59
2ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

PROGRAMA DE ANALISIS DE SEÑALES BIOELECTRICAS IMPLEMENTACION DE METODOS EN EL DOMINIO DEL TIEMPO Y LA FRECUENCIA

T E S I S

Que para obtener el título de:

INGENIERO EN COMPUTACION

P r e s e n t a :

MARIA ELENA MARTINEZ PEREZ

Director de Tesis: Dr. Augusto Fernández Guardiola

México, D. F. 1992

TESIS CON
FALLA DE ORIGEN

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

INDICE TEMATICO

PAG.

RESUMEN..... 1

I. INTRODUCCION..... 3

I.1. Clasificación de las señales..... 3
I.2. Origen de las señales bioeléctricas..... 7
I.3. Características de algunas señales bioeléctricas..... 12
I.4. Procesamiento de una señal..... 28
I.5. Métodos aplicados en el análsis de señales..... 34

II. PLANTEAMIENTO DEL PROBLEMA..... 45

III. DESARROLLO DEL PROGRAMA..... 47

III.1. Características generales..... 47
III.2. Métodos de análisis implementados..... 48
III.3. Arquitectura básica..... 68
III.4. Descripción de las funciones principales.. 79
III.5. Funcionamiento general..... 116

IV. PRUEBAS Y APLICACIONES..... 139

V. CONCLUSIONES..... 153

REFERENCIAS..... 155

APENDICE A. Características generales del PCLAB-812... A.1

APENDICE B. Listado de las principales funciones..... B.1

RESUMEN.

La presente tesis tiene como objetivo el diseño de un programa de análisis de señales bioeléctricas aplicando métodos en el dominio del tiempo y en el dominio de la frecuencia. Este programa ha sido diseñado con el propósito de resolver problemas específicos de análisis para un laboratorio de investigaciones en neurociencias, por lo que las señales a analizar son de origen bioeléctrico y generadas por el Sistema Nervioso (SN). Estas señales eléctricas son, casi siempre, débiles, del orden de μV, siendo necesario amplificarlas con dispositivos especiales.

Una señal, en general, es una función de tiempo que contiene información. Esta puede ser generada directamente desde la fuente original, pero en ocasiones contiene más elementos de los que requerimos (ruido), es decir, viene mezclada con otro tipo de información que puede no ser de nuestro interés; para lograr resaltar la señal deseada existen diversos métodos especiales de procesamiento.

A veces lo que se requiere, es transmitir la señal desde el lugar de adquisición hasta un lugar remoto para monitorear algún proceso, en este caso el procesamiento de la señal va dirigido hacia la eliminación de componentes generados durante la transmisión debido a la distorsión inherente al canal de comunicación. En otras ocasiones es necesario almacenar la información para su análisis posterior, de manera que se requieren procesos de almacenaje eficientes como compactación, corrección,

etc.

El sistema que se presenta (GRETA) está diseñado en lenguaje C estándar (Turbo C++ de Borland versión 2.0), realiza las labores de adquisición, almacenaje y procesamiento de señales bioeléctricas con fines de investigación. Se aplican algoritmos de análisis en el dominio del tiempo como promedios, integrales, autocorrelación y correlación cruzada de amplitudes, así como autocorrelación de intervalos; en el dominio de la frecuencia se aplica el algoritmo de Cooley-Tukey de la transformada rápida de Fourier, para la obtención de los espectros de potencia de la señal en estudio, también se cuenta con promedios en frecuencia.

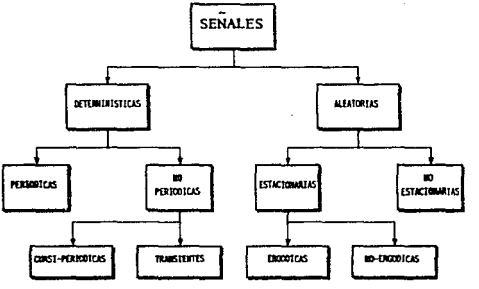
El sistema GRETA tiene un módulo de osciloscopio digital con el que es posible desplegar ocho canales a la vez, esto es de gran ayuda ya que un osciloscopio analógico común de laboratorio cuenta con sólo cuatro canales y no tiene memoria. Con este módulo es posible calibrar el sistema de registro y obtener una señal limpia y libre de artefactos antes de iniciar la adquisición.

I.INTRODUCCION.

I.1. CLASIFICACIÓN DE LAS SEÑALES.

Es importante identificar las características generales de las señales, para poder así seleccionar la herramienta de análisis adecuada.

Las señales se clasifican [5], según el diagrama de bloques de la figura 1, en dos grupos principales: las señales determinísticas y las no determinísticas o aleatorias.



Fugura 1. Clasificación de señales.

Las señales determinísticas son aquellas que pueden ser descritas mediante relaciones matemáticas explícitas. Las señales aleatorias no pueden ser descritas matemáticamente, se describen

solamente en términos probabilísticos y por medios estadísticos. Cualquier señal bioeléctrica es el resultado de un fenómeno físico -químico y es por lo tanto gobernado por ciertas leyes. Si estas leyes fueran conocidas por completo por nosotros, podríamos tener todas las señales exactamente expresadas y predecir sus valores, como nuestro interés es aplicar métodos de procesamiento de señales para poder descubrir algunas de las leyes que gobiernan ciertos fenómenos, el problema es decidir si una señal dada se considera aleatoria o determinística. Por ejemplo, cuando se analiza la señal de electrocardiograma (EKG), podríamos estar interesados en las características generales (forma de onda) del complejo QRS, entonces consideraríamos la señal como determinística, ya que su forma tiene poca variación en el tiempo; o bien, si estuviéramos interesados en estudiar los cambios del intervalo R-R, entonces se consideraría como una señal aleatoria, ya que este intervalo depende de muchos factores fisiológicos variables. Todo depende del tipo de información que se desee saber o estudiar de cierto fenómeno. Para cada caso habría una herramienta de análisis distinta.

Las señales determinísticas se dividen en dos subgrupos: las señales periódicas y las no periódicas. Las señales periódicas son señales en las que $x(t) = x(t + T)$, donde T es el período. Las señales periódicas son convenientes ya que es suficiente un período para la descripción total del fenómeno. En el dominio de la frecuencia, la descripción está dada por medio de series de Fourier, en donde sólo toman parte la frecuencia fundamental y sus

armónicos. Las señales no periódicas consisten de dos clases. Las señales cuasi-periódicas que son aquellas que no son periódicas en el sentido matemático pero tienen una descripción discreta en el dominio de la frecuencia. Esta descripción en frecuencia difiere de la periódica en que las frecuencias que participan no son armónicos de alguna frecuencia fundamental. La combinación de varias señales periódicas sin relación, crean una señal cuasi-periódica.

La señal transiente (no permanente) es una señal determinística que no tiene las propiedades discutidas anteriormente.

Las señales aleatorias son mucho más difíciles de analizar. Una señal aleatoria es una función muestra de un proceso aleatorio. Una función muestra de un proceso aleatorio se distingue de otras en su descripción temporal. Estas poseen, sin embargo, las mismas propiedades estadísticas. El conjunto completo de funciones muestra (infinito) producidas por un proceso aleatorio es llamado serie. La descripción de una señal aleatoria está dada por el conjunto de funciones de densidad de probabilidades.

Un proceso estacionario es un proceso en el cual sus propiedades estadísticas no son función del tiempo. Una clase importante de señales aleatorias estacionarias son las señales ergódicas. Se dice que un proceso aleatorio X(t) es ergódico, en la forma más general, si todas sus propiedades estadísticas pueden ser determinadas (con probabilidad uno) de una función muestra, que representa una realización posible del proceso, es decir, un proceso estrictamente estacionario aleatorio se dice que es

ergódico si todos los promedios de la serie o los promedios basados en variables aleatorias muestreadas del proceso son iguales a los correspondientes promedios en tiempo para cualquier miembro específico de la serie.

La estacionariedad y la ergodicidad son propiedades que permiten el uso de métodos de procesamiento prácticos. Un proceso que es no estacionario (y por lo tanto no ergódico) es difícil de analizar. Muchas veces uno se ve forzado a asumir que un proceso es ergódico a pesar que *a priori* sabemos que esta afirmación es falsa. Por ejemplo, cuando se procesa la señal del electroencefalograma (EEG), no se puede tener a disposición la serie completa. Sólo se tiene una función muestra. Se está, por lo tanto, forzado a asumir ergodicidad y a estimar las propiedades estadísticas requeridas a partir de muestras restringidas (en lugar de la serie completa). Como las herramientas para el procesamiento de señales no estacionarias son limitadas, generalmente se divide una señal no estacionaria en segmentos, cada uno de los cuales se considera estacionario. El tamaño de los segmentos depende de las propiedades de no estacionariedad. En señales de voz, los segmentos se eligen con duraciones de alrededor de 10 ms, mientras que en el análisis de EEG los segmentos pueden ser del orden de varios segundos.

Otra clasificación básica de señales, que es de gran significado desde el punto de vista de su procesamiento, es la de señales continuas y señales discretas en el tiempo, este criterio se aplica a señales de cualquier tipo. Las señales continuas en el tiempo son aquellas que, en general, están definidas en cualquier

instante. Las herramientas de procesamiento que se aplican a este tipo de señales son la transformada de Fourier y la de Laplace y otros métodos "analógicos". En términos de hardware, estas señales son tratadas por sistemas analógicos (filtros, amplificadores, computadoras analógicas). Las señales discretas en el tiempo son aquellas que están definidas solo en algunos instantes. Generalmente estas señales son también muestreadas en amplitud. Usualmente pensamos en señales discretas como el resultado de una señal continua que ha sido muestreada en tiempo y cuantificada en amplitud, sin embargo existen señales que son discretas por naturaleza. Estas señales son procesadas por medio de métodos discretos como son la transformada Z y la Transformada Discreta de Fourier. En términos de hardware, estas señales son tratadas por medio de sistemas digitales incluyendo, desde luego, las computadoras digitales. Los avances de la tecnología digital han hecho que el procesamiento y análisis de señales en la mayoría de los casos sea con señales discretas.

I.2. ORIGEN DE LAS SEÑALES BIOELECTRICAS.

El mecanismo de procesamiento de información más importante en el sistema biológico de los seres vivos es la red neuronal. El sistema biológico tiene muchos medios de transmisión de información. Probablemente el más importante es la transmisión de información neuronal. La Neurofisiología, que se encarga del

estudio de las funciones neurales, ha sido la llave de entendimiento de los sistemas internos de comunicación y control del sistema biológico. Básicamente la investigación en neurofisiología se basa en la habilidad para hacer mediciones de actividades químicas y electroquímicas que toman lugar en una célula aislada, o en un grupo de células.

Muchas de las funciones de las neuronas y de las células musculares son físico-químicas por naturaleza. Estas funciones, sin embargo, producen cambios en el campo eléctrico, el que puede ser monitoreado por medio de electrodos a distancia. Los así llamados potenciales bioeléctricos ayudan al neurofisiólogo en el estudio de las funciones celulares. Las mediciones directas de los fenómenos químicos, por ejemplo cambios en la concentración de iones, pueden realizarse por medio de transductores especiales (por ejemplo, electrodos selectivos a iones). Sin embargo, este tipo de mediciones son mucho más difíciles de hacer.

La fuente de las señales bioeléctricas es la célula nerviosa o la célula muscular. La unidad básica de procesamiento en el sistema neurológico es la célula nerviosa llamada neurona. La tarea de la neurona es el procesamiento, transmisión y adquisición de información. Las neuronas que están encargadas de la transmisión de información son generalmente más largas, y sirven para transmitir información hacia o desde el cerebro. Algunas otras células nerviosas especiales se han desarrollado para servir como sensores.

Existen una gran variedad de mecanismos y sensores para traducir diferentes tipos de estímulos (presión, luz, temperatura,

etc.) a señales eléctrico-químicas. El sistema nervioso central se encarga de la tarea del procesamiento y control de la información, por lo que existen numerosos tipos de neuronas con tareas distintas, sin embargo la estructura general de una neurona se puede explicar en base a la figura 2.

Las partes importantes de la neurona son [5], el cuerpo de la célula (soma), las dendritas, y el axón. El cuerpo de la célula consiste de una membrana que rodea a un fluido intracelular que contiene varios elementos necesarios para el funcionamiento de la célula. Hay mucha variación en el tamaño de las células.El diámetro puede ser tan pequeño como unas cuantas micras y tan grande como decenas de micras. Está circundado por una membrana excitable, la cual tiene un espesor en el rango de 50 a 150 Å. La membrana celular se extiende hacia varios lugares generando estructuras parecidas a ramificaciones llamadas dendritas. Estas extensiones son utilizadas para la interconexión con otras células nerviosas.

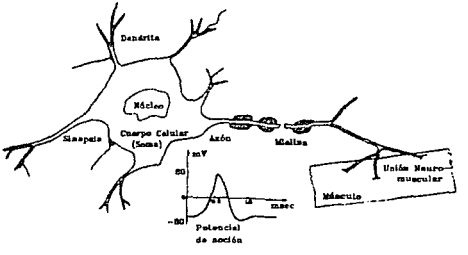


Figura 2. La célula nerviosa.

El axón sirve como mecanismo de salida de la unidad nerviosa, es una extensión de la célula con un tamaño que puede ser de alrededor de 50 μm (en la corteza cerebral) o por arriba de varios metros (en nervios periféricos de grandes mamíferos). El diámetro del axón está en el rango de menor de 0.5 μm hasta alrededor de 1 mm. La mayoría de los axones están cubiertos por fundas interrumpidas de mielina, las cuales incrementan la velocidad en la transmisión de información.

