altow  = ydi - yis + 1;
 llena_str(&borde,205,anchow);
 borde[0] = '╔';
 borde[anchow-1] = '╗';
 gotoxy(1,1);
 cprintf("%s",borde);
 llena_str(&pinta,' ',anchow);
 pinta[0]='║';
```

```
 pinta[anchow-1]='|';


 for (i=2; i<altow; i++ )
 {
   gotoxy(1, i);
   cprintf("%s",pinta);
 }
 borde[0] ='L';
 borde[anchow-1] = 'J';
 _wscroll = 0;
 cprintf("%s",borde);
 _wscroll = 1;
 gotoxy(1,1);
}


lee_datmuestreo(pmodo,parranque,psenal, plogica, pfrec, ptiempo)
int *pmodo;
unsigned long *pfrec, *ptiempo;
unsigned char *psenal,*parranque,*plogica;
/* ---------------------------------------------------------------------
  Muestra la ventana de Asignacion de parametros de muestreo, y hace el
  requerimiento al usuario de estos parametros
  PARAMETROS:
    pmodo     : modo de muestreo
    parranque : modo de arranque
    psenal    : señal de arranque
    plogica   : nivel logico de la señal de arranque
    pfrec     : frecuencia de muestreo
    ptiempo   : tiempo de muestreo
---------------------------------------------------------------------*/
{
 int y, x, y1, x1, yup = 3, ydw = 8,t, i1,i4,i5;
 char c1, c2, ch1,ch2, cad[6],buff[8], *buff2,
 mensjm1[3][16] = { "8 Digitales","16 Digitales","16 Dig/2 Analog"},
 mensjm2[2][12] = { "Manual", "Controlado"},
 mensjm5[16][5] =
  { "D0.H","D1.H","D2.H","D3.H","D4.H","D5.H","D6.H", "D7.H",
    "D0.L","D1.L","D2.L","D3.L","D4.L","D5.L","D6.L", "D7.L" };

 /* inicializaciones */
 i1 = *pmodo;
 i4 = *parranque;
 i5 = (*psenal)*((*plogica+1)%2);

 /* muestra el formulario de parametros de muestreo  */
 y = yup + 1;
 textattr(0x71);
 abre_winbord(12,6,67,17);
 gotoxy(25,2);
 cprintf("MUESTREO");
```

```
gotoxy(3,y);
cprintf("Modo de muestreo       :   ");
cprintf("%-16s", mensjm1[i1] );
gotoxy(3,y+1);
cprintf("Frec. de muestreo  [Hz]:   ");
cprintf("%lu",*pfrec);
gotoxy(3,y+2);
cprintf("Tiempo de muestreo [ms]:   ");
cprintf("%lu",*ptiempo);
gotoxy(3,y+3);
cprintf("Modo de arranque       :   ");
cprintf("%-12s", mensjm2[*parranque]);
gotoxy(3,y+4);
cprintf("Señal de control       :   ");
cprintf("%-4s", mensjm5[(*psenal)+8*(1-*plogica)]);
gotoxy(25,y+6);
cprintf("F10:Salir");


gotoxy(30,4);
buff[0] = 6;
do
{
   y=wherey();
   x=wherex();
   c1 = getch();
   if (c1 == 13)
   {
     textattr(0x07);
     switch(y-yup)
     {
       case 1:
              if (c1 == 13)
              {
                i1++;
                i1%=3;
              cprintf("%-16s", mensjm1[i1]);
              *pmodo = i1;
              }
              break;
      case 4:
              if (c1 == 13)
              {
              i4++;
              i4%=2;
              cprintf("%-12s", mensjm2[i4]);
              *parranque = i4;
              }
              break;
      case 5:
              if ( (c1 == 13)&&(i4) )
              {
```

```
            i5++;
            i5%=16;
            cprintf("%-4s", mensjm5[i5]);
            *psenal = i5%8;
            if (i5 > 7) *plogica = 0;
            else *plogica =1;
            }
            break;
      case 2:
            do
            {
            clrfield(x,y,x+6);
            buff[0] =6;
            buff2 = cgets(buff);
            t = sscanf(buff2,"%5[0-9]", cad );
            }
            while (!t);
            *pfrec = atol(cad);
            break;
      case 3:
            do
            {
            clrfield(x,y,x+6);
            buff[0] =6;
            buff2 = cgets(buff);
            t = sscanf(buff2,"%5[0-9]", cad );
            }
            while (!t);
            *ptiempo = atol(cad);
            break;
      }
textattr(0x71);
}
if (c1 == 0 )
{
  c2=getch();
  switch(c2)
  {
    case 'H':
            if (y> yup+1) y--;
            break;
    case 'P':
            if (y< ydw) y++;
            break;
  };
gotoxy(30,yup+1);
cprintf("%-16s", mensjm1[i1]);
gotoxy(30,yup+2);
cprintf("%lu        ",*pfrec);
gotoxy(30,yup+3);
cprintf("%lu        ",*ptiempo);
gotoxy(30,yup+4);
```

```
    cprintf("%-12s", mensjm2[*parranque]);
    gotoxy(30,yup+5);
    if (i4)
      cprintf("%-4s", mensjm5[(*psenal)+8*(1-*plogica)]);
    else
      clrfield(30,yup+5,x+5);

    }
    gotoxy(30,y);
 }
 while (c2!=68);
}


clrfield(x,y,xclr)
int x,y,xclr;
/*-------------------------------------------------------------------------
  Limpia parte de una linea.
  PARAMETROS:
  x         : columna inicial de la linea
  y         : numero de la linea
  xclr      : columna final de la linea
-------------------------------------------------------------------------*/
{
char espacios[80]=
   {
"                                                                        "
   };
 espacios[xclr-x] = 0;
 gotoxy(x,y);
 cprintf("%s",espacios);
 gotoxy(x,y);
}
```

```
/************************
MODULO:  calcpar
************************/

#define     seg1    0x5000
#define     seg2    0x6000
#define     seg3    0x7000
#define     seg4    0x8000