La información que llega a la neurona proveniente de otras neuronas es introducida a través de unas uniones llamadas sinapsis. Las sinapsis se localizan en las dendritas o en el soma. Las sinapsis pueden incrementar o decrementar el voltaje a través de la membrana. El funcionamiento de la célula está basado en los efectos integrativos (en tiempo y espacio) de estos cambios de potencial.

La membrana celular se puede considerar como la división entre dos medios, el fluido extracelular y el fluido intracelular. Estos dos fluidos tienen concentraciones deferentes de iones (los tres principales iones son potasio [K+], sodio [Na+] y cloro [Cl-]). La membrana tiene diferente permeabilidad a estos iones que se encuentran en las soluciones. Como resultado de la transmisión de iones, por medio de la difusión y otros mecanismos, se genera un voltaje a través de la membrana; este potencial a través de la membrana en reposo es de aproximadamente 80 mV (siendo negativa la parte interna de la célula con respecto a la parte externa).

Algunas membranas tienen características de excitabilidad. Cuando una membrana es excitada por medio de estímulos eléctricos,

mecánicos o químicos, la permeabilidad de la membrana para la transferencia de iones sufre algunos cambios. Estos cambios causan que el potencial de la membrana en reposo aumente, y se vuelva positivo por un período de tiempo corto, y después, cuando la membrana se repolariza, regresa a su potencial de reposo normal. Este curso de cambios en el potencial se conoce como potencial de acción, siendo un fenómeno que se propaga a lo largo del axón.

Las células nerviosas y musculares tienen membranas excitables. La forma y duración de los potenciales de acción difiere entre varias células. Los potenciales de acción musculares son generalmente mucho más largos en duración.

La excitación de la membrana se causa únicamente si el estímulo sobrepasa un nivel umbral (cerca de 20 mV). Una vez que se a cruzado el umbral y se provoca un potencial de acción, el umbral cambia. Seguido de la iniciación del potencial de acción, durante un cierto período (entre 1 o 2 ms), el umbral se vuelve infinito. Este período es llamado período refractario total en donde ningún nuevo potencial de acción puede ser iniciado. Después de este período el valor del umbral vuelve a su nivel normal de reposo, de acuerdo a una función de decaimiento. El período en el cual el umbral decae a su nivel de reposo es llamado período refractario relativo. Durante este período un nuevo potencial de acción puede ser provocado siempre y cuando el estímulo sea lo suficientemente fuerte para cruzar el umbral relativamente alto.

Las células nerviosas o musculares no funcionan de manera aislada sino en grandes grupos. Los efectos acumulados de todas las

mecánicos o químicos, la permeabilidad de la membrana para la transferencia de iones sufre algunos cambios. Estos cambios causan que el potencial de la membrana en reposo aumente, y se vuelva positivo por un período de tiempo corto, y después, cuando la membrana se repolariza, regresa a su potencial de reposo normal. Este curso de cambios en el potencial se conoce como potencial de acción, siendo un fenómeno que se propaga a lo largo del axón.

Las células nerviosas y musculares tienen membranas excitables. La forma y duración de los potenciales de acción difiere entre varias células. Los potenciales de acción musculares son generalmente mucho más largos en duración.

La excitación de la membrana se causa únicamente si el estímulo sobrepasa un nivel umbral (cerca de 20 mV). Una vez que se a cruzado el umbral y se provoca un potencial de acción, el umbral cambia. Seguido de la iniciación del potencial de acción, durante un cierto período (entre 1 o 2 ms), el umbral se vuelve infinito. Este período es llamado período refractario total en donde ningún nuevo potencial de acción puede ser iniciado. Después de este período el valor del umbral vuelve a su nivel normal de reposo, de acuerdo a una función de decaimiento. El período en el cual el umbral decae a su nivel de reposo es llamado período refractario relativo. Durante este período un nuevo potencial de acción puede ser provocado siempre y cuando el estímulo sea lo suficientemente fuerte para cruzar el umbral relativamente alto.

Las células nerviosas o musculares no funcionan de manera aislada sino en grandes grupos. Los efectos acumulados de todas las

células activas en la vecindad produce un campo eléctrico que se propaga a través del volumen conductor que está compuesto por los varios tejidos del cuerpo. La actividad de un músculo, o de alguna red neuronal, puede ser medida indirectamente a través de electrodos colocados, por ejemplo, en la piel. La adquisición de este tipo de información es fácil. La información, sin embargo, es muy difícil de analizar. Esta información es el resultado de toda la actividad neuronal y muscular de lugares desconocidos, transmitidas a través de un medio heterogéneo. A pesar de estas dificultades, las señales eléctricas, monitoreadas sobre la superficie de la piel, son de una enorme importancia clínica. El Electroencefalograma (EEG), el Electrocardiograma (EKG), el Electromiograma (EMG), y otras señales como estas, son rutinariamente utilizadas para el diagnóstico clínico de sistemas neuronales y musculares. La interpretación de la información está basada principalmente en la gran experiencia estadística recolectada a través de los años.

I.3. CARACTERISTICAS DE ALGUNAS SEÑALES BIOELECTRICAS.

Los rangos típicos de niveles y frecuencias de varios tipos de señales bioeléctricas se discuten en esta sección. Se da sólo un bosquejo de rangos debido a las grandes variaciones que existen en este tipo de señales, y por su fuerte dependencia con respecto a los métodos de adquisición [6].

A. Potencial de acción.

Es el potencial generado por la excitación de una membrana de un nervio o una célula muscular. El potencial de acción generado por una sola célula puede ser medido por medio de microelectrodos insertados dentro de la célula y un electrodo de referencia colocado en el fluido extracelular. El microelectrodo tiene una impedancia de entrada muy alta. Supone el uso de un amplificador con muy poco ruido y debe de usarse una capacitancia de entrada.

En la mayoría de las aplicaciones la forma del potencial de acción no es de interés. Lo que interesa son los intervalos entre espigas (ver figura 2). Se detecta el tiempo de ocurrencia de una espiga y se utilizan métodos de procesos puntuales [7][16].

Cuando los potenciales de acción de más de una célula son registrados por el electrodo, se requieren técnicas de análisis de trenes multiespigas [7]. El rango de nivel típico de un potencial de acción es de 100 mV. El ancho de banda requerido es de aproximadamente 2 kHz.

B. Electroneurograma (ENG).

El campo generado por un nervio puede ser medido sin tener que penetrar la membrana celular. Con un electrodo aguja insertado en el nervio o también electrodos de superficie colocados sobre la piel se puede medir la señal. El voltaje monitoreado no será, en general, el de un solo potencial de acción, sino la contribución de muchos potenciales de acción transmitidos a través del volumen del conductor.

La figura 3 muestra el ENG tomado con electrodos de superficie del nervio mediano. El rango de niveles de voltaje es de 5 μV a 10 mV con un ancho de banda alrededor de 1 kHz. El ENG es utilizado clínicamente para calcular la velocidad de conducción de un nervio. Esta información se requiere para la detección de daños o regeneración de fibras nerviosas. Debido a las bajas amplitudes involucradas en el monitoreo de ENG se utilizan generalmente métodos de promediación sincronizados para incrementar la relación señal-ruido.

C. Electroretinograma (ERG).

El ERG es el potencial generado por la retina. El ERG evocado es el más comúnmente usado, el cual es el potencial generado por un corto destello de luz. El ERG se utiliza clínicamente y en investigaciones oftalmológicas. Para propósitos de investigación el potencial se registra implantando un electrodo en la retina y un electrodo de referencia en cualquier lugar de la superficie que sea neutro eléctricamente. Para usos clínicos, se emplea un electrodo de córnea (generalmente hecho con lentes de contacto). El electrodo de referencia es colocado en el lobulo, en la sien, o en la frente.

Los niveles de voltaje de el ERG están en el rango de 0.5 μV a 1 mV en aplicaciones clínicas. Se adquieren voltajes mucho más altos en experimentos de investigación cuando el electrodo es implantado en la córnea. El ancho de banda requerido para el procesamiento de el ERG es alrededor de 0.2 a 200 Hz. Las técnicas de procesamiento aplicadas al ERG son principalmente promedios

sincronizados. Se aplican también métodos no lineales.

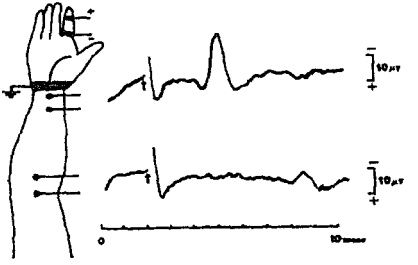


Figura 3. Potencial de acción evocado sensorial del nervio mediano en el codo y en la muñeca despues de la estimulación del dedo índice.(De Lenman, J.A.R. y Ritchie, A.E.,*Clinical Electromiography*, Pitman Medical and Scientific, London,1970). Citado por Cohen.

D.Electro-Oculograma (EOG).

El EOG es el llamado potencial córneo-retinal. Este potencial se utiliza para medir la posición del ojo, tanto para propósitos de investigación (estudio del sueño) como para usos clínicos. La señal es medida mediante el uso de un par de electrodos superficiales colocados al lado derecho e izquierdo de los ojos y arriba y abajo de los ojos. Los niveles de amplitud se encuentran en el rango de 10 μV a 5 mV. La señal requiere un rango de frecuencia de DC a 100 Hz.

E. Electroencefalograma (EEG).

El registro de la actividad eléctrica del cerebro es conocida como electroencefalograma (EEG). Es ampliamente utilizada tanto para propósitos clínicos como para investigación. Se han desarrollado métodos para investigar el funcionamiento de varias partes del cerebro a través del EEG [10][15]. Se utilizan tres tipos de registros. El registro profundo se realiza introduciendo electrodos aguja dentro del tejido neural del cerebro. Los electrodos pueden ser colocados en la superficie expuesta del cerebro, método conocido como electrocorticograma. El método más frecuentemente usado, es el registro no invasivo sobre el cráneo a través de electrodos de superficie, que registran los campos eléctricos propagados de diferentes estructuras internas del cerebro.

La investigación de la actividad eléctrica del cerebro se divide generalmente en dos modos. El primero es el registro de la actividad espontánea del cerebro, la cual es el resultado del campo eléctrico generado por el cerebro sin haberle asignado ninguna tarea específica. El segundo es el registro de potenciales evocados (EP) [7]. Estos son los potenciales generados por el cerebro como resultado de un estímulo específico (como un destello de luz, un sonido, etc.).

El registro de superficie del EEG depende de la localización de los electrodos. En registros múltiples de EEG clínicos rutinarios, los electrodos se colocan en las regiones frontal (F), central (C), temporal (T), parietal (P), y occipital

(O), con dos electrodos comunes o de referencia colocados en los lobulos, de acuerdo al mapa de la matriz 10/20 internacional que se ilustra en la figura 4. Se emplean entre 6 y 32 canales, siendo 8 o 16 los más comúnmente utilizados. Se registran las diferencias de potencial entre los electrodos. Existen tres tipos de registros: el unipolar, el de referencia de promedios, y el bipolar.

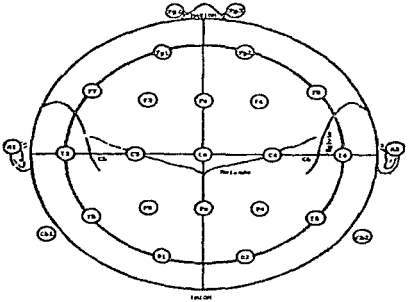


Figura 4. Mapa de la matriz 10/20 internacional. Fp= Fronto polar (Polar Frontal), F=Frontal, C=Central, T=Temporal, P=Parietal, O=Occipital. A=Referencia, Pg=Faringeo y Cb=Cerebelar. [11]

El rango de ancho de banda para el EEG de cráneo es de DC a 100 Hz, con la mayor distribución de potencia en el rango de 0.5 a 60 Hz. Las amplitudes en el EEG del cráneo se encuentran en el rango de 2 a 100 μV. La densidad espectral de potencia del EEG varía grandemente en diferentes estados. El análisis en frecuencia del EEG ha sido la herramienta principal de procesamiento en el

diagnóstico neurológico durante muchos años [8][10][15]. Ha sido utilizada en el diagnóstico de epilepsia, lesiones cerebrales, trastornos psiquiátricos, desordenes del sueño, entre otros. La mayor porción del espectro del EEG ha sido subdividida en bandas:

Rango delta - La parte del espectro que ocupa el rango de 0.5 a 3.5 Hz es el rango delta. Las ondas delta aparecen en niños pequeños, en el sueño profundo, y en algunas enfermedades cerebrales. En el adulto alerta, la actividad delta es considerada como anormal.

Rango teta- El rango teta es la parte del espectro que ocupa el rango de frecuencias de 4 a 7.5 Hz. Los componentes transientes de la actividad teta han sido encontrados en sujetos adultos normales durante el estado de alerta. La actividad teta ocurre principalmente en las áreas temporal y central y es más común en los niños.

Rango alfa - El rango alfa es la parte del espectro que ocupa frecuencias desde 8 a 13.5 Hz. Este tipo de ritmos es común en sujetos normales, mejor vistos cuando el sujeto esta despierto, con los ojos cerrados, bajo condiciones de relajación. Se cree que la fuente de las ondas alfa se encuentra en los lobulos occipitales (ver figura 5).

Rango beta - El rango beta es la parte del espectro que ocupa el rango de frecuencias entre 14 a 22 o más Hz. El ritmo beta es registrado en sujetos adultos normales principalmente en las regiones precentrales, pero también aparecen en otras

regiones. El rango beta ha sido subdividido a su vez en dos: Beta I es el rango de altas frecuencias y Beta II es el rango de bajas frecuencias. El rango Beta II se presenta durante la activación intensa del Sistema Nervioso Central, mientras que Beta I disminuye en dicha activación. Los sedantes y algunos barbitúricos causan un incremento en la actividad beta con amplitudes generalmente de 100 μV.

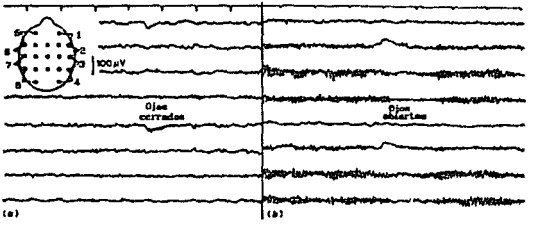


Figura 5. Registros de EEG. (a) Sujeto con completa ausencia de ondas alfa, (b) sujeto con ondas alfa que disminuyen por 1 s seguida de ojos abiertos.(De Kiloh, L.G.,McComas, A.J., Osselton,J.W., and Upton,A.R.M., *Clinical Electroencephalography*, 4th ed., Butterworths, London, 1981). Citado por Cohen.

El análisis en el dominio del tiempo también se utiliza en el procesamiento del EEG para detectar ondas de corta duración y bajo voltaje. Esto es aplicado generalmente en el análisis del sueño.

Se pueden ver muchas anormalidades a través del EEG. La epilepsia es una condición en donde tienen lugar descargas neuronales sin control en algún lugar del Sistema Nervioso Central. Esto ataca la actividad de varios músculos y otras funciones involuntariamente mientras inhibe otras. Se conocen muchos tipos de epilepsia, entre estos están el "gran" y "pequeño mal", la epilepsia mioclónica, y otras (ver figura 6) [10][15].

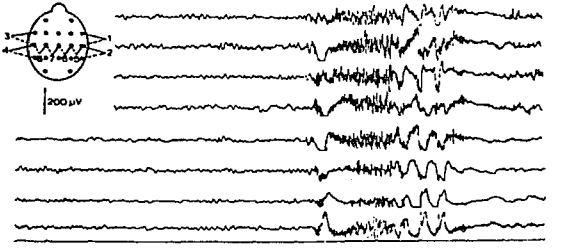


Figura 6. Crisis epiléptica generalizada.(De Kiloh, L.G., McComas, A.J.,Osselton,J.W.,y Upton,A.R.H., *Clinical Electroencephalography*, 4th ed., Butterworths, London,1981) Citado por Cohen.

F. Potenciales Evocados (EP).

La actividad eléctrica del cerebro evocada por estímulos sensoriales se conoce como Potenciales Evocados (EP) o respuestas evocadas (ER). Generalmente se mide sobre la región del cerebro que corresponda con la modalidad del estímulo. Existen principalmente

tres tipos de potenciales evocados que se utilizan con mayor frecuencia:[7]

- *Potenciales Evocados Visuales (VEP)*. El VEP se registra sobre el cráneo en la región del lobulo occipital. Los estímulos son destellos de luz o patrones visuales. El VEP tiene una amplitud en el rango de 1 a 20 μV con un ancho de banda de 1 a 300 Hz. La duración de un VEP es de aproximadamente 200 ms. Los VEP se han utilizado para el diagnóstico de esclerosis múltiple (el nervio óptico se ve afectado comúnmente por esta enfermedad), para diagnosticar la ceguera de color, para revisar la agudeza visual. La figura 7 muestra un VEP típico [18].

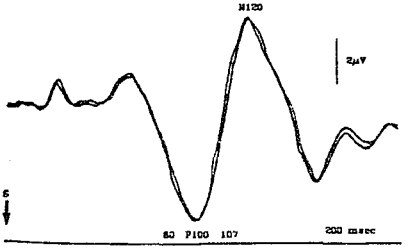


Figura 7. Respuesta visual promedio: ancho de banda de 2 a 300 Hz, promedio de 64 respuestas.(De Kiloh,L.G., McComas,A.J., Osselton,J.W., y Upton,A.R.H., *Clinical Electroencephalography*, 4th ed., Butterworths, London, 1981). Citado por Cohen.

- *Potenciales Evocados Somatosensoriales (SEP)*. El SEP es registrado con electrodos de superficie colocados sobre la corteza sensorial. Los estímulos pueden ser eléctricos o mecánicos aplicados a la piel. La duración del SEP cortical es de alrededor de 25 a 50 ms, con un ancho de banda de 2 a 3000 Hz. El SEP subcortical es mucho más largo, dura por lo menos 200 s. La figura 8 muestra un SEP cortical y subcortical. Los SEPs son utilizados para proveer información concerniente a la ruta en la columna dorsal entre las fibras nerviosas periféricas y la corteza.

- *Potenciales Evocados Auditivos del tallo cerebral. (AEP)*. Los AEPs son registrados a través de electrodos colocados en el vertex. Los estímulos auditivos pueden ser un chasquido, una serie de tonos, ruido blanco, entre otros. El AEP se divide en el potencial primero (con latencia alrededor de milisegundos), el potencial temprano (8 ms), el potencial medio (de 8 a 50 ms),y el potencial tardío (de 50 a 500 ms). Los primeros 10 ms de respuesta están asociados con la actividad del tallo cerebral. Estos potenciales evocados auditivos del tallo cerebral son de muy baja amplitud (alrededor de 0.5 μV). El AEP tiene un ancho de banda de 100 a 3000 Hz. Los AEPs son utilizados para revisar deficiencias auditivas, especialmente en niños. La figura 8 muestra un típico AEP cortical y subcortical.

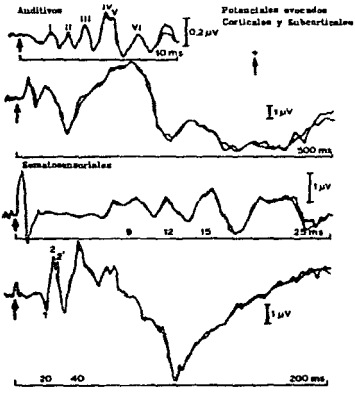


Figura 8. Potenciales evocados auditivos y somatosensoriales, subcorticales (10 ms) comparadas con respuestas equivalentes corticales (200 a 500 ms): ancho de banda subcortical de 150 a 1500 Hz, 2000 promedios; ancho de banda cortical de 2 a 75 Hz, 64 promedios. (De Kiloh, L.G., McComas,A.J., Osselton,J. W., y Upton,A.R.M., *Clinical Electroencephalography*, 4th ed., Butterworths, London, 1981). Citado por Cohen.

G. Electromiograma (EMG).

El EMG es el registro del potencial eléctrico generado por el músculo. La actividad muscular puede ser monitoreada a través de electrodos colocados en la piel. La señal recibida da información concerniente a la actividad eléctrica total asociada con la contracción muscular. Existen tres tipos de señales de EMG:

- *Electromiograma de una Fibra Aislada*. Es el potencial de acción registrado de una fibra muscular aislada con una duración de alrededor de 1 ms, con amplitudes de algunos milivolts. El ancho de banda utilizado es de 500 Hz a 10 kHz.

- *Potenciales de Acción de Unidad Motora*. Al conjunto que consiste en una célula nerviosa, una fibra nerviosa, uniones neuromusculares, y las fibras musculares se la llama Unidad Motora. Los potenciales de acción registrados por este conjunto, que se hace a través de electrodos anidados concéntricos, se conocen como potenciales de acción de la unidad motora. La duración de estos potenciales es de alrededor de 2 a 10 ms, con amplitudes en el rango de 100 μV a 2 mV. El ancho de banda requerido para este proceso es de 5 Hz a 10 kHz. La figura 9 muestra un tren de estos potenciales.

- *Electromiograma de superficie*. Es la adquisición no invasiva de EMG por medio de electrodos de superficie. Este registro proporciona una información a groso modo sobre el músculo a investigar. Las amplitudes dependen del músculo a ser investigado y de los electrodos utilizados, un rango normal es de 50 μV a 5 mV. El ancho de banda requerido (para músculos del esqueleto) es de 2 a 500 Hz (para músculos lisos es de 0.01 a 1 Hz). La figura 10 muestra un ejemplo de registro de EMG de la superficie respiratoria.

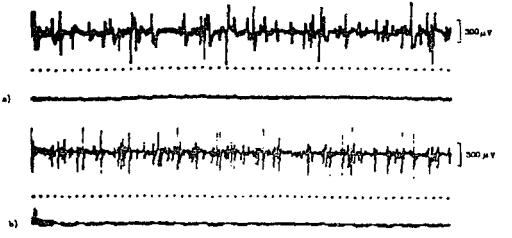


Figura 9. Registro de potenciales de unidad motora normales registrados del (a) primer músculo dorsal interóseo y (b) músculo frontal. (De Lenman, J.A.R., y Ritchie, A.E., *Clinical Electroencephalography*, Pitman Medical and Scientific, London, 1970) Citado por Cohen.

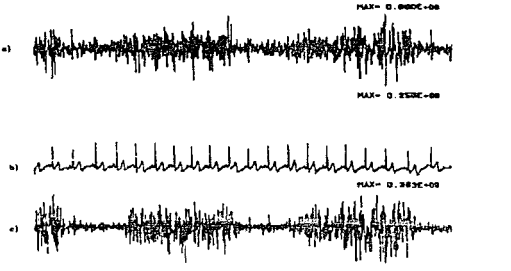


Figura 10. EMG y EKG durante la respiración. (a) EMG del músculo diafragma, (b) EKG, (c) EMG del músculo intercostal.[40]

H. Electrocardiograma (ECG, EKG).

El EKG es la actividad eléctrica registrada del corazón. La actividad mecánica del funcionamiento del corazón está ligada con la actividad eléctrica. El EKG es una herramienta importante para el diagnóstico del funcionamiento del corazón. Una señal típica del EKG se muestra en la figura 11.

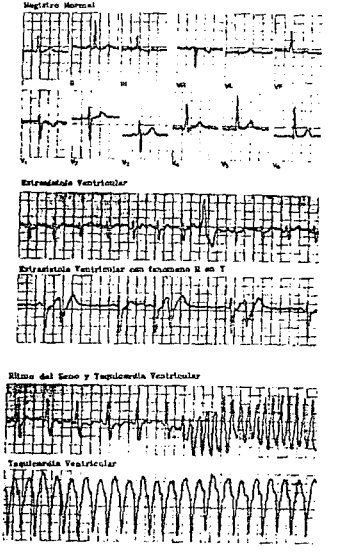


Figura 11. Electrocardiograma.[6]

El ciclo eléctrico del corazón inicia en el nodo seno auricular (SA) en el lado derecho la aurícula; que es un conjunto de nervios no estables. El nodo SA pasa el corazón. Los impulsos del nodo SA causan la contracción de la aurícula, generando la onda p del EKG. Los impulsos viajan, conducidos por fibras, entre la aurícula y el nodo ventrículo-auricular (AV) el cual controla la transmisión de impulsos entre la aurícula y los ventrículos. El tiempo de conducción aurículo-ventricular es del orden de 120 a 220 ms. Un sistema especial de conducción, formado por los facículos de His y de Purkinje, transmiten los impulsos a la parte baja y externa de los ventrículos. La contracción de los ventrículos produce la acción de bombeo del corazón, generando el complejo QRS en el EKG. Alrededor de 150 ms después los ventrículos se repolarizan, causando la onda T del EKG. La repolarización de la aurícula, raramente es vista en el EKG. En los casos raros en que se ve, aparece entre las ondas P y Q del EKG, y es llamada onda TA. Una onda adicional, llamada onda U, se registra algunas veces después de la onda T. Se cree que esto se debe a la repolarización de los músculos ventrículo papilares.

El ritmo cardíaco, o la velocidad de los latidos del corazón, es un proceso aleatorio. Generalmente se mide a través de los intervalos R-R. Durante el sueño la velocidad del corazón disminuye, mientras que se acelera durante el ejercicio, stress emocional, o fiebre. Los disturbios del ritmo, como la arritmia, pueden ocurrir bajo muchas condiciones anormales. Algunas veces una porción del miocardio descarga independientemente, causando un

latido de corazón fuera de la secuencia SA normal; esto se conoce como extrasístole o contracción preventricular (CPV). Cuando las descargas independientes continúan, el corazón puede entrar en un estado de fibrilación aurículo o ventricular. Algunas veces la causa de este fenómeno es un bloqueo en las rutas normales neuronales del corazón (por ejemplo, el facículo de His o las fibras de Purkinje). La figura 11 muestra el registro de varios EKGs patológicos.

Un EKG convencional consiste en complejos PQRST con amplitudes de varios milivolts. Se procesa generalmente en bandas de frecuencias de 0.05 a 100 Hz, en donde se incluye la mayoría de la energía del EKG.

I.4. PROCESAMIENTO DE UNA SEÑAL.

La mayoría de las señales de interés en la biomedicina son continuas (analógicas), y están definidas sobre una variable de rango continuo (el tiempo). Es importante, sin embargo, analizar las señales discretas, las cuales están definidas en instantes discretos. La tecnología digital moderna, tanto en términos de hardware como de software, hacen del procesamiento discreto mucho más ventajoso sobre el procesamiento analógico. Las ventajas son debido a que generalmente es posible convertir una señal analógica a una señal discreta de manera que pueda ser aplicado el procesamiento discreto. La conversión se hace a través de un sistema analógico-digital (A/D) que muestrea y cuantifica la señal

en tiempos discretos. Generalmente el muestreo se realiza uniformemente, pero algunas veces también es utilizado el muestreo no uniforme [1].

Una de las principales tareas en el procesamiento de señales es la aplicación de filtros, para la estimación de varios parámetros, y para realizar transformaciones a las señales (por ejemplo, la transformada de Fourier). Cuando los resultados de un procesamiento no se requieren inmediatamente después de la adquisición de la señal, se utilizan métodos de procesamiento fuera de línea (off-line). Cuando los resultados se requieren durante o inmediatamente después de la adquisición, se utilizan métodos de procesamiento en tiempo real o en línea (on-line).