#include <stdio.h>;
#include <math.h>;

float caltime(unsigned int s1,unsigned int c1,unsigned int s2,
unsigned int c2);

unsigned leetime(unsigned int *systick);

cal_param(modo,frec,tiempo,pmuestras,sep_muestra,falla)
unsigned long frec,tiempo;
unsigned int *pmuestras;
double *sep_muestra;
char *falla;
/*-------------------------------------------------------------------------
  Tiene como objetivo obtener el valor inicial del contador de retardo,
  el numero de muestras que corresponde al tiempo especificado y el tiempo
  en milisegundos de separacion entre dos muestras sucesivas.
  PARAMETROS:
  modo       : En modo 2 el calculo debe realizarse efectuando la lectura
               de 4 puertos, en modo 1 con la lectura de 2 puertos y en
               modo 0 efectuando la lectura de un solo puerto
  frec       : Frecuencia de muestreo
  tiempo     : Tiempo de muestreo
  pmuestras  : Retorna el numero de muestreos que hay que efectuar
  sep_muestra: Retorna el tiempo de separacion entre muestras en miliseg.
  falla      : Retorna 0 si la frecuencia es demasiado baja, 1 si es dema-
               siado alta
  Retorna el valor del contador para el retardo en la funcion de muestreo
-------------------------------------------------------------------------*/
{
 long double f1;
 double xr,xi,xf;
 unsigned int xc, qm,cnt1,syst1,cnt2,syst2,veces = 20;
 float tm,tl,tmuest;

 cnt1 = leetime(&syst1);
 tmuestreo (modo,1,0xffff);
 cnt2 = leetime(&syst2);
 tm = caltime(syst1,cnt1,syst2,cnt2);
/* tm es es el tiempo en segundos que toma en hacer 0xffff+1 muestreos */
 cnt1 = leetime(&syst1);
 tretardo(veces,0xffff);
 cnt2 = leetime(&syst2);
```

```
 tl = caltime(syst1,cnt1,syst2,cnt2)/(1.0 * veces);
/* tl es el tiempo en segundos que toma en hacer un lazo 0xffff veces */
/*printf("tl:%e ",tl); */

 f1 = 1.0*0x10000/frec;
 xr = (f1 - tm)/tl;

 xf = modf(xr,&xi);
 xc = xf >0.5 ? xr + 1.0 : floor(xr);

 if (xr >0xffff)
    {
    *falla = 0;
    return(0);
    };
 if  (xr<=0.5)
    {
    *falla = 1;
    return(0);
    };
/* printf(" %e %d ",xr,xc);*/

/* Asignacion de qm , que contiene el valor al numero de muestras con que
   se llamara a t2muestreo */
 if (xc > 100)
    qm = 100+ 10000*(400/xc);
 else
    qm =0x7000;

 cnt1= leetime(&syst1);
 t2muestreo(modo,xc,qm);
 cnt2 = leetime(&syst2);
/* printf("\n %u %u %u %u \n",syst1,cnt1,syst2,cnt2);*/
 tmuest = caltime(syst1,cnt1,syst2,cnt2);
/* tmuest  es el tiempo  que lleva tomar qm    muestras */
 *sep_muestra = tmuest*1000/qm;
/* separacion entre muestras, en ms  */
 *pmuestras = tiempo*qm/(tmuest*1000.0);

/* printf(" %e  %u %e\n",  tmuest, *pmuestras, *sep_muestra);*/
 return(xc);
}


tmuestreo(modo,retardo,muestras)
unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Funcion que realiza el muestreo, pero sin el lazo de retardo.
  PARAMETROS:
  modo      : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
              primeros y en modo 0 solo el puerto 0x0340
  retardo   : Es usado para que esta funcion guarde similitud con la fun
```

```
              cion muestreo
  muestras   : Un numero de 2 bytes que especifica el numero de lecturas
               que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/
{

          _SI =  muestras;
          _BX = 0;

 if (modo==2)
   {
   asm     push ds;
 lazo1:

   asm {
         mov ax,seg1
         mov ds,ax

         mov dx, 0x0344
         in al, dx

         mov dx,0x0340
         in al,dx
         mov [bx], al

         inc dx
         mov ax,seg2
         mov ds,ax
         in al, dx
         mov [bx], al

         inc dx
         mov ax,seg3
         mov ds,ax
         in al, dx
         mov [bx], al

         inc dx
         mov ax,seg4
         mov ds,ax
         in al, dx
         mov [bx], al

         inc bx

         cmp bx,si
         jne lazo1

         pop ax
         mov ds,ax
     }
   }
```

```
if (modo==1)
  {
  asm     push ds;
lazo2:

  asm {
        mov ax,seg1
        mov ds,ax

        mov dx, 0x0344
        in al, dx

        mov dx,0x0340
        in al,dx
        mov [bx], al

        inc dx
        mov ax,seg2
        mov ds,ax
        in al, dx
        mov [bx], al

        inc bx

        cmp bx,si
        jne lazo2

        pop ax
        mov ds,ax
      }
  }

if (modo==0)
  {
  asm {
        push ds
        mov ax,seg1
        mov ds,ax
      }

lazo3:

  asm {

        mov dx, 0x0344
        in al, dx

        mov dx,0x0340
        in al,dx
        mov [bx], al

        inc bx
```

```
          cmp bx,si
          jne lazo3

          pop ax
          mov ds,ax
        }
    }

 return;
}



tretardo(veces,retardo)
unsigned int veces,retardo;
/*-------------------------------------------------------------------------
  Funcion que se usa en el calculo de parametros de muestreo. Representa el
  tiempo que toma ejecutar repetidas veces el lazo de retardo.
  PARAMETROS:
  veces      : Numero de veces que se ejecuta un lazo exterior
  retardo    : Nuemero de veces que se ejecuta el lazo de retardo
-------------------------------------------------------------------------*/
{
   _DI = retardo;
   _DX = veces;
repite:
     asm  mov cx,di;
retar2:
     asm  loop retar2;
     asm  dec dx;
     asm  jnz repite;
}



t2muestreo(modo,retardo,muestras)
unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Funcion que se usa en el calculo de parametros de muestreo. Representa
  el tiempo que toma realizar un determinado numero de muestras con un
  retardo especificado por la variable retardo.
  PARAMETROS:
  modo       : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
               primeros y en modo 0 solo el puerto 0x0340
  retardo    : Un numero de 2 bytes. Se usa con un lazo, para provocar un
               retardo entre muestras sucesivas
  muestras   : Un numero de 2 bytes que especifica el numero de lecturas
               que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/
{
```

```
        _DI =  retardo;
        _SI =  muestras;
        _BX = 0;

if (modo==2)
  {
  asm     push ds;
lazo1:

  asm {
        mov cx,di
        mov ax,seg1
        mov ds,ax

        mov dx, 0x0344
        in al, dx

        mov dx,0x0340
        in al,dx
        mov [bx], al

        inc dx
        mov ax,seg2
        mov ds,ax
        in al, dx
        mov [bx], al

        inc dx
        mov ax,seg3
        mov ds,ax
        in al, dx
        mov [bx], al

        inc dx
        mov ax,seg4
        mov ds,ax
        in al, dx
        mov [bx], al

        inc bx
    }

retar1:

asm  {  loop retar1

        cmp bx,si
        jne lazo1

        pop ax
        mov ds,ax
    }
```

```
   }

if (modo==1)
   {
 asm     push ds;
 lazo2:

 asm {
         mov cx,di
         mov ax,seg1
         mov ds,ax

         mov dx, 0x0344
         in al, dx

         mov dx,0x0340
         in al,dx
         mov [bx], al

         inc dx
         mov ax,seg2
         mov ds,ax
         in al, dx
         mov [bx], al

         inc bx
     }

 retar2:

 asm  {  loop retar2

         cmp bx,si
         jne lazo2

         pop ax
         mov ds,ax
     }
   }

if (modo==0)
   {
 asm   {
         push ds
         mov ax,seg1
         mov ds,ax
       }
 lazo3:

 asm {
         mov cx,di
```

```
          mov dx, 0x0344
          in al, dx

          mov dx,0x0340
          in al,dx
          mov [bx], al

          inc bx
       }

 retar3:

 asm  {  loop retar3

          cmp bx,si
          jne lazo3

          pop ax
          mov ds,ax
       }
    }

return;
}

unsigned leetime(systick)
unsigned int *systick;
/*-------------------------------------------------------------------------
  Obtiene el valor actual del conteo de ticks del sistema y del contador
  0 del timer 8253
  PARAMETROS:
  systick     : Valor actual del conteo de los ticks del sistema                    de 4 puertos,
en modo 1 con la lectura de 2 puertos y en