Dependiendo de la aplicación, son de gran importancia para el procesamiento de la señal, el tiempo de procesamiento y el tamaño de la memoria requeridos. El procesamiento fuera de línea puede ser realizado por computadoras de propósito general. Mientras que el procesamiento en tiempo real requiere de máquinas especiales dedicadas, por ejemplo, un sistema para realizar correlación rápida, maquinas especializadas en hacer transformadas de Fourier por hardware, y en general circuitos de hardware especializado, como el llamado "softwire" (software alambrado) [17].

En base a todo lo anterior se describe un diagrama de bloques representativo del equipo típico en un laboratorio de análisis y procesamiento de señales en la figura 12.

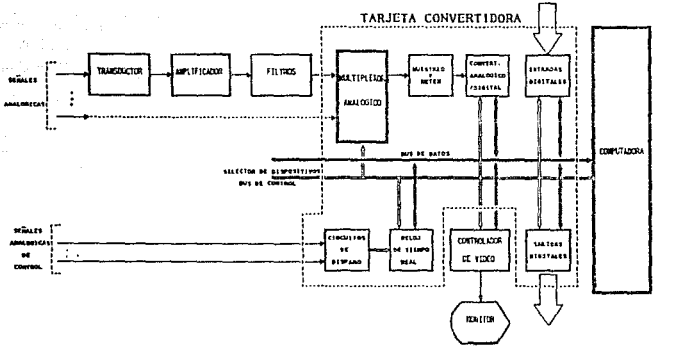


Figura 12. Sistema típico de adquisición, conversión y procesamiento de datos. [24]

La medición computarizada y los sistemas de control pueden ser divididas en dos familias: los sistemas de adquisición de datos y sistemas de control de procesos digitales directos. En ambos casos los transductores toman parámetros físicos como amplitudes, intervalos, temperaturas, tensiones, o posición y los convierten en voltajes o corrientes eléctricas. Una vez en forma eléctrica, todo el resto del procesamiento de la señal es realizado por circuitos electrónicos. La señal primero pasa a través de un filtrado

analógico y un acondicionador. Después la señal es convertida a forma digital y luego adquirida por la computadora para el procesamiento digital. Los resultados producidos por la computadora son utilizados como retroalimentación en un sistema de control o solo son desplegados en un monitor. Si los elementos de control o las unidades de despliegue son dispositivos analógicos, la salida de la computadora digital debe pasar a través de un convertidor digital-analógico (D/A), produciendo así señales analógicas manejables. Este sistema está compuesto por un número de unidades diferentes, conectadas a la computadora a través de un canal de comunicación (line bus) [2].

El canal de comunicación se divide en dos canales: canal de control y selección de dispositivos (command bus), y el canal de entrada y salida de datos (data bus).

El canal de control es utilizado por la computadora para seleccionar un solo dispositivo a la vez para la comunicación, y para seleccionar la operación a realizar por el dispositivo seleccionado (como leer, escribir, entrada ,salida ,etc.).

El canal de datos lleva el dato desde el registro de datos del dispositivo periférico seleccionado hacia la computadora, y desde la computadora hacia el registro de datos del dispositivo periférico seleccionado.

Cada dispositivo interfase está compuesto por un registro de datos y compuertas entrada/salida, un selector de dispositivo, un control decodificador, y banderas flip-flop. Las banderas flip-flop suelen estar conectadas a las entradas de interrupción. Las

banderas flip-flop de un dispositivo en particular informan a la computadora si el dispositivo está listo u ocupado para la comunicación.

El sistema mostrado en la figura 12 está compuesto por las siguientes unidades: transductor, amplificador, filtro, multiplexor analógico, circuitos de muestreo-retén, convertidor analógico-digital, convertidor digital-analógico, desplegado en tubo de rayos catódicos (osciloscopio) o desplegado en monitor de computadora, reloj de tiempo real, circuito de disparo, entradas digitales, salidas digitales y computadora [9].

Amplificadores y Filtros. La primera parte del sistema, para la adquisición y procesamiento de datos, consiste en extraer la señal que va a ser medida. El inicio del procesamiento de la señal se realiza con un amplificador y posiblemente un filtro analógicos. El propósito del amplificador es realizar una o más de las siguientes funciones: aumentar el nivel de la señal, atenuar la señal, convertir la señal de corriente a voltaje, o separar la señal diferencial del ruido común. Para la mayoría de los dispositivos analógicos, como el muestreador-retén y el convertidor analógico-digital, los niveles de voltaje deseados en la salida del amplificador son de 5 a 10 V en escala completa.

Seguido de los amplificadores puede ser necesario el uso de filtros. Los filtros se utilizan principalmente por dos razones: 1.- para reducir el ruido y para incrementar la relación señal/ruido, y 2.- para limitar el ancho de banda de la señal y para evitar los componentes de alta frecuencia en la señal, si

estos componentes no son necesarios. De esta manera la señal puede ser muestreada a una velocidad moderada, y el procesamiento de datos es más barato.

Multiplexores Analógicos y Circuitos Muestreador-Retán. Los multiplexores analógicos son utilizados para compartir el convertidor analógico-digital entre varios canales analógicos diferentes. Los multiplexores analógicos tienen muchas entradas analógicas y una sola salida. El multiplexor está compuesto por un número de interruptores analógicos. Cada interruptor conecta una entrada analógica a la salida común. Los interruptores pueden ser seleccionados por un código digital de entrada a través del canal de la computadora (por software). Sólo una entrada es conectada a la salida en cada vez. Generalmente los canales de entrada están conectados secuencialmente a la salida del multiplexor.

La salida del multiplexor analógico entra a un circuito muestreador-retén el cual muestrea la salida del multiplexor un instante de tiempo específico y luego retiene el nivel de voltaje en su salida hasta que el convertidor analógico-digital realiza la operación de conversión.

Entradas y Salidas Digitales. En algunos casos el dispositivo periférico es digital. En estas circunstancias los datos ya están en forma digital, y es posible una comunicación directa con la computadora. Para eliminar la necesidad del diseño de una interfase, muchos sistemas de laboratorio tienen una unidad de entradas digitales y una unidad de salidas digitales, como se muestra en el diagrama de la figura 12. Los componentes de

interfase, como el selector de dispositivos, las banderas flip-flop, los registros de datos, y compuertas, ya están construidos dentro de la unidad. Las líneas de entrada y salida digitales están en directa disposición para el usuario.

Si el usuario tiene un instrumento que genera directamente datos digitales, las líneas de salida del instrumento pueden ser conectadas directamente a las líneas de entrada digital.

Si el usuario tiene un dispositivo de control o despliegue que opera con datos digitales, las líneas de salida digital pueden ser conectadas directamente a las líneas de entrada de dicho dispositivo.

I.5. METODOS APLICADOS EN EL ANALISIS DE SEÑALES.

La actividad de los biólogos, neurofisiólogos, y otros científicos e ingenieros no está restringida a la simple descripción de trenes de espigas o formas de onda continuas que constituyen las señales básicas en el proceso de comunicación biológica. El desarrollo de gran variedad de técnicas computacionales, el alcance de modelos computacionales para enmarcar y probar hipótesis y desarrollar nuevas teorías, todo esto ha engendrado nuevas interpretaciones, nuevas preguntas, y nuevas direcciones en la investigación cuantitativa del proceso de comunicación biológica.

La comunicación es un proceso de intercambio de mensajes entre dos puntos utilizando algún tipo de señal, como se representa en la

figura 13.

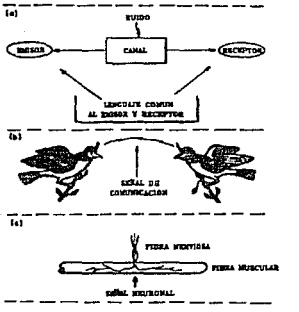


Figura 13. Sistema de comunicación. (a) Intercambio de información entre emisor y receptor a través de un canal ruidoso. (b) Comunicación externa entre dos organismos vivos. (c) Comunicación interna a través de procesos neuronales. [24]

Durante la transmisión de información se suma a la señal de interés una señal de ruido en el canal de comunicación. Una de las tareas en el estudio de la comunicación biológica es la distinción entre la señal de interés y la señal de ruido. La señal biológica de comunicación normalmente se distorsiona por fluctuaciones aleatorias, que son causadas por la actividad biológica en curso, por el medio ambiente, y finalmente por la instrumentación utilizada. Debido a estos factores, el método que más comúnmente se utiliza en este dominio es el de análisis estadístico [20][17].

Desde el punto de vista de las señales, la comunicación biológica puede dividirse en dos grupos: procesos puntuales (discretos), y procesos continuos [24].

Desde el punto de vista de sistema, la comunicación biológica puede también dividirse en dos grupos: la comunicación interna entre procesos neuronales, y la comunicación externa entre dos organismos vivos; en ambos casos las técnicas computacionales son las mismas.

Existen tres funciones que son frecuentemente utilizadas en el análisis biológico de datos: la distribución de probabilidad, la función de correlación (ambas en el dominio del tiempo), y el espectro de potencia (en el dominio de la frecuencia). Estas funciones describen a una señal aleatoria, en amplitud, forma, y frecuencia, como se hace con las señales determinísticas.

La distribución de probabilidades describe las propiedades de amplitud de un dato aleatorio. Como el dato fluctúa de una manera aleatoria, la distribución de probabilidades muestra cuáles rangos de amplitud son más probables, y cuáles rangos de amplitud son menos probables.

La función de correlación describe la dependencia de una señal en un instante con una señal en otro instante. La diferencia entre los dos instantes se llama intervalo de tiempo o intervalo de correlación. La función de correlación se grafica como función del intervalo de tiempo. El valor máximo de la función de correlación muestra el intervalo de tiempo para el cual es mayor la influencia entre las dos señales.

El espectro de potencia muestra la composición de frecuencias de una señal. Toda señal puede ser aproximada con la suma de señales senoidales de diferentes frecuencias, amplitudes y fases. El espectro de potencia muestra las amplitudes de las ondas seno en función de sus frecuencias.

Procesos puntuales. [16] Los potenciales de acción o espigas generadas por una neurona pueden ser considerados como procesos puntuales para los cuales una de las características básicas es el histograma de intervalos entre eventos consecuentes. Esta es una estimación de la función de densidad de probabilidad considerando los intervalos mencionados como una muestra de una variable aleatoria (ver figura 14). Si la serie de eventos estudiados corresponden al proceso renovado, entonces el conocimiento de esta densidad por sí sola es suficiente para la descripción del proceso.

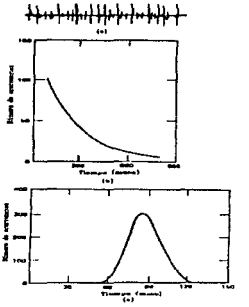


Figura 14. (a) Ejemplo de un potencial de acción de una neurona aislada registrada con un microelectrodo. (b) Ejemplo de un histograma de intervalos entre espigas consecutivas de la actividad espontánea de una neurona aislada. (c) Otro ejemplo de un histograma de intervalos. [24]

El segundo paso en la descripción, es la aproximación de este histograma experimental a algún tipo general de distribución como el Gaussiano o exponencial caracterizados por sólo algunos parámetros. El mismo problema se encuentra para decidir si dos histogramas experimentales son producidos por el mismo generador (homogeneidad de la población). Algunas veces es suficiente con pruebas estadísticas simples, como por ejemplo, el valor promedio estimado por el promedio aritmético, la variancia estimada por la desviación cuadrática, o el coeficiente de asimetría o exceso, para realizar la detección de cambios espontáneos en la actividad observada (figura 15).

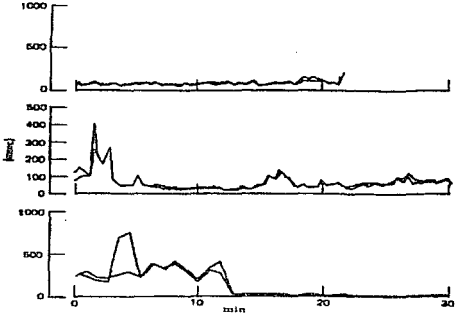


Figura 15. Promedio aritmético (línea gruesa) y desviación estándar (línea delgada) de intervalos entre espigas de la actividad unitaria espontánea de una neurona aislada como función del tiempo después de la aplicación de una dosis de droga particular. Se dan tres ejemplos. [24]

Para detectar la influencia de la estimulación repetitiva en la actividad observada se utiliza frecuentemente el llamado histograma extendido ("dwell histogram"). Se parece al estimador de la función de densidad de probabilidad condicionada en la que se considera que ocurre un evento en cierto tiempo, si el estímulo es aplicado en el tiempo cero. Cuando la actividad observada no está influenciada por el estímulo y si se satisfacen otros supuestos, el histograma extendido no se distribuye uniformemente sobre el período interestímulo. Por lo tanto en algunos casos, la evaluación del histograma extendido, que se refiere a la aceptación o rechazo de la hipótesis sobre la influencia del estímulo en la actividad, se basa solamente en una aproximación (figura 16).

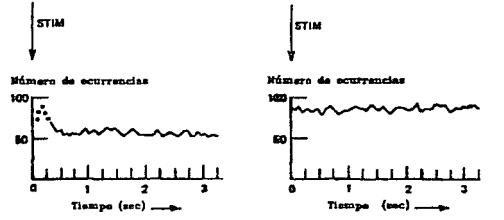


Figura 16. Histograma extendido de dos neuronas. (a) Muestra la estricta dependencia al estímulo. (b) No existe dependencia al estímulo. [24]

Para detectar las dependencias de periodicidades escondidas en una serie de eventos se utiliza el estimador de la función de autocorrelación. Este estimador de la función de densidad de

probabilidad condicionada, es parecido al histograma extendido, pero el papel del estímulo satisface consecuentemente a cada evento analizado en el caso anterior (figura 17).