  Retorna el valor del contador 0 del timer 8253
-------------------------------------------------------------------------*/
{
unsigned char trsc = 0x0c2, stat,cntl,cnth;
unsigned int  cnt, comprueb;
 asm {
     mov ah,0
     int 0x1a };
 *systick = _DX;
 asm {
     mov al, trsc
     out 0x043, al
     in al, 0x040
     mov stat, al
     in al, 0x040 };
 cntl = _AL;
 asm in al, 0x040;
 cnth = _AL;
```

```
 asm {
     mov ah,0
     int 0x1a );
 comprueb = _DX;
 if (comprueb != *systick)
   printf("!!!!!!");
   if (cnth > 0x80)
      *systick = comprueb;
 cnt = cnth*0x0100 + cntl;
 return(cnt);
}

float caltime(s1,c1,s2,c2)
unsigned int s1,c1,s2,c2;
/*-------------------------------------------------------------------------
  Obtiene el tiempo en segundosque existe entre dos pares ticks-contador
  PARAMETROS:
  s1    : Valor de los ticks del sistema para el primer par
  c1    : Valor del contador 0 del 8253 para el primer par
  s2    : Valor de los ticks del sistema para el segundo par
  c2    : Valor del contador 0 del 8253 para el segundo par

  Retorna un numero real, con el tiempo transcurrido entre dos pares,
  en segundos
-------------------------------------------------------------------------*/
{
unsigned long l;
unsigned int x1,x2;
float r;

 if (s1>s2)
    s1 = 0x10000 -s1;
 l = (s2 -s1)*0x10000+c1-c2;
 r = l/1193180.0;
 return(r);
}
```

```
/***********************
MODULO: muestreo
***********************/

#define    seg1   0x5000
#define    seg2   0x6000
#define    seg3   0x7000
#define    seg4   0x8000



muestreo(modo,retardo,muestras)

unsigned int modo,retardo,muestras;
/*-------------------------------------------------------------------------
  Realiza el muestreo de las señales que se leen por los puertos
  0x0340, 0x0341, 0x0342, 0x0342
  PARAMETROS:
  modo      : En modo 2 utiliza los 4 puertos, en modo 1 utiliza los dos
              primeros y en modo 0 solo el puerto 0x0340
  retardo   : Un numero de 2 bytes. Se usa con un lazo, para provocar un
              retardo entre muestras sucesivas
  muestras  : Un numero de 2 bytes que especifica el numero de lecturas
              que se efectuaran sobre los puertos
-------------------------------------------------------------------------*/
{

         _DI =  retardo;
         _SI =  muestras;
         _BX = 0;

if (modo==2)
   {
 asm     push ds;
 lazo1:

 asm {
         mov cx,di
         mov ax,seg1
         mov ds,ax

         mov dx, 0x0344
         in al, dx

         mov dx,0x0340
         in al,dx
         mov [bx], al

         inc dx
         mov ax,seg2
         mov ds,ax
         in al, dx
```

```
        mov [bx], al

        inc dx
        mov ax,seg3
        mov ds,ax
        in al, dx
        mov [bx], al

        inc dx
        mov ax,seg4
        mov ds,ax
        in al, dx
        mov [bx], al

        inc bx
    }

retar1:

asm  {  loop retar1

        cmp bx,si
        jne lazo1

        pop ax
        mov ds,ax
    }
  }

if (modo==1)
  {
asm     push ds;
lazo2:

asm {
        mov cx,di
        mov ax,seg1
        mov ds,ax

        mov dx, 0x0344
        in al, dx

        mov dx,0x0340
        in al,dx
        mov [bx], al

        inc dx
        mov ax,seg2
        mov ds,ax
        in al, dx
        mov [bx], al
```

```
            inc bx
        }

 retar2:

 asm  {  loop retar2

            cmp bx,si
            jne lazo2

            pop ax
            mov ds,ax
        }
    }

if (modo==0)
    {
 asm  {
            push ds
            mov ax,seg1
            mov ds,ax
        }
 lazo3:

 asm {
            mov cx,di

            mov dx, 0x0344
            in al, dx

            mov dx,0x0340
            in al,dx
            mov [bx], al

            inc bx
        }

 retar3:

 asm  {  loop retar3

            cmp bx,si
            jne lazo3

            pop ax
            mov ds,ax
        }
    }
return;
}

arranque(modo,bstart,nivel)
```

```
unsigned char modo,bstart,nivel;
/*-----------------------------------------------------------------------
  Determina el momento en que se debe comenzar el muestreo, dependiendo de
  sus parametros.
  PARAMETROS:
  modo   : Si es 0, arranca inmediatamente. Si es 1, arranca cuando la
           señal de arranque tiene el nivel especificado.
  bstart : Byte de arranque. El bit n se encuentra seteado si la señal n
           es la señal de arranque.
  nivel  : Nivel de la señal de arranque. 1 = alto, 0 = bajo.
-----------------------------------------------------------------------*/
{
char ch;

  if ( modo==1 )
    {
         _CL = bstart;
         _BL = nivel;
     lazo_arr:

     asm {
         mov dx, 0x0344
         in al, dx

         mov dx,0x0340
         in al,dx

         cmp bl,0
         jnz label1
         not al
         }

     label1:
     asm {
         and al,cl
         cmp al,0

         jz  lazo_arr
          }
     }
     return;
  }
```

```
/***********************
MODULO: pantalla
***********************/

#define      seg1    0x5000
#include <dos.h>;
#include <graphics.h>;
#include <stdlib.h>;