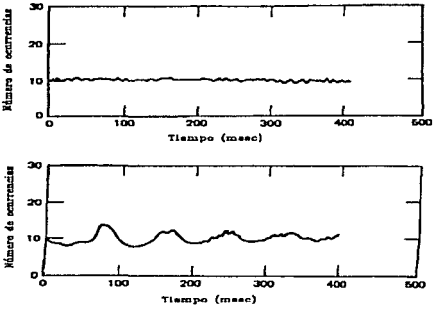


Figura 17. Función de autocorrelación de la actividad impulso espontánea de la misma neurona utilizada en la figura 14. [24]

Para detectar la mutua dependencia de dos series de eventos se utiliza la función de correlación cruzada [25], donde los eventos de un proceso son considerados como estímulos; o la distribución de ocurrencias hacia atrás o hacia adelante en el tiempo [7][13].

Procesos continuos. Un ejemplo típico de una señal neurofisiológica continua es el EEG. Contrariamente a la actividad registrada de una unidad neuronal tomando potenciales eléctricos de una pequeña vecindad de una célula a través de microelectrodos, la actividad del EEG se registra sobre la superficie de la piel o desde la profundidad del cerebro con electrodos relativamente

largos. La presencia de diferentes tejidos entre los generadores locales y la actividad eléctrica del cerebro, influencian en el potencial observado en la superficie.

Una de las estadísticas más simples de primer orden utilizadas tanto en actividad espontánea como en la evocada del EEG, es el histograma de amplitudes que bajo ciertas condiciones es el estimador de primer orden de la función de densidad de probabilidad (figura 18). Se evalúan algunos parámetros que caracterizan la forma de este histograma; por ejemplo, en base al coeficiente de asimetría se puede detectar automáticamente la forma típica del EEG. En procesamientos posteriores se puede tratar de aproximar estos histogramas con algunas distribuciones teóricas (como por ejemplo una normal) para simplificar su descripción.

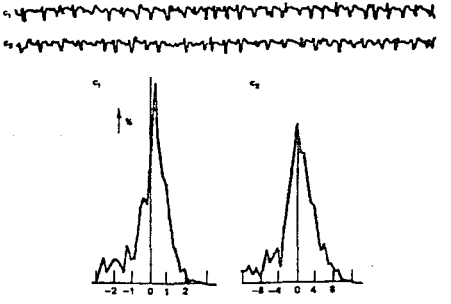


Figura 18. Histograma de amplitud del EEG junto con las partes del registro EEG original. Frecuencia de muestreo es de 60 muestras/segundo. [21]

La estimación del promedio de la serie en un tiempo elegido relativo al estímulo, representa la estimación estadística más útil en la evaluación de la actividad evocada del EEG. Esta poderosa herramienta para la detección de los llamados potenciales evocados, que son, en el EEG, los cambios de actividad eléctrica lenta de una parte determinada del cerebro por un estímulo dado. Este método permite extraer una respuesta de muy baja amplitud (la señal) escondida en la actividad espontánea aleatoria (ruido) (figura 19).[18] Por otro lado, procesar la evaluación de la desviación estándar de cada uno de los puntos del potencial evocado, representa un ejemplo de la estadística simple de segundo orden. Esta desviación estándar juega un papel muy importante en la estimación de la validez de los potenciales evocados evaluados simultáneamente.

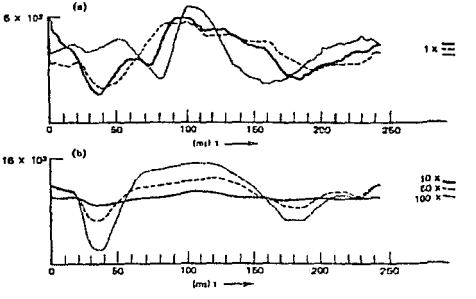


Figura 19. (a) Registro superimpuesto individual de la actividad de EEG seguida del estímulo. (b) Resultado del potencial evocado sumando los registros individuales mostrados. [20]

Otros de los métodos considerados como estadística de segundo orden en el procesamiento del EEG son la autocorrelación, la correlación cruzada y la correspondiente función de densidad espectral. Los correlogramas son evaluados por convolución, y la función de densidad espectral se evalúa directamente con filtros digitales o utilizando la transformada rápida de Fourier [9][13].

Tanto la función de correlación como la función de densidad de probabilidad, se pueden considerar iguales en la medida en que se trate de un proceso estocástico estacionario. Se presentan algunos ejemplos de correlogramas y espectrogramas de la evaluación del EEG en las figuras 20 y 21.

La relación de la potencia de la actividad del EEG en diferentes bandas de frecuencias, es en algunos casos muy importante desde el punto de vista clínico, por lo que la función de densidad de probabilidad será pronto una característica importante en clínica [15].

Por medio del análisis espectral se pueden reconocer los diferentes estados del sueño o los diferentes niveles de vigila.

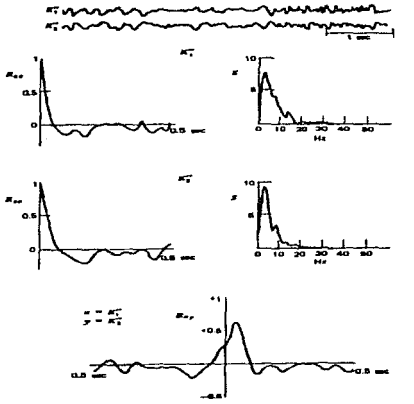


Figura 20. Ejemplo de dos registros originales de EEG tomados de dos partes diferentes del cerebro y sus autocorrelogramas (Rxx), sus espectrogramas (S), y sus correlogramas cruzados (Rxy). [24]

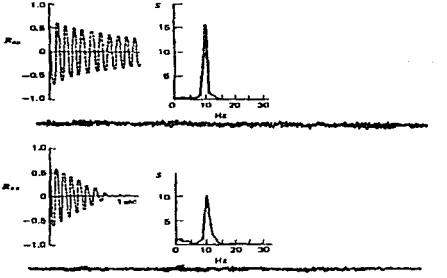


Figura 21. Ejemplos de actividad EEG periódica y sus correspondientes autocorrelogramas (Rxx) y espectrogramas (S). [24]

II. PLANTEAMIENTO DEL PROBLEMA.

En base al marco teórico anterior, se plantea la necesidad de diseñar un sistema de análisis de señales bioeléctricas que tenga la capacidad de adquirir, almacenar y analizar, con diferentes métodos tanto en tiempo como en frecuencia, series de señales.

La adquisición debe consistir en tomar series de señales con *n* número de barridos. Un barrido es, en este caso, la adquisición de 16 canales con 256 puntos por canal.

El almacenaje debe permitir guardar en disco la serie adquirida en uno de dos formatos, ASCII o BINARIO, esto es con el propósito primero, de hacer un análisis posterior de la información adquirida con el sistema GRETA y segundo, de hacer portables las series adquiridas; es decir, que las series adquiridas con el sistema GRETA puedan ser utilizadas por otros sistemas para su análisis.

Finalmente, para el análisis se deben implementar métodos en:

- Dominio del tiempo. Promedios de barridos, medición de tiempo y voltaje, integral, autocorrelación y correlación cruzada de amplitudes, de un canal seleccionado; como procesos continuos. Y obtener la autocorrelación de intervalos de un canal, como proceso puntual.

-Dominio de la frecuencia. Espectros de potencia utilizando el algoritmo de Cooley-Tukey (1965)[9] de la transformada rápida de Fourier, y promedios de barridos.

III. DESARROLLO DEL PROGRAMA.

III.1. CARACTERISTICAS GENERALES.

El sistema GRETA está diseñado en lenguaje C estándar utilizando el compilador Turbo C++ versión 2.0 de Borland, tiene un tamaño de 4,194 líneas de código fuente y ocupa 137 Kbytes de código ejecutable en disco.

El sistema GRETA está diseñado bajo un entorno gráfico con uso de mouse, por lo que corre en una PC XT compatible con adaptador gráfico VGA y un puerto serial.

Utiliza una tarjeta convertidora PC-LabCard-812, la cual contiene un convertidor de 12 bits con 16 canales A/D y 2 canales D/A; las principales características técnicas de esta tarjeta se anexan en el apéndice A.

Se utilizó lenguaje C ya que actualmente es el lenguaje más eficiente para la programación de sistemas de este tipo. Respecto a la generación de código se utilizó el modelo HUGE de memoria el cual permite tanto bloques de código como estructuras de datos mayores de 64 Kbytes. Todos los apuntadores utilizados son de 32 bits normalizados.

Respecto a la interfase gráfica se utilizó el estándar BGI y se diseñaron librerías de funciones para la generación de ventanas y otras entidades gráficas como menús y el manejo del mouse.

El programa provee un detector para coprocesadores INTEL IX87, que en caso de existir lo utiliza, y en caso de no existir lo emula.

Las rutinas críticas en cuanto a tiempo de ejecución se escribieron directamente en ensamblador utilizando comandos INLINE, que permiten la introducción de código en ensamblador en cualquier programa escrito en C.

III.2. METODOS DE ANALISIS IMPLEMENTADOS.

El objetivo de esta sección es hacer una descripción formal de los métodos de análisis implementados en el sistema GRETA.

DOMINIO DEL TIEMPO.

Promedio.[20] El promedio o valor medio estimado es una de las pruebas estadísticas conocidas como de primer orden para procesos aleatorios. El promedio de una variable aleatoria X se define como

$$\overline{X} \quad o \quad E(x) \triangleq \sum_{\forall i} \alpha_i P_x(\alpha_i) \qquad \ldots (1)$$

para una variable aleatoria discreta, y

$$\overline{X} \quad o \quad E(x) = \int_{-\infty}^{\infty} \alpha f_x(\alpha)\, d\alpha \qquad \ldots (2)$$

para una variable aleatoria continua. Basándonos en la ecuación 1, si consideramos que un experimento se realiza N veces en un fenómeno aleatorio y que α_1 ocurre N_1 veces, α_2 ocurre N_2 veces, y así sucesivamente, terminando con α_m ocurriendo N_m veces, entonces el valor promedio denominado $\overline{X}$, es:

$$\bar{X} = \frac{\alpha_1 N_1 + \alpha_2 N_2 + \cdots + \alpha_m N_m}{N} \qquad \ldots (3)$$

o

$$\bar{X} = \sum_{i=1} \alpha_i \frac{N_i}{N}$$

$$\bar{X} = \sum_{\forall i} \alpha_i P_x(\alpha_i) \quad como \quad \lim_{N\to\infty} \frac{N_i}{N} = P_x(\alpha_i) \qquad \ldots (4)$$

Para el caso continuo los posibles valores se dividen en rangos, digamos a α_i como $\alpha_i+\Delta\alpha$, a α_2 como $\alpha_2+\Delta\alpha$, ..., a α_m como $\alpha_m+\Delta\alpha$. Si los diferentes rangos ocurren N_1, N_2, ..., N_m veces respectivamente, en N ensayos del experimento, entonces si $\Delta\alpha$ se encuentra dentro del rango de $\alpha_i < \alpha < \alpha_i+\Delta\alpha$, se puede decir que el valor α_i se puede obtener N_i veces. El valor promedio que se obtiene es aproximadamente

$$\frac{1}{N}\left(\sum_i \alpha_i N_i\right) \qquad \ldots (5)$$

que se puede expresar como

$$\bar{X} = \sum_{\forall i} \alpha_i P(rango\ \alpha_i < x < \alpha_i + \Delta_\alpha)$$

$$\bar{X} = \sum_{\forall i} \alpha_i f_x(\alpha_i)\, \Delta_\alpha$$

$$\bar{X} = \int_{-\infty}^{\infty} \alpha f_x(\alpha)\, d\alpha \qquad \ldots (6)$$

como se definió en la ecuación 2.

Antes de explicar lo que el valor promedio significa en términos de aplicación, es necesario definir otros conceptos. El valor medio de cualquier función de variable aleatoria g(X), denotado como $\overline{g(X)}$ o E[g(X)], se define como

$$\overline{g(x)} \quad o \quad E[g(x)] \triangleq \sum_{\forall I} g(\alpha_I) P_x(\alpha_I) \qquad \ldots (7)$$

para una variable aleatoria discreta y

$$\overline{g(x)} \quad o \quad E[g(x)] \triangleq \int_{-\infty}^{\infty} g(\alpha) f_x(\alpha) d\alpha \qquad \ldots (8)$$

para una variable aleatoria continua.

Algunos casos especiales de las ecuaciones 7 y 8 son de mucha utilidad. Haciendo una descripción para variables aleatorias continuas tenemos primero,

$$E(X^2) \quad o \quad \overline{X^2} \triangleq \int_{-\infty}^{\infty} \alpha^2 f_x(\alpha) d\alpha \qquad \ldots (9)$$

donde $\overline{X}^2$ es el valor medio cuadrático. Segundo,

$$E[(X-\overline{X})^2] \quad o \quad \sigma_x^2 \triangleq \int_{-\infty}^{\infty} (\alpha-\overline{X})^2 f_x(\alpha) d\alpha \qquad \ldots (10)$$

donde σ_x^2 es llamada la **variancia**. Tercero,

$$\sigma_x \triangleq \sqrt{\int_{-\infty}^{\infty} (\alpha - \overline{X})^2 f_x(\alpha)\, d\alpha} \qquad \ldots (11)$$

donde σ_x se llama **desviación estándar** de $\overline{X}$. Cuarto,

$$E(X^n) \quad o \quad \overline{X^n} \triangleq \int_{-\infty}^{\infty} \alpha^n f_x(\alpha)\, d\alpha \qquad \ldots (12)$$

donde $\overline{X}^n$ es llamado el **momento n-ésimo**. Y finalmente,

$$E[(X-\overline{X})^n] \quad o \quad \overline{(X-\overline{X})^n} \triangleq \int_{-\infty}^{\infty} (\alpha - \overline{X})^n f_x(\alpha)\, d\alpha \qquad \ldots (13)$$

donde $\overline{(X-\overline{X})^n}$ es llamado el **momento n-ésimo central**.