unsigned char huge *pt, *qt, *pmax, *pmin ;
unsigned int hg=0, lg=1,paso=100;
int modo;
char textop[30][52] =
  { "F2= \x1a:",
    "Avanza en las coordenadas del tiempo segun el ",
    "Paso especificado",
    " ",
    "F3= \x1b:",
    "Retrocede en las coordenadas del tiempo segun el ",
    "Paso especificado",
    " ",
    "F4= Ir-a :  ",
    "Permite visualizar las señales muestreadas en una",
    "la ubicacion con respecto al tiempo asignada",
    "F5= Cambio-Sup:",
    "Conmuta las señales presentadas el la mitad supe-",
    "rior de la pantalla",
    " ",
    "F6= Cambio-Inf:",
    "Conmuta las señales presentadas el la mitad infe-",
    "rior de la pantalla",
    " ",
    "F7: Paso",
    "Permite asignar un intervalo de tiempo diferente",
    "para el Avance o Retroceso"
    ," "
    } ;


ver_pantalla(separ,v_modo, muestras)
double separ;
int v_modo;
unsigned int muestras;
/*-------------------------------------------------------------------------
  Maneja la pantalla de visualizacion de señales. Presenta el menu de esta
  pantalla y espera por una tecla funcional. Llama a las funciones de
  graficacion
  PARAMETROS:
  separ     : tiempo de separacion entre muestras sucesivas en miliseg.
  v_modo    : modo de muestreo (8, 16 o 18 señales)
  muestras  : Numero de muestreos realizados
-------------------------------------------------------------------------*/
```

```
{
   char a[80]=
{"F1:Ayuda  F2: \x1a  F3: \x1b  F4:Ir-a  F5:Cambio-Sup  F6:Cambio-Inf  F7:Paso  F10:Sal.");

   struct REGPACK reg;
   unsigned char ch1,ch2;
   unsigned int dir,of=0,q,ret,muest;
   int   gdriver = 2, gmode = 5, grerror;
   char pathtodriver[20] = {""};


   modo=v_modo;
   pt = MK_FP(seg1,of);
   pmax = MK_FP(seg1,muestras);
   pmin = MK_FP(seg1,0);
   initgraph(&gdriver,&gmode, &pathtodriver[0]);
   grerror = graphresult();
   if (grerror != grOk)
      cprintf("Error de Graficos: %s\n", grapherrormsg(grerror));

   do
   {
      reg.r_ax = 0x0011;
      intr(0x10, &reg);
      wrln(&a[0],80,0,0);
      muestragr(hg,lg);
      marcos();
      escalas(separ);
      do
        ch1=getch();
      while (ch1);
      ch2=getch();
      q = procesa(ch1,ch2,separ);
    }
    while (q);

    reg.r_ax = 0x0003;
    intr(0x10, &reg);
}


marcos()
/* dibuja el marco de la pantalla */
  {
    line(27,40,27,456);
    line(613,40,613,456);
    line(27,40,613,40);
    line(27,456,613,456);
    outtextxy(608,470,"[ms]");
   }

escalas(separ)
```

```
double separ;
/*-------------------------------------------------------------------------
Presenta las coordenadas de tiempo en la pantalla de visualizacion de
señales
separ   : tiempo de separacion entre muestras en milisegundos
-------------------------------------------------------------------------*/
{
 unsigned int x,consec,temp;
 int dec,sig;
 double es;
 char ses[8];

 temp = (FP_SEG(pt)-seg1)*16+FP_OFF(pt);
 for (x=0; x<576; x+=100)
   {
   line(32+x,456,32+x,463);
   consec= temp+x;
   es = consec*separ;
   fcvt(es,7,&dec,&consec);
   gcvt(es,dec+1,ses);
   outtextxy(x+32,470,ses);
   }
}

etiquetas(t,p)
int t,p ;
/*-------------------------------------------------------------------------
Escribe los nombres de las señales que se presentan en pantalla
PARAMETROS:
t   : 1 si se presentan los nombres de las señales D0 a D7
      2 si se presentan los nombres de las señales D7 a D15
      3 si se presentan los nombres de las señales A0 y A1
p   : controla la posicion vertical a partir de la cual se displayan los
      nombres de las señales
-------------------------------------------------------------------------*/
{
int f=1;

     if (!modo) f=2;
     switch (t)
     {
      case 1:
             outtextxy(0,52+25*0*f+p*200,"D0");
             outtextxy(0,52+25*1*f+p*200,"D1");
             outtextxy(0,52+25*2*f+p*200,"D2");
             outtextxy(0,52+25*3*f+p*200,"D3");
             outtextxy(0,52+25*4*f+p*200,"D4");
             outtextxy(0,52+25*5*f+p*200,"D5");
             outtextxy(0,52+25*6*f+p*200,"D6");
             outtextxy(0,52+25*7*f+p*200,"D7");
             break;
       case 2:
```

```
            outtextxy(0,52+25*0+p*200,"D8");
            outtextxy(0,52+25*1+p*200,"D9");
            outtextxy(0,52+25*2+p*200,"D10");
            outtextxy(0,52+25*3+p*200,"D11");
            outtextxy(0,52+25*4+p*200,"D12");
            outtextxy(0,52+25*5+p*200,"D13");
            outtextxy(0,52+25*6+p*200,"D14");
            outtextxy(0,52+25*7+p*200,"D15");
            break;
       case 3:
            outtextxy(2,52+p*220,"5");
            outtextxy(2,52+64+p*220,"0");
            outtextxy(0,52+32+p*220,"A0");
            outtextxy(2,152+p*220,"5");
            outtextxy(2,152+64+p*220,"0");
            outtextxy(0,152+32+p*220,"A1");
      }
   }


muestragr(h,l)
unsigned int h,l;
/*-------------------------------------------------------------------------
  Muestra las señales leidas en la pantalla de visualizacion
  PARAMETROS
  h   : Determina que bloque de señales se presenta en la zona superior de
        la pantalla
  l   : Determina que bloque de señales se presenta en la zona inferior de
        la pantalla
-------------------------------------------------------------------------*/
{
   switch (h)
     {
     case 0:
            mapgr(pt,50);
            etiquetas(1,0);
            break;
     case 1:
            mapgr(pt+0x10000,50);
            etiquetas(2,0);
            break;
     case 2:
            mapanalog(pt+0x20000,120);
            mapanalog(pt+0x30000,220);
            etiquetas(3,0);
            break;
     }

   if (modo>0)
     switch (l)
       {
```

```
        case 0:
             mapgr(pt,250);
             etiquetas(1,1);
             break;
        case 1:
             mapgr(pt+0x10000,250);
             etiquetas(2,1);
             break;
        case 2:
             mapanalog(pt+0x20000,340);
             mapanalog(pt+0x30000,440);
             etiquetas(3,1);
             break;
      }
    }


procesa(c1,c2,separ)
unsigned char c1,c2;
double separ;
/*
  Ejecuta una de las opciones del menu de la pantalla de visualizacion de
  señales, segun la tecla funcional presionada
  PARAMETROS:
  c1   : Primer codigo de la tecla presionada
  c2   : Segundo codigo de la tecla presionada
  RETORNA:
  Si se presiona F10 retorna 0, 1 de otra manera
*/

{
unsigned int q = 1,temp,t;
char  cad[8], buff3[8];
float tcoor;


   if (!c1)
      switch (c2)
      {
      case 59:
              ayudap();
              setviewport(0,0,639,479,1);
              break;
      case 60:
              pt = (FP_SEG(pt+paso)*16+FP_OFF(pt+paso)+576) >pmax
                   ? MK_FP(seg1,0) : pt + paso;
              break;
      case 61:
             if ( (FP_SEG(pmin)*16+FP_OFF(pmin)+paso)
              > FP_SEG(pt)*16+FP_OFF(pt))
                 qt = MK_FP(seg1,0);
              else
```

```
                qt = pt -paso;

            pt =qt;
            break;
        case 62:
            outtextxy(0,16,"tiempo [ms]:        " );
            do
            {
              gnumstr(120,16,&buff3[0]);
              outtextxy(608,16,buff3);
              t = sscanf(buff3,"%5[0-9]", cad );
            }
            while (t == 0);
            temp = atoi(cad);
            tcoor = temp / separ;
            temp = tcoor;
            pt = MK_FP(seg1,temp);
            break;
        case 63:
            ++hg;
            hg %= modo+1;
            break;
        case 64:
            ++lg;
            lg %= modo+1;
            break;
        case 65:
            outtextxy(0,16,"tiempo [ms]:        " );
            do
            {
              gnumstr(120,16,&buff3[0]);
              outtextxy(608,16,buff3);
              t = sscanf(buff3,"%5[0-9]", cad );
            }
            while (t == 0);
            temp = atoi(cad);
            paso = temp/separ;
            break;
        case 68:
            q = 0;
            break;
        );
        return (q);
    }


mapanalog(p,lin0)
unsigned char huge *p;
unsigned int lin0;
/*-------------------------------------------------------------------------
  Grafica una señal analogica
```

```
  PARAMETROS
  p    : Puntero a la direccion de memoria del inicio de los datos para la
         porcion a graficar
  lin0 : Determina la posicion vertical en la pantalla a partir de la cual
         se grafica
-------------------------------------------------------------------------*/
{
   unsigned int i;
   unsigned char dat;
   struct REGPACK reg;

   for (i=32; i<608; i++)
   {
      dat = *p >>2;
      reg.r_ax = 0x0c01;
      reg.r_bx = 0x0000;
      reg.r_cx = i;
      reg.r_dx = lin0 - dat;
      intr(0x10,&reg);
      p++;
    );
}


mapgr( p, l)
unsigned char huge *p;
int l;
/*-------------------------------------------------------------------------
  Grafica un bloque de 8 señales digitales.
  PARAMETROS:
  p   : Puntero a la direccion de memoria del inicio de los datos para la
        porcion a graficar
  l   : determina la posicion en la pantalla a partir de la cual se
        grafica
-------------------------------------------------------------------------*/
{
int j,i,k,ex,f=1;
unsigned char dat;

  if (!modo) f =2;
  for (j=4; j<76; j++)
  {
    p += 8;
    for (k=0; k<8; k++)
    {
      dat= 0;
      for( i=7; i>=0; i--)
      {
        --p;
        dat >>= 1;
        dat |=  (*p << k) & 0x80;
      )
```

```
      p += 8;
      ex= (l+25*f*(7-k))*80+j;
      /* 7-k para invertir el orden de presentacion */
      pokeb(0xa000,ex,dat);
      pokeb(0xa000,(ex + f*800),~dat);
    }
  }
}


wrln(p,n,x,y)

unsigned char huge *p;
unsigned int x,y,n;
{
struct REGPACK reg;

      reg.r_ax = 0x1300;
      reg.r_es = FP_SEG(p);
      reg.r_bp = FP_OFF(p);
      reg.r_bx = 0x0001;  //at
      reg.r_cx = n;
      reg.r_dx = y*256+x;
      intr(0x10,&reg);
}

gnumstr(x,y,s)
int x,y;
char *s;
{
 unsigned char c;
 int i=0;

 do
 {
   *(s+i+1)=0;
   c = getch();
   if ( (c>47)&&(c<58) )
   {
      *(s+i) = c;
      outtextxy(x,y,s);
      i++;
   }
 }
 while ( c!=13 );

}


ayudap()
/*-------------------------------------------------------------------------
  Muestra una ventana de ayuda para la pantalla de visualizacion de
```

```
  señales. Efectua el avance/retroceso de pantalla
-------------------------------------------------------------------------*/
{
int ysup=4, lini=0, linxwin=10,i;
char c;

 setviewport(80,64,560,352,1);
 setlinestyle(0, 0, 3);
 do
   {
    ysup=4;
    clearviewport();
    line(0,0,0,288);
    line(0,0,480,0);
    line(480,288,0,288);
    line(480,288,480,0);
    outtextxy(14*16,16,"AYUDA");
    outtextxy(16*2,272,"Pag.Dn:Siguiente    Pag.Up:Anterior    F10:Salir");
    gdisp_help(ysup,lini,lini+linxwin);
    c=getch();
    if (!c) c=getch();
    switch (c)
      {
       case 'Q': lini += linxwin+1;
                 if (lini>22)  lini =0;
              break;
       case 'I': lini -= linxwin+1 ;
                 if (lini <0) lini =0;
              break;
      }
   }
 while (c!=68);
 setlinestyle(0, 0, 1);
}

gdisp_help( ysup, lini, lfin)
int ysup, lini, lfin;
/*-------------------------------------------------------------------------
  muestra en modo grafico el contenido del arreglo de cadenas texto, desde
  el elemento  lini hasta el elemento lfin
  PARAMETROS:
  ysup      : linea de posicion vertical inicial
  lini      : elemento inicial a mostrar
  lifi      : elemento final a mostrar
-------------------------------------------------------------------------*/
{
int y,i,x;

 y = ysup*16;
 x=3*8;
 for (i=lini; i<=lfin; i++)
   {
```

```
    if (i>22)
      outtextxy(x,y," ");
    else
      outtextxy(x,y,textop[i]);
    y+=16;
   }
}
```

```
/**********************
MODULO: reporte
**********************/

#define     Max_sen      18
#define     Max_a_amp    128
#define     Min_a_amp    64
#define     Max_d_amp    200
#define     Min_d_amp    20
#define     seg1         0x5000
#define     seg2         0x6000
#define     seg3         0x7000
#define     seg4         0x8000
#define     Dfac         1.8
#define     Afac         1.4
#define     Esp_esc      10
#define     I_marco      50
#define     Max_text     80