En términos probabilísticos el valor medio o promedio $\overline{X}$ de una función localiza el centro de gravedad del área bajo la curva de densidad de probabilidad de una variable aleatoria X. La variancia σ_x^2 de una variable aleatoria X en cierto sentido es una medida de la "aleatoriedad" de la variable, es decir, indica los valores de tendencia central respecto al promedio.

Ahora bien, desde el punto de vista del procesamiento de señales el promedio se considera en algunos casos como un filtro pasa bajas. Si yo adquiero un número N de barridos y obtengo el promedio de éstos lo que obtengo es la señal filtrada, se eliminan

las altas frecuencias que generalmente, en este contexto es ruido. Por otra parte sabemos que la relación señal-ruido está dada por:

$$SNR = \frac{Valor\ Medio\ de\ la\ Señal}{Valor\ Medio\ del\ Ruido}$$

En base a lo anterior se define que:

$$SNR = \frac{\overline{X}}{\sigma_x^2}$$

Integral. La integral representa el área bajo la curva de una función dada, es decir, nos da como información la cantidad de energía contenida en dicha función.

El método de integración que se utilizó en la implementación de la integral de los segmentos de señales adquiridos fue el conocido como **sumas de Riemann**, llamado así en honor del matemático Georg Friedrich Bernhard Riemann (1826-1866).

Sea f una función continua definida en el intervalo cerrado $[a,b]$. Dividimos este intervalo en n subintervalos escogidos *cualesquiera* $n-1$ puntos intermedios entre a y b. Sean $x_0 = a$ y $x_n = b$, y sean $x_1, x_2, \ldots, x_{n-1}$ los puntos intermedios de manera que

$$x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n$$

Los puntos $x_0, x_1, x_2, \ldots, x_{n-1}, x_n$ no son necesariamente equidistantes. Sea $\Delta_1 x$ la longitud del primer subintervalo tal que $\Delta_1 x = x_1 - x_0$; sea $\Delta_2 x$ la longitud del segundo subintervalo tal que

$\Delta_2 x = x_2 - x_1$; y así sucesivamente, de tal manera que la longitud del *i-ésimo* subintervalo sea $\Delta_i x$, y

$$\Delta_i x = x_i - x_{i-1}$$

Al conjunto de todos esos subintervalos del intervalo $[a,b]$ se le llama **partición** del intervalo $[a,b]$. Sea Δ una partición; la longitud del subintervalo más largo de la partición Δ, se llama **norma** de la partición y se denota por $\|\Delta\|$.

Escojamos un punto en cada subintervalo de la partición Δ: sea ξ_1 el punto escogido en $[x_0,x_1]$ tal que $x_0 \leq \xi_1 \leq x_1$. Sea ξ_2 el punto escogido en $[x_1,x_2]$ tal que $x_1 \leq \xi_2 \leq x_2$, y así sucesivamente, de manera que ξ_i sea el punto seleccionado en $[x_{i-1},x_i]$, y $x_{i-1} \leq \xi_i \leq x_i$. Formando la siguiente suma tenemos:

$$f(\xi_1)\Delta_1 x + f(\xi_2)\Delta_2 x + \ldots + f(\xi_i)\Delta_i x + \ldots + f(\xi_n)\Delta_n x$$

o bien

$$\sum_{i=1}^{n} f(\xi_i)\Delta_i x \qquad \ldots (14)$$

que es la expresión de la **suma de Riemann.**

En el caso particular de esta tesis, se aplica la integral a un segmento de señal de 256 puntos, por lo que se realiza la suma acumulada de los valores absolutos de cada uno de estos 256 puntos, considerando $\Delta x = 1$, de manera que se obtiene la integral de la función rectificada.

Función de autocorrelación y correlación cruzada.[24] La función

de correlación se considera como una prueba estadística de segundo orden y describe, en general, la dependencia de los valores de los datos en un tiempo con respecto a los valores de los datos en otro tiempo. La correlación se aplica generalmente al análisis de datos aleatorios, y puede ser utilizado para detectar señales periódicas inmersas en ruido. También la correlación provee de una medida de similaridad entre dos formas de ondas. Se utilizan dos funciones de correlación: la **autocorrelación** y la **correlación cruzada.** La autocorrelación mide la similaridad de una señal con una versión retardada en el tiempo de ella misma, mientras que la correlación cruzada mide el grado de similaridad de una forma de onda (la fuente, la entrada, el estímulo) con una segunda forma de onda (la salida, la respuesta).

La definición matemática de la función de autocorrelación está dada por la siguiente ecuación:

$$R_{xx}(C) = \lim_{T\to\infty} \frac{1}{T}\int_0^T x(t)x(t+C)\,dt \qquad \ldots(15)$$

La figura 22 muestra una pequeña parte de una forma de onda aleatoria. La ecuación 15 toma el valor de $x(t)$ en el tiempo t y el valor $x(t + C)$ en el tiempo $(t + C)$ y realiza el producto de ambos. Esta operación se repite para todo valor de t dentro del intervalo $0 < t < T$. La integral representa la sumatoria de todos los productos, y la operación 1/T presenta el promedio sobre la observación en el tiempo T. El resultado del producto promedio se aproxima a la función de autocorrelación exacta tanto como T tienda

a infinito.

La definición matemática de la función de correlación cruzada está dada por la ecuación:

$$R_{xy}(C) = \lim_{T\to\infty} \frac{1}{T}\int_0^T x(t)\,y(t+C)\,dt \qquad \ldots(16)$$

La única diferencia entre las ecuaciones 15 y 16 es que en el cálculo de la correlación cruzada, la señal $x(t)$ se multiplica por una versión retardada de la señal $y(t)$, en lugar de por una versión retardada de sí misma.

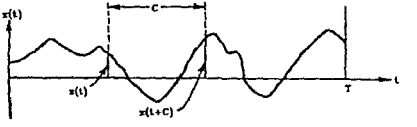


Figura 22. Forma de onda aleatoria. Para formar la función de autocorrelación para el intervalo C, debe realizarse el producto $x(t)\ x(t + C)$ para cada valor de t. [24]

Propiedades y Aplicaciones de la Función de Correlación. La figura 23 muestra una gráfica típica de autocorrelación Rxx(C) contra intervalos de desplazamiento de tiempo C para cuatro formas de onda, onda seno, onda seno más ruido aleatorio, ruido aleatorio de banda limitada, y ruido aleatorio de banda ancha.

La función de autocorrelación tiene las siguientes propiedades:

1. La autocorrelación es una función par con su máximo valor en C = 0. Como las funciones pares son simétricas con respecto al valor C = 0, es suficiente calcular la función de correlación solamente para los valores positivos de C.

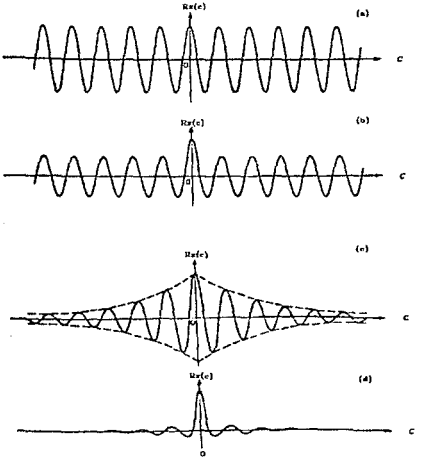


Figura 23. Función de autocorrelación para algunas señales típicas. Las señales son: (a)Onda seno; (b) Onda seno más ruido aleatorio; (c) Ruido aleatorio de banda limitada; (d) Ruido aleatorio de banda ancha.[24]

2. El valor máximo Rxx(0) de la función de autocorrelación es igual al valor cuadrático medio de la función de tiempo. Para propósitos de desplegado es conveniente normalizar el valor máximo

a 1, desplegando la función normalizada:

$$\frac{R_{xx}(C)}{R_{xx}(0)}$$

3. El valor de la función de correlación en C > ∞ es igual al cuadrado del valor promedio de la función de tiempo.

4. Si la función de tiempo contiene componentes periódicas, la función de autocorrelación contendrá componentes con un mismo período, figura 23a.

5. Si la función de tiempo contiene sólo componentes aleatorios, la función de autocorrelación se aproximará exponencialmente a cero mientras más se incremente C, figura 23d.

6. Si la función de tiempo está compuesta por dos o más componentes, la función de correlación será la suma de las funciones de correlación de cada una de las componentes individuales. La figura 23b presenta la función de correlación de una onda seno más ruido aleatorio; ésta se obtiene de la suma de las funciones desplegadas en las figuras 23a y d.

7. La función de correlación cruzada cumple también con las propiedades de la 3 a la 6. Sin embargo, la función de correlación cruzada no es necesariamente una función par, el valor máximo de Rxx(C) ocurrirá para aquel valor del intervalo de corrimiento C, para el cual las dos señales *x* y *y*, sean más parecidas.

Las aplicaciones típicas de la correlación incluyen:

1. La determinación de la ruta de transmisión y el retardo de propagación de las ondas eléctricas, mecánicas, acústicas, o

sísmicas.

2. Detección de señales muy débiles inmersas en ruido.

3. La indicación de epilepsia a través de la comparación de los electroencefalogramas de los dos hemisferios del cerebro.

4. La medición de la función impulso-respuesta de sistemas complejos en presencia de ruido.

5. Estudio de la enfermedad de Parkinson y análisis de la frecuencia de temblor.

6. Investigaciones en comunicaciones y habla.

Cuando se analizan procesos continuos el análisis que se hace de correlación se basa en la comparación de formas de onda, correlación de amplitudes, como se verá en la sección III.4 donde se describirán los algoritmos.

En el caso de procesos puntuales, como pueden ser trenes de espigas, lo que interesa conocer no es la forma, ya que en un proceso puntual sólo existe la presencia o no de un evento, sino los intervalos de duración entre espiga y espiga; por lo que se hace un análisis de correlación de intervalos.

DOMINIO DE LA FRECUENCIA.

Transformada de Fourier.[3] Las señales están definidas generalmente en el dominio del tiempo. Algunas veces se requiere representarlas en el dominio de la frecuencia, en donde se obtiene la distribución de amplitudes y fases respecto a la frecuencia. La representación en el dominio de la frecuencia es muy ventajosa en

muchos aspectos, como en el filtrado. Estas dos representaciones se relacionan mediante la *Transformada de Fourier (TF)* y la *Transformada Inversa de Fourier (TIF)*. Considérese la señal x(t) dada en el dominio del tiempo, su representación en frecuencia X(ω), se obtiene con la integral de Fourier:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t} dt \qquad \ldots(17)$$

en su forma exponencial, o bien:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, (\cos\omega t + j \sin\omega t)\, dt \qquad \ldots(18)$$

en su forma trigonométrica.

Este par de ecuaciones se pueden escribir simbólicamente como $X(\omega) = F\{x(t)\}$.

La transformación inversa, del dominio de la frecuencia al dominio del tiempo, está dada por la transformada inversa de Fourier:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t} d\omega \qquad \ldots(19)$$

Esta última ecuación se escribe simbólicamente como $x(t) = F^{-1}\{X(\omega)\}$.

Las señales en el dominio del tiempo con las que se trabaja son reales. En el dominio de la frecuencia, sin embargo, la

representación $X(\omega)$ es, en general, un valor complejo de ω. Se puede escribir $X(\omega)$ como:

$$X(\omega) = |X(\omega)|\, e^{j\theta(\omega)} \qquad \ldots(20)$$

donde $|X(\omega)|$ es la amplitud y $\theta(\omega)$ es la fase de la representación en el dominio de la frecuencia.

No todas las señales reales pueden ser transformadas al dominio de la frecuencia utilizando la ecuación 17. Las condiciones suficientes de existencia de $X(\omega)$ están dadas por las condiciones de Dirichlet:

1. $\int_{-\infty}^{\infty} |x(t)|\,dt < \infty$, $x(t)$ es absolutamente integrable.
2. $x(t)$ tiene un número finito de discontinuidades y un número finito de puntos extremos en cualquier intervalo finito.

Existen funciones muy útiles como la función impulso (delta), la función escalón, o las funciones seno y coseno pares, las cuales no cumplen con las condiciones de Dirichlet. Estas funciones no tienen transformada de Fourier, sin embargo, tienen la transformada en el límite.

Algunas propiedades de la Transformada de Fourier.