#include <stdio.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>

int
   existe_sen[18],
   org_sen[18],
   D_marco,
   max_col=960;
unsigned char
   prlinea[1000],
   id_sen[Max_sen][11];
FILE *printer;
int pos[21][3]={12,4,40,24,5,5,49,5,5};
char datos[21][50]=
 { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};


reporte(modo,paso,dmarco)
int modo,dmarco;
double paso;
/*-------------------------------------------------------------------------
  Llama a las funciones de generacion de reporte: ventana de asignacion de
  datos para reporte, calculo de parametros e impresion de reporte
  PARAMETROS:
  modo      : modo de muestreo:0=8 señales,1=16 señales,2=18 señales
  paso      : separacion en milisegundos entre dos muestras sucesivas
  dmarco    : posicion del marco derecho (en puntos de impresion )
-------------------------------------------------------------------------*/

{
```

```
 int  a_num, d_num, a_amp, d_amp, i,flag;
 char tit[40],c;
 double tini, tfin;
 unsigned int pini, pfin,cmd;

 D_marco = dmarco;
 cmd=rp_ventana(modo,&tini,&tfin,tit);
 textattr(0x07);

 if  (cmd==27)
   {
   rp_calparam( I_marco+Esp_esc, dmarco, &a_num, &d_num, &a_amp, &d_amp);
   while ( (biosprint(2,0,0)&0x28) )
     {
     gotoxy(2,25);
     cprintf("Revise que la impresora esté lista ...   Presione una tecla");
     c=getch();
     }
   printer = fopen("LPT1","w+b");
   gotoxy(2,25);
   cprintf("         Imprimiendo . . .                                    ");
   pini = tini/paso;
   pfin = tfin/paso;
   imp_diagrama(tini,tfin,pini,pfin,paso,a_amp,d_amp,tit);
   gotoxy(2,25);
   cprintf("                                                              ");
   }
 }

rp_ventana(modo,tini,tfin,tit)
int modo;
double *tini, *tfin;
char *tit;
/*---------------------------------------------------------------------------
  Abre la ventana de asignacion de datos de muestreo, controla el input de
  los campos de acuerdo al modo especificado
  Retorna el codigo de la tecla ESCAPE o de F10
  PARAMETROS:
  modo  : modo de muestreo, como en la funcion reporte
  tini  : retorna el tiempo a partir del cual se imprimira el reporte
  tfin  : retorna el tiempo hasta el cual se imprimira el reporte
  tit   : retorna el titulo del reporte
  tambien manipula los objetos globales existe_sen e id_sen
---------------------------------------------------------------------------*/

{
int i, flag;
long ttemp;
unsigned int ich;
 textattr(0x71);
 abre_winbord(8,3,72,24);
 gotoxy(30,2);
```

```
cprintf("REPORTE");
gotoxy(3,4);
cprintf("Titulo : ");
cprintf("%s",datos[0]);
gotoxy(3,5);
cprintf("Tiempo inicial [ms]: ");
cprintf("%s",datos[1]);
gotoxy(30,5);
cprintf("Tiempo final [ms]: ");
cprintf("%s",datos[2]);
gotoxy(20,20);
cprintf("Esc: Imprimir      F10: Salir");

for (i=0; i<8; i++)
  {
   gotoxy(3,9+i);
   cprintf("D%-2d :  ",i);
   cprintf("%-s",datos[i+3]);
   pos[i+3][0]=  10;
   pos[i+3][1]=  9+i;
   pos[i+3][2]=  8 ;
  }
if (modo>0)
  for (i=8; i<16; i++)
  {
   gotoxy(30,i+1);
   cprintf("D%-2d :  ",i);
   cprintf("%-s",datos[i+3]);
   pos[i+3][0]=  37;
   pos[i+3][1]=  1+i;
   pos[i+3][2]=  8 ;
   }
if (modo==2)
  {
   gotoxy(30,17);
   cprintf("A0  :  ");
   cprintf("%-s",datos[19]);
   gotoxy(30,18);
   cprintf("A1  :  ");
   cprintf("%-s",datos[20]);
   pos[19][0]=  37;
   pos[19][1]=  17;
   pos[19][2]=  8 ;
   pos[20][0]=  37;
   pos[20][1]=  18;
   pos[20][2]= 8 ;
  }

i=0;
do
  {
   do
```

```
      {
       flag = 1;
       ich = inputf(i);
       if  ((i==1)||(i==2))
          flag = valida(i);
      }
    while (!flag);

    if (ich==72*256)
       i--;
    if (ich==80*256||ich==13)
       i++;
    if (i>20)
       i=20;
    if (i<0)
       i=0;
  }
 while ( !((ich==27)||(ich==68*256)) );

 textmode(C80);
 textattr(0x07);
 sprintf(tit,"%s",datos[0]);
 ttemp = atol(datos[1]);
 *tini = ttemp;
 ttemp = atol(datos[2]);
 *tfin = ttemp;
 for (i=0; i<18; i++)
   if  ( datos[i+3][0]<33 )
      existe_sen[i] = 0;
   else
     {
      existe_sen[i] = 1;
      sprintf(id_sen[i],"%s",datos[i+3]);
     }
 return(ich);
}


valida(i)
int i;
/*-------------------------------------------------------------------------
Retorna 1 si datos[i] es una cadena numerica o nulo, 0 de otra manera
PARAETROS:
i  : indice que corresponde al elemento del arreglo datos que se validara
-------------------------------------------------------------------------*/

{
int t; char cad[8];

 if (!datos[i]) return(1);
 t = sscanf(datos[i],"%5[0-9]",cad);
 if (t)
```

```
   sprintf(datos[i],"%s",cad);
 return (t);
}

inputf(i)
int i;
/*-------------------------------------------------------------------------
  Lee una tecla del teclado y retorna su codigo. Si la tecla es ENTER,
  lee el campo i del formulario de datos para impresion del reporte.
  Retorna el codigo de la tecla presionada antes de ingresar el campo.
  PARAMETROS:
  i  : Especifica el campo que se leera
-------------------------------------------------------------------------*/

{
char *buff2, buff[42], forma[10],forma2[10];
unsigned int c1,j;

  gotoxy(pos[i][0],pos[i][1]);
  c1 = getch();
  if (!c1)
     c1 = getch()*256;
  if (c1==13)
    {
    textattr(0x07);
    clrfield(pos[i][0],pos[i][1],pos[i][0]+pos[i][2]);
    buff[0]=pos[i][2]+1;
    buff2=cgets(buff);
    sprintf(forma2,"%%%dc",buff[1]);
    sscanf(buff2,forma2,datos[i]);
    datos[i][buff[1]]=0;
    sprintf(forma,"%%-%ds",pos[i][2]);
    gotoxy(pos[i][0],pos[i][1]);
    textattr(0x71);
    cprintf(forma,datos[i]);
    }
   return(c1);
}

rp_calparam( imarco, dmarco, a_num, d_num, a_amp, d_amp)
int imarco, dmarco, *a_num, *d_num, *a_amp, *d_amp ;
/*-------------------------------------------------------------------------
  Obtiene el numero de señales analogicas y de señales digitales a
  imprimir, la amplitud con que seran impresas las señales analogicas y
  digitales en el reporte. Tambien manipula los objetos globales existe_sen
  y org_sen.
  PARAMETROS:
  imarco    : distancia del marco izquierdo, en puntos de impresion
  dmarco    : distancia del marco derecho, en puntos de impresion
  anum      : numero de señales analogicas
  dnum      : numero de señales digitales
  a_amp     : amplitud de impresion de las señales analogicas
```