1. Linearidad. La TF es un operador lineal. Si $X_1(\omega) = F\{x_1(t)\}$ y $X_2(\omega) = F\{x_2(t)\}$, entonces para cualquier constante arbitraria a_1 y a_2, se tiene:

$$F\{a_1 x_1(t) + a_2 x_2(t)\} = a_1 X_1(\omega) + a_2 X_2(\omega) \qquad \ldots(21)$$

2. Teorema de Convolución. El teorema de convolución es una herramienta importante en el análisis en frecuencia. La integral de convolución de dos funciones x1(t) y x2(t), para funciones estacionarias, está definida por:

$$x(t) = \int_{-\infty}^{\infty} x_1(\tau)\, x_2(t-\tau)\, d\tau \qquad \ldots(22)$$

La integral de convolución se escribe simbólicamente como:

$$x(t) = x_1(t) \otimes x_2(t) \qquad \ldots(23)$$

La importancia de la ecuación 22 en el análisis de sistemas radica en el hecho de que la salida de un sistema lineal está dado por la convolución de la entrada con la respuesta impulso del sistema. Puede mostrarse fácilmente que la FT de x(t) es:

$$F\{x(t)\} = F\{x_1(t) \otimes x_2(t)\} = X_1(\omega) \cdot X_2(\omega) \qquad \ldots(24)$$

donde $X_i(\omega) = F\{x_i(t)\}$. De aquí, la operación de convolución en el dominio del tiempo, que es una operación relativamente complicada, se convierte en una simple operación de multiplicación en el dominio de la frecuencia.

Considérese ahora la convolución de dos funciones, $X_1(\omega)$ y $X_2(\omega)$, en el dominio de la frecuencia:

$$X(\omega) = \int_{-\infty}^{\infty} X_1(u)\, X_2(\omega - u)\, du = X_1(\omega) \otimes X_2(\omega) \qquad \ldots(25)$$

Puede verse fácilmente que la TIF de X(ω) es:

$$x(t) = F^{-1}\{X(\omega)\} = 2\Pi \cdot x1(t) \cdot x2(t)$$

De aquí, la convolución de dos funciones en el dominio de la frecuencia está dado por 2Π veces la multiplicación de dos funciones en el dominio del tiempo.

3. *Teorema de Parseval*. Considere la energía, E, de una función del tiempo x(t):

$$E = \int_{-\infty}^{\infty} x^2(t)\, dt \qquad \ldots(26)$$

Reemplazando x(t) en la ecuación 26 por $F^{-1}\{X(\omega)\}$ tenemos:

$$E = \int_{-\infty}^{\infty} x(t) \left(\frac{1}{2\Pi} \int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t} d\omega \right) dt \qquad \ldots(27)$$

Intercambiando el orden de la integración obtenemos:

$$E = \int_{-\infty}^{\infty} x^2(t)\, dt = \frac{1}{2\Pi} \int_{-\infty}^{\infty} X(\omega)\, X(-\omega)\, d\omega \qquad \ldots(28)$$

Para x(t) real, se tiene $X(\omega) = X^*(-\omega)$ (donde el símbolo * denota el complejo conjugado), de aquí:

$$E = \int_{-\infty}^{\infty} x^2(t)\,dt = \frac{1}{2\Pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \qquad \ldots(29)$$

La ecuación 29 se conoce como el *teorema de Parseval*, el cual establece que la energía total de una señal puede ser calculada por la integral en el dominio del tiempo y en el dominio de la frecuencia. Nótese que si se requiere la energía en una banda de frecuencia de $\omega 1 \leq \omega \leq \omega 2$, se tiene que integrar sobre esa banda y sobre la banda de frecuencias negativas $-\omega 2 \leq \omega \leq \omega 1$. Como $|X(t)|^2$ es una función par, en ω, tenemos:

$$E_{\omega_1-\omega_2} = 2\cdot\frac{1}{2\Pi}\int_{\omega_1}^{\omega_2} |X(\omega)|^2 d\omega \qquad \ldots(30)$$

Se puede definir la *densidad espectral de energía*, $\hat{S}(\omega)$, por:

$$\hat{S}(\omega) = \frac{1}{\Pi}|X(\omega)|^2 \qquad \ldots(31)$$

La ecuación 30 puede escribirse:

$$E_{\omega_1-\omega_2} = \int_{\omega_1}^{\omega_2} \hat{S}(\omega)\,d\omega \qquad \ldots(32)$$

y la energía total es:

$$E = \int_0^{\infty} \hat{S}(\omega)\,d\omega \qquad \ldots(33)$$

4. Transformada de Fourier de Señales Periódicas. Las señales periódicas no satisfacen las condiciones de Dirichlet por lo tanto no tienen TF. Sin embargo, si nosotros asumimos que la señal existe sólo en un intervalo de tiempo finito (-τ/2, τ/2), entonces la TF sí existe. Se puede encontrar la TF en el límite cuando τ tiende a infinito.

Se puede calcular, sin embargo, la TF de una señal periódica de otra forma. Considérese la serie compleja de Fourier de la señal:

$$x(t) = \sum_{n=-\infty}^{\infty} a_n e^{jn\omega_0 t} \qquad \ldots (34)$$

donde T = 2Π/ω0 es el período de la señal, y an son los coeficientes de la expansión.

Aplicando la TF a la ecuación 34 se tiene:

$$X(\omega) = F\left\{\sum_{n=-\infty}^{\infty} a_n e^{j\omega\omega_0 t}\right\} = 2\Pi \sum_{n=-\infty}^{\infty} a_n \delta(\omega - n\omega_0) \qquad \ldots (35)$$

De aquí, que la TF de una señal periódica es un tren de funciones impulso localizados en los armónicos de la frecuencia fundamental, ω0, cada una multiplicada por 2Π veces el correspondiente coeficiente de la serie de Fourier.

Transformada Discreta y Transformada Rápida de Fourier

(TDF,TRF). Considérese una señal de banda limitada en tiempo x(t), que ha sido muestreada con un intervalo de muestreo T_s. Asumimos que la frecuencia de muestreo $\omega_s = 2\Pi/T_s$ obedece el teorema de Nyquist de manera que la secuencia muestreada contiene toda la información presente en la señal.

Denotamos una secuencia muestreada finita por $\{x(nT_s)\}$, n = 0,1, ... ,N-1.

Se define la *transformada discreta de Fourier (TDF)*, como un operador lineal sobre la secuencia {x(nT)} tal que:

$$X\left(k\frac{\omega_s}{N}\right) = \sum_{n=0}^{N-1} x(nT_s)\, e^{-j2\Pi k\frac{n}{N}}; \quad k=0,1,\cdots,N-1 \qquad \ldots(36)$$

La secuencia:

$$\left\{X\left(k\frac{\omega_s}{N}\right)\right\}$$

es, en general, una secuencia compleja. La transformación que mapea esta secuencia compleja de regreso a la secuencia $\{x(nT_s)\}$ se llama *transformada discreta inversa de Fourier (TDIF)*:

$$x(nT_s) = \frac{1}{N}\sum_{k=0}^{N-1} X\left(k\frac{\omega_s}{N}\right)e^{2\Pi n\frac{k}{N}}; \qquad n=0,1,\cdots,N-1 \qquad \ldots(37)$$

Nótese que T_s y ω_s/N son constantes, por eso se denotan:

$$x_n = x(nT_s) \qquad y \qquad X_k = X\left(k\frac{\omega_s}{N}\right)$$

Las transformaciones de las ecuaciones 36 y 37 se pueden escribir simbólicamente como:

Xk = TDF{xn}; xn = TDIF{Xk} ...(38)

Consideremos ahora una secuencia muestreada {x(nTs)} como una función del tiempo generada por la multiplicación de la señal x(t) con el operador muestra ideal, $\delta_T(t)$. Denotamos la señal muestreada como x*(t) y obtenemos:

$$x^*(t) = \sum_{n=0}^{N-1} x(t)\,\delta(t-nT_s) \qquad ...(39)$$

La TF de la señal x*(t), llamada X*(ω), se calcula fácilmente:

$$F[\delta(t-nT_s)] = e^{-j\omega nT_s} \qquad ...(40)$$

entonces, de la ecuación 39 tenemos:

$$X^*(\omega) = F[x^*(t)] = \sum_{n=0}^{N-1} x(nT_s)\,e^{-j\omega nT_s} \qquad ...(41)$$

La ecuación 41 describe la TF de una señal muestreada. Esta TF, X*(ω), es una función continua de ω. Muestreando, en el dominio de la frecuencia, esta función continua en intervalos de frecuencias de ωs/N = 2Π/NTs. Obtenemos una secuencia de las muestras de TF:

$$\left\{X*\left(k\frac{\omega_s}{n}\right)\right\}, \quad k=\cdots,-1,0,1,\cdots$$

$$X*\left(k\frac{\omega_s}{N}\right) = \sum_{n=0}^{N-1} x(nT_s)\, e^{-j2\Pi k\frac{n}{N}} \; ;k=\cdots,-2,-1,0,1,2,\cdots \quad \ldots(42)$$

Nótese que el conjunto de N miembros k = 0, 1,..., N-1 de la secuencia infinita (ecuación 42) es igual a la TDF de la ecuación 36.

Como conclusión, se puede afirmar que la TDF (ecuación 36) es una secuencia de N muestras distribuidas no uniformes de TF de la señal x(t). La TDF tiene algunas propiedades que son en naturaleza iguales a las de la TF.

La TDF es una herramienta importante en el procesamiento de señales discretas por las mismas razones que la TF es importante para el procesamiento de señales continuas. El calculo directo de la TDF requiere aproximadamente N^2 operaciones de multiplicación y adición complejas. En 1965, Cooley y Tukey, presentaron un método eficiente para calcular la TDF. Su método, conocido como *transformada rápida de Fourier (TRF)* (del inglés FFT), requiere solamente N $\log_2$ N operaciones (donde N es una potencia de 2). Para N = 1024, el número de operaciones requeridas para el cálculo con TRF es de diez veces menor que el número requerido en el cálculo directo.

El algoritmo de Cooley y Tukey para el cálculo de la TRF será descrito en la sección III.4 donde se describirá la función que realiza la TRF en el sistema GRETA.

III.3. ARQUITECTURA BASICA.

El sistema GRETA está constituido por cuatro unidades como se muestra en el diagrama de bloques de la figura 24: la unidad de primitivos gráficos PRIMIFUN, la unidad de gráficos avanzados GRAFFUN, la unidad de adquisición y uso del convertidor PC-LabCard-812 ADFUN, y la unidad de osciloscopio digital OSCILO.

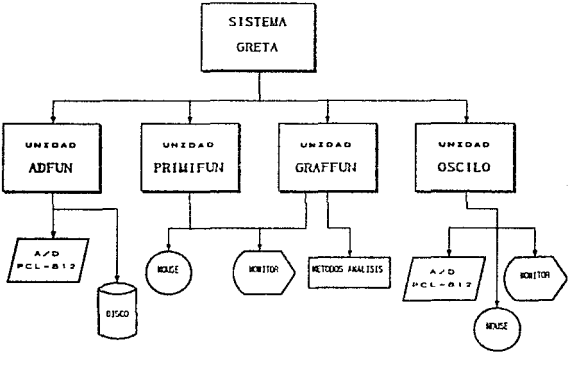


Figura 24. Diagrama de bloques de la arquitectura básica del sistema GRETA.

La unidad de primitivos gráficos **PRIMIFUN** cuenta con funciones que se ocupan del entorno gráfico general, es decir, tiene funciones para realizar los menús de opciones, el uso del mouse, ventanas de despliegue, lectura de parámetros, etc. El siguiente es un listado del encabezado de las funciones que la constituyen.

```
/* Definiciones de Primitivos Gráficos  PRIMIFUN.H*/

/* Definición de funciones */

extern float readfloat(char *texto);
extern char far * readtext(char far *texto);
extern int far * salva_vent(int x1,int y1,int x2,int y2);
extern void restaura_vent(int x,int y,int far *ptres);
extern void parametros_prompt(int fondo,int marco,int chr);
extern void parametros_menu(int x,int y,int tx,int ty,
             int fondo,int marco,int chr,int save,int far
             *ptmenu);
extern void parametros_opcion(int fondo,int marco,int chr,int
             distancia);
extern void desplaza(int x,int y,int sentido);
extern void menu_enter(char  *texto);
extern int menu(void);
extern void ventana(int x,int y,int tx,int ty,int fondo,int marco);
extern int  entra_ventana(int x1,int y1,int x2, int y2);
extern int  sal_ventana(int x1,int y1,int x2,int y2);
extern void detecta_boton(void);
extern void suelta_boton(void);
extern void Mshow(int showstat);
extern Mstatus Mpos();
extern Mstatus Mpressed(int button);
extern Mresult  *Mreset();
extern void plot(int x,int y,int color);
extern void presentacion(void);
extern void titulo(void);
extern void par_config(void);
```

La función **readfloat** lee un número real en modo gráfico, tiene como parámetro de entrada un apuntador a un carácter que es el que contiene la dirección donde inicia el letrero que aparece en la ventana de lectura, y regresa un número real que es el parámetro

leído. La función **readtext** de forma similar lee un texto, por ejemplo el nombre de un archivo, tiene el mismo parámetro de entrada que readfloat, y regresa un apuntador lejano que contiene la dirección donde se almacenan los caracteres que componen el nombre leído.

La función **salva_vent** guarda en memoria los pixeles contenidos en la ventana que se indica con los parámetros de entrada. La coordenada (x1,y1) corresponde a la esquina superior izquierda, y la coordenada (x2,y2) corresponde a la esquina inferior derecha de la ventana a salvar; la función regresa un apuntador lejano a un área de memoria que corresponde a la dirección donde se guarda la imagen. La función **restaura_vent** regresa al video la imagen que se guardó en memoria, por medio de la función salva_vent, tiene como parámetros de entrada la coordenada de la esquina superior izquierda (x,y), y el apuntador lejano a entero que contiene la dirección en memoria donde se encuentra la ventana almacenada.