```
  d_amp     : amplitud de impresion de las señales analogicas
-------------------------------------------------------------------------*/
{
int wa_num=0, wd_num=0, wa_amp, wd_amp, td_amp, i,cnt=0;

 for (i=0; i< Max_sen; i++ )
   if (existe_sen[i])
      if (i>15)
        wa_num++;
      else
        wd_num++;

 wa_amp = Max_a_amp*2;

 if (wd_num)
   do
     {
      wa_amp = wa_amp/2;
      wd_amp = ( dmarco - imarco - Afac*wa_num*wa_amp)/( Dfac*wd_num );
     }
   while (wd_amp < Min_d_amp);
 else
   {
    wd_amp = 0;
    wa_amp = Max_a_amp;
   }

 for (i=0; i< Max_sen; i++ )
    if (existe_sen[i])
       if (i>15)
          if (wa_num==2)
             org_sen[i] = dmarco - wd_num*wd_amp*Dfac-(i-15)*wa_amp*Afac;
          else
             org_sen[i] = dmarco - wd_num*wd_amp*Dfac-wa_amp*Afac;
       else
         {
          cnt++;
          org_sen[i] = dmarco - cnt*wd_amp*Dfac;
         }

 *a_num = wa_num;
 *d_num = wd_num;
 *a_amp = wa_amp;
 *d_amp = wd_amp;
 return;
}


imp_diagrama(tini,tfin,pini,pfin,paso,a_amp, d_amp,rp_titulo)
double tini,tfin,paso;
unsigned int pini,pfin;
int a_amp, d_amp;
```

```
char *rp_titulo;
/*-------------------------------------------------------------------------
  Funcion que imprime el diagrama de tiempo segun lo especifican los parametros
  PARAMETROS:
  tini      : tiempo inicial en milisegundos
  tfin      : tiempo final en milisegundos
  pini      : numero de la muestra inicial a imprimir
  pfin      : numero de la muestra final a imprimir
  paso      : separacion entre dos muestras sucesivas, en milisegundos
  a_amp     : amplitud maxima de las señales analogicas
  d_amp     : amplitud maxima de las señales digitales
  rp_titulo : titulo del reporte
-------------------------------------------------------------------------*/
{
unsigned int pag=0, psal, pactual, punxpag;
char ses[8];
double es;
int dec,sig;

 punxpag = 600;
 pactual = pini;
 do
   {
   pag++;
   imp_cabecera(pag,rp_titulo);
   es = pactual*paso;
   fcvt(es,7,&dec,&sig);
   gcvt(es,dec+1,ses);
   imp_nombre_sen(ses,a_amp,d_amp);
   inic_grprinter();
   imp_marco(1,a_amp,d_amp);
   psal = imp_cuerpo(punxpag,&pactual,pfin,a_amp, d_amp);
   imp_marco(0x80,a_amp,d_amp);
   imp_pie(paso);
   if (!psal)
     salto_pag();
   }
 while (!psal);
}

/*=========================================================*/

imp_cabecera(pag,rp_titulo)
unsigned int pag;
char *rp_titulo;
/*-------------------------------------------------------------------------
  Imprime la cabecera de pagina, que contiene el titulo y numero de pagina
  del reporte.
  PARAMETROS:
  pag       : numero de pagina actual
  rp_titulo : titulo del diagrama de tiempo
-------------------------------------------------------------------------*/
```

```
{
char *tx_pag,l_imp[Max_text];
 inic_pr();
 sprintf(l_imp,
"    ┌──────────────────────────────────────────────────────────────────────────────┐ ");
 fputs(l_imp,printer);
 sprintf(l_imp,
"    | TITULO:  %-40s           PAGINA: %3d |\n" , rp_titulo, pag);
 fputs(l_imp,printer);
 sprintf(l_imp,
"    └──────────────────────────────────────────────────────────────────────────────┘\n");
 fputs(l_imp,printer);
}


imp_marco(tipo,a_amp,d_amp)
unsigned char tipo;
int a_amp, d_amp;
/*-----------------------------------------------------------------------
  Imprime el marco superior o inferior de una pagina del reporte
  PARAMETROS:
  tipo      : Es el patron grafico a imprimir
  a_amp     : amplitud maxima de las señales analogicas
  d_amp     : amplitud maxima de las señales digitales
-----------------------------------------------------------------------*/
{
int c,i,temp;

 limpia_prlinea();
 temp=d_amp;

 for (c=I_marco; c<D_marco; c++)
    prlinea[c] = tipo;

 for (i=0; i<Max_sen; i++)
    if (existe_sen[i])
      {
       if (i>15)
           temp = a_amp;
       prlinea[org_sen[i]]=0xff;
       prlinea[org_sen[i]+temp] = 0xff;
      }

 imprime();
}


imp_nombre_sen(s_tini,a_amp,d_amp)
char *s_tini;
int a_amp, d_amp;
/*-----------------------------------------------------------------------
  Imprime las etiquetas que identifican a las señales que se van a presen-
```

```
  tar en el diagrama de tiempo
  PARAMETROS:
  s_tini     : Una cadena que contiene la representacion en caracteres del
               tiempo inicial
  a_amp      : amplitud maxima de las señales analogicas
  d_amp      : amplitud maxima de las señales digitales
-------------------------------------------------------------------------*/
{
int c, i, flag =0,tmp;
char  prtext[Max_text+1],rp_l[Max_text+1], nl=10;

 fputc(27,printer);
 fputc(48,printer);
 for ( i=0; i< Max_text-1; i++)
    prtext[i] = ' ';

 for (c=0;  c<8; c++)
   {
   for (i=0; i<Max_sen; i++)
       if (existe_sen[i])
          {
          tmp = i<16 ? d_amp : a_amp;
          prtext[(org_sen[i]+tmp/2)/12] = id_sen[i][c];
          }
   if (*(s_tini+c)!=0)
       prtext[I_marco/12] = *(s_tini+c);
   else
       flag = 1;
   if (flag)
       prtext[I_marco/12] = 32;
   if (c==7)
      {
       prtext[3]='m';
       prtext[4]='s';
      }
   sprintf(rp_l,"%.79s\n",prtext);
   fputs(rp_l,printer);
   }
}


imp_pie(paso)
double paso;
/*-------------------------------------------------------------------------
  Imprime los pies de pagina del diagrama de tiempo
  PARAMETROS:
  paso       : separacion entre dos muestra sucesivas
-------------------------------------------------------------------------*/
{
 char l_imp[Max_text];

 sprintf(l_imp,
```

```
 "    TIEMPO POR DIVISION: %-10.2f ms\n",
         paso*100.0);
  fputs(l_imp,printer);
}

imp_cuerpo(linxpag, pactual, pfin, a_amp, d_amp)
unsigned int linxpag, *pactual, pfin;
int  a_amp, d_amp;
/*-------------------------------------------------------------------------
  Imprime el cuerpo del reporte para cada pagina del diagrama de tiempo.
  Retorna 1 si llega al fin del reporte, 0 de otra manera.
  PARAMETROS:
  linxpag    : Numero de linea graficas por pagina
  pactual    : Numero de la muestra actual
  pfin       : Numero de la muestra final
  a_amp      : amplitud maxima de las señales analogicas
  d_amp      : amplitud maxima de las señales digitales
-------------------------------------------------------------------------*/
{
unsigned int frac, lin=0,prel;
int r =0,k,c=0;

 frac = 256/a_amp;
 prel = *pactual;
 for (lin = 0; lin < linxpag; lin++ )
   {
    if ( prel > pfin )
      {
       for (k=lin%8; k<8; k++)
          imp_escala(lin+c++);
       imprime();       r =1;       break;
      }
    if ( (lin%8 == 0)&&(lin) )
       imprime();
    if (lin%8 == 0 )
       limpia_prlinea();
    imp_escala(lin);
    imp_analog(prel,lin,a_amp,frac);
    imp_digit(prel,lin,d_amp);
    prel ++;
   }
 *pactual = prel;
 return (r);
}


imp_escala(lin)
/*-------------------------------------------------------------------------
  Imprime una division o indicador de coordenadas del tiempo para el cuerpo
  del diagrama de tiempo.
  PARAMETROS:
  lin        : linea grafica actual
```

```
-------------------------------------------------------------------------*/
{
int i;
 envia(I_marco,lin,1);
 envia(D_marco,lin,1);
 if (lin%100 == 0)
    for (i=I_marco - 12; i<I_marco; i++ )
       envia(i,lin,1);
}


imp_digit(prel,lin,d_amp)
unsigned int prel, lin, d_amp;
/*-------------------------------------------------------------------------
  Escribe las señales digitales
  PARAMETROS:
  prel      : Numero de la muestra actual a imprimir
  lin       : linea grafica actual
  d_amp     : amplitud maxima de las señales digitales
-------------------------------------------------------------------------*/

{ unsigned char huge *punt,mask,i,dat;
 mask = 1;
 punt = MK_FP(seg1, prel);

 for (i=0; i<16; i++ )
   {
    if (i>7)
       punt = MK_FP(seg2, prel);
    if (existe_sen[i])
      {
       dat = *punt & (mask << (i%8));
       if (dat)
          envia(org_sen[i]+d_amp,lin,1);
       else
          envia(org_sen[i],lin,1);
       }
   }
}


imp_analog(prel,lin,a_amp,frac)
unsigned int prel, lin, a_amp, frac;
/*-------------------------------------------------------------------------
  Escribe las señales analogicas
  PARAMETROS:
  prel      : Numero de la muestra actual a imprimir
  lin       : linea grafica actual
  a_amp     : amplitud maxima de las señales analogicas
  frac      : fraccion de la amplitud original
-------------------------------------------------------------------------*/
{
```

```
unsigned char huge *punt;

 if (existe_sen[16])
   {
    punt = MK_FP(seg3 , prel);
    envia(org_sen[16]+ (*punt)/frac, lin, 1);
   }
 if (existe_sen[17])
   {
    punt = MK_FP(seg4, prel);
    envia(org_sen[17]+ (*punt)/frac, lin, 1);
   }
}


envia(col,lin,dat)
unsigned int col, lin;
unsigned char dat;
/*-----------------------------------------------------------------------
  Escribe en el arreglo prlinea, que es mandado a la impresora por la
  funcion imprime
  PARAMETROS:
  col       : columna de impresion en el modo grafico
  lin       : linea grafica actual
  dat       : dato a escribir
-----------------------------------------------------------------------*/
{
 prlinea[col] |= (dat << ( 7-(lin%8) ) );
}


/* Funciones para el manejo de la impresora */

imprime()
/*-----------------------------------------------------------------------
  Recorre el arreglo prlinea, que es un arreglo de bytes, y va poniendo en
  la impresora cada elemento
-----------------------------------------------------------------------*/
{
int c,n1,n2,stat;
char ch;
unsigned char prbyte;

  sleep(2);
 /* este retardo sirve para darle un poco mas de tiempo a la impresora */
  n1 = max_col%256;
  n2 = max_col/256;
  biosprint(0,27,0);
  biosprint(0,76,0);
  biosprint(0,n1,0);
  biosprint(0,n2,0);
  stat = biosprint(2,prbyte,0);
```

```
  if (stat&0x20)
      {
       printf("ponga papel y presione enter");
       scanf("%c",&ch);
       }
  for (c=0; c<max_col; c++)
      {
       prbyte = prlinea[c];
       biosprint(0,prbyte,0);
      }
}

limpia_prlinea()
/*-----------------------------------------------------------------------
  Llena el arreglo prlinea con ceros
-----------------------------------------------------------------------*/
{
 int i;
 for (i=0; i<1000; i++)
    prlinea[i] = 0;
}

salto_pag()
/*-----------------------------------------------------------------------
 Manda a la impresora el caracter de salto de pagina
-----------------------------------------------------------------------*/
{
 fputc(12,printer);
}

inic_grprinter()
/*-----------------------------------------------------------------------
  Inicializa la impresora para modo grafico
-----------------------------------------------------------------------*/
{
 biosprint(0,27,0);
 biosprint(0,65,0);
 biosprint(0,8,0);
}

inic_pr()
/*-----------------------------------------------------------------------
  Inicializa la impresora
-----------------------------------------------------------------------*/
{
 fputc(27,printer);
 fputc(64,printer);
 fputc(27,printer);
 fputc(120,printer);
 fputc(1,printer);
}
```

BIBLIOGRAFIA

1. IBM PS/2 TECHNICAL REFERENCE, International Business Machines Corporation, 1987.
2. PS/2 VIDEO PROGRAMMING, R. Wilton, Revista Byte, ejemplar especial anual 1987, Editorial Mc.Graw-Hill.
3. CATALOGO DE DISPOSITIVOS LINEALES, National, 1988.
4. INGENERIA DE SOFTWARE, R. Farley, Editorial Mc.Graw-Hill, 1987.
5. TURBO C++ PROGRAMMERS GUIDE, Borland International Inc, 1990.
6. APUNTES DEL CURSO DE MEMORIAS Y PERIFERICOS, Fac. de Ingeniería Eléctrica, ESPOL, 1991.