La función **parametros_prompt** inicializa los parámetros de la ventana en donde se lee un número real o un nombre, que son el color del fondo de la ventana, color del marco y color de los caracteres desplegados. La función **parametros_menu** inicializa los parámetros de las ventanas de menús, que son el tamaño de la ventana de (x,y) a (tx,ty), color del fondo, color del marco, color de los caracteres desplegados, si se va a salvar la ventana en memoria (save=1) o no (save=0), y finalmente el apuntador lejano a un entero que contendrá la dirección en memoria donde se guardará la ventana, si es el caso. La función **parametros_opcion** de manera

análoga inicializa los parámetros de las opciones que contendrá el menú, que son el color del fondo de la ventana de la opción, color del marco, color de los caracteres desplegados, y la distancia en pixeles que habrá entre una opción y otra.

La función **desplaza** realiza un desplazamiento del desplegado de la primera opción del menú, en *x* y/o en *y* con respecto a la coordenada superior izquierda de la ventana del menú; y el sentido indica si las opciones del menú serán en forma horizontal (sentido =0) o en forma vertical (sentido=1). Con la función **menu_enter** se inicializa el contenido de las opciones del menú, el parámetro es un apuntador a un carácter. La función **menu** es la función principal que realiza la llamada a las demás funciones antes mencionadas (y las funciones de mouse que se describirán a continuación), y genera los menús, esta función regresa un entero que corresponde al número de la opción seleccionada. La función **ventana** dibuja una ventana (bloque uniforme) de tamaño de (x,y) a (tx,ty), con color del fondo y color del marco.

La función **entra_ventana** detecta si la posición del mouse está dentro de la ventana definida por las coordenadas (x1,y1) y (x2,y2), la función regresa un valor entero. Si es igual a 1 el mouse está dentro de la ventana, si es distinto de 1 el mouse está fuera de ella. La función **sal_ventana** detecta que el mouse, una vez que está dentro de la ventana, salga de ella; la ventana se define nuevamente por las coordenadas (x1,y1) y (x2,y2) y regresa un valor entero que es igual a 1 si el mouse sale de la ventana y distinto de 1 si no ha salido de ella. Estas dos funciones tiene la

finalidad de hacer la selección de las opciones de un menú, entre otras.

La función **detecta_boton** detecta si ha sido presionado algún botón, cualquiera, del mouse. Con **suelta_boton** se detecta si el botón presionado ha sido soltado, esto es para que no se detecte el botón más de una vez si es que éste se mantiene presionado durante un cierto período. La función **Mshow** enciende (TRUE) el cursor del mouse o lo apaga (FALSE). Mpos regresa la posición x,y del mouse en la pantalla del monitor, esté o no encendido su cursor. La función **Mpressed** pregunta por el status del botón indicado como parámetro de entrada "button", y regresa si ha sido presionado y en qué posición x,y se encuentra el mouse en ese momento, y cuántas veces ha sido presionado. La función **Mreset** es de tipo Mresult, esta función inicializa el mouse e indica si está presente o no en el sistema.

La función **plot** enciende un pixel del monitor en la posición x,y de un color definido, esta función está escrita en ensamblador y es más rápida que la función putpixel de Borland. La función **presentacion** despliega la pantalla de fondo del sistema GRETA; la función **titulo** despliega el nombre del sistema, y finalmente la función **par_config** despliega en la parte superior de la pantalla una ventana de condiciones actuales de los parámetros de configuración del sistema. Estos son: modo de disparo del convertidor, número de barridos, tiempo de barrido, modo de lectura, modo de escritura, nombre de los archivos de lectura y escritura.

La unidad de gráficos avanzados **GRAFFUN**, contiene las funciones de desplegado de series, de implementación de métodos de análisis como correlación, FFT, cursor de medición de voltajes, tiempos,integrales total y de segmento y escalamiento en tiempo y voltaje. A continuación se presenta el listado de los encabezados de las funciones.

```
/* Definición de Gráficos de Alto Nivel GRAFFUN.H*/

/* Definición de funciones */

extern void cursor(int far *dat,int x,int y,int num_canal);
extern void despliega16(int seleccion);
extern void despliega_1(int num_canal);
extern void correlación(int canal1,int canal2);
extern void canal_umbral(int canal);
extern void cuenta_intervalos(int canal,int num_intervalo);
extern void auto_cor_int(int canal,int num_intervalo,int
                         num_correlación);
extern void dvd(complex a,complex b,complex *r); /* aritmética
extern void mul(complex a,complex b,complex *r); de complejos*/
extern void add(complex a,complex b,complex *r);
extern void FFT(int m,int isign,int channel);     /* FFT */
extern void despliegaFFT1(int canal);
extern void cursor_fft(int canal);
```

La función **cursor** despliega una señal de un canal determinado en una ventana para su medición; los parámetros de entrada a la función son: la posición de la ventana en la pantalla expresada en pixeles con respecto a la esquina superior izquierda (x,y), la señal a desplegar que está contenida en un arreglo de datos enteros apuntado por *dat*, y el número de canal. Con esta función se puede hacer un escalamiento de la señal en tiempo y voltaje, se pueden hacer corrimientos de la señal a la izquierda o a la derecha del eje del tiempo, se puede hacer medición de tiempo y voltaje en un

punto determinado, y obtener la integral total o la integral en un segmento de la señal. Esta función se describirá con más detalle en la siguiente sección.

La función **despliega16** despliega un barrido en la pantalla, un barrido consta, como ya se mencionó, de la adquisición de 16 canales de 256 puntos cada uno, el barrido puede ser en tiempo o en en frecuencia; el parámetro selección se refiere a que aparezca un menú de selección de canal (seleccion=1) si se desea hacer análisis utilizando la función *cursor* para el caso de barridos en tiempo, o la función *cursor_fft* para el caso de barridos en frecuencia, o que sólo despliegue el barrido sin menu (seleccion=0). La función **despliega_1** despliega sólo un canal del barrido en tiempo utilizando la función cursor, el parámetro de entrada es el número de canal.

La función **correlación** implementa el método de correlación cruzada y autocorrelación de amplitudes, los parámetros son los números de canales a ser correlacionados si canal1 = canal2 se trata de autocorrelación de amplitud, si canal1 ≠ canal2 se trata de correlación cruzada de amplitud.

La función **canal_umbral** despliega en una ventana una señal de 512 puntos del canal seleccionado con el parámetro *canal*, en línea, es decir, no es una señal adquirida previamente, sino la que es leída en ese momento del canal del convertidor. Con esta función el usuario puede definir un umbral; en base a éste, con la función **cuenta_intervalos**, mide el tamaño de los *n* intervalos definidos por el parámetro *num_intervalos* y los guarda en un arreglo; de manera

que un proceso continuo se traduce en un proceso puntual. Una vez hecho lo anterior con la función **auto_cor_int** se aplica el algoritmo de autocorrelación de intervalos, esta función tiene como parámetros de entrada el número de canal, el número de intervalos, y el número de correlaciones, es decir, el tamaño de la ventana de correlación. El algoritmo se aplica sobre un arreglo global llamado *intervalo* que contiene las medidas de los *n* intervalos definidos.

Las funciones **dvd** (división), **mul** (multiplicación), y **add** (adición) son funciones de aritmética de complejos, las tres son operaciones binarias y tienen como parámetros los complejos *a* y *b* que son los operandos, y el resultado se guarda en la dirección en la que apunta r. Finalmente la función **FFT** implementa el algoritmo de Cooley y Tukey para el cálculo de la transformada de Fourier.

La función **despliegaFFTi** despliega la transformada de un canal indicado por el parámetro de entrada *canal*; y la función **cursor_fft** despliega en una ventana la FFT del canal seleccionado por el parámetro *canal*, y permite hacer mediciones de frecuencia y potencia del punto deseado.

La unidad de adquisición y uso del convertidor PC-LabCard-812 **ADFUN**, contiene las funciones que se encargan de la programación de la tableta convertidora, inicialización de conversión, modo de disparo, tiempo de barrido, modo de almacenaje, modo de lectura, etc. A continuación se presenta el listado de los encabezados de sus funciones.

```
/* Encabezado de funciones del convertidor PCLAB.  ADFUN.H */

/* Definición de funciones */

extern void ad_init(void);
extern void ad_finit(void);
extern void prom_init(void);
extern void prom_finit(void);
extern void fft_init(void);
extern void fft_finit(void);
extern void set_ad_rate(float sw_time);
extern int convierte(int canal);
extern void adq_16_can(void);
extern void set_trigger_mode(int mode);
extern void set_disk_mode_read(int mode);
extern void set_disk_mode_write(int mode);
extern int read_disk(char const *namefile);
extern int write_disk(char const  *namefile);
extern void archivo_sec(char * nombre);
```

Las funciones **ad_init** y **ad_finit** inicializan y liberan, respectivamente, un área de memoria de 4096 enteros apuntada por *adbuff* (apuntador lejano a entero, 32 bits), en la que se almacenan las adquisiciones de un barrido (16 canales por 256 puntos cada uno). Las funciones **prom_init** y **prom_finit** inicializan y liberan, respectivamente, un área de memoria de 4096 enteros apuntada por *prombuff*, en la que se realiza y almacena el resultado de los *n* barridos promediados. Y las funciones **init_fft** y **finit_fft** inicializan y liberan, respectivamente, un área de memoria de 4096 reales (float) apuntada por *adfft*, en la que se realizan y almacenan los resultados del calculo de la FFT y los promedios en frecuencia. Esto es lo que se conoce como manejo de memoria dinámica.

La función **set_ad_rate** programa a la tarjeta convertidora con la frecuencia de muestreo deseada, el parámetro de entrada es el float *sw_time* que se refiere al tiempo que debe emplear en

convertir 256 puntos; por lo que la frecuencia de muestreo (*rate*) con la que esta función programa a la tarjeta será (256/sw_time)*16 canales, ya que la tarjeta trabaja con "multiplexaje". La función **convierte** lee de la tarjeta convertidora el dato convertido del canal seleccionado con el parámetro de entrada *canal* y regresa el dato convertido en una variable entera. La función **adq_16_can** es una función escrita en ensamblador que adquiere los 256 puntos de los 16 canales, es decir un barrido a la vez, según el tiempo de barrido, y el modo de disparo del convertidor; esta función guarda las conversiones en el área de memoria apuntada por *adbuff*.

La función **set_trigger_mode** inicializa el modo de disparo del convertidor: NO_TRIGGER=0 que indica que no hay espera para convertir, en cuanto se llama a la función se inicia la conversión; KB_TRIGGER=1 que indica que hay que esperar a que se presione una tecla para iniciar la conversión; y TTL_TRIGGER=2 que indica que hay que esperar un pulso externo TTL (sincronía) para iniciar la conversión.

Las funciones **set_disk_mode_read** y **set_disk_mode_write** inicializan el modo de lectura y escritura, respectivamente; los modos son: BIN=0 con el que se leen o se escriben en disco los datos en binario; y ASCII=1 con el que se leen o escriben en disco los datos en código ASCII.

Las funciones **read_disk** y **write_disk** leen o escriben un barrido en un archivo en disco, según el modo de lectura o escritura definido. El parámetro de entrada de estas dos funciones es un apuntador a un carácter constante que apunta al nombre del

archivo. Y finalmente, la función **archivo_sec** lleva la secuencia del nombre de la serie tanto para escritura como para lectura; es decir, un archivo corresponde a un barrido, una serie es una colección de *n* barridos, la función archivo_sec inserta automáticamente un número de secuencia al nombre de la serie, según el número de barridos, para cada archivo.

Por último la unidad de osciloscopio digital **OSCILO**, cuenta con las funciones necesarias para hacer las veces de un osciloscopio analógico desplegando ocho canales a la vez.

```
/* Definición de Modulo Osciloscopio Digital */

/* Definición de Funciones */

extern int os_convierte(int canal);
extern void os_set_sample(void);
extern void os_set_ad(int rate);
extern void os_pantalla(void);
extern void os_menu_principal(void);
extern void os_encabezado(void);
extern int  os_menu(int nopciones,int c_barra1,int c_letrero1,
                    int c_barra2,int c_letrero2,int marco);
extern void osciloscopio(void);
extern float os_readfloat(char *texto);
```

La función **os_convierte** es equivalente a la función convierte de la unidad ADFUN, las funciones **os_set_sample** y **os_set_ad** hacen las veces de la función set_ad_rate de la unidad ADFUN, con la diferencia de que el cálculo de la frecuencia de muestreo ahora no se hace para los 256 puntos por 16 canales, sino de los 640 puntos de la pantalla VGA por 8 canales que se despliegan a la vez (multiplexados).

ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

La función **os_pantalla** pone la cuadrícula en la pantalla (escalas del osciloscopio). Con la función **os_menu_principal** se cambian los parámetros de tiempo de barrido (segundos por división), factor de división de voltaje (volts por división), y conjunto de canales seleccionados (del 1 al 8 o del 9 al 16). La función **os_encabezado** actualiza el encabezado de los parámetros en uso. La función **os_menu** hace las veces de la función menu del la unidad PRIMIFUN. La función **os_osciloscopio** hace el desplegado de los ocho canales seleccionados, según parámetros. Y finalmente, la función **os_readfloat** hace las veces de la función readfloat de la unidad PRIMIFUN para la lectura de los parámetros.

III.4. DESCRIPCION DE LAS FUNCIONES PRINCIPALES.

En esta sección se hará una descripción de las funciones principales que contienen las unidades del sistema GRETA a través de sus diagramas de flujo. Los listados de estas funciones así como los encabezados completos de las unidades se encuentran en el apéndice B.

De la unidad ADFUN se describirán las funciones **set_ad_rate, convierte** y **adq_16_can.**

La función **set_ad_rate** inicializa la frecuencia de muestreo con la que la tarjeta convertidora PCL-812 va a convertir. El parámetro de entrada a esta función es el float *sw_time* que es el tiempo de barrido que el usuario configura antes de iniciar la adquisición, el tiempo de barrido se refiere al tiempo en que tarda la tarjeta en adquirir 256 puntos de un canal. La frecuencia de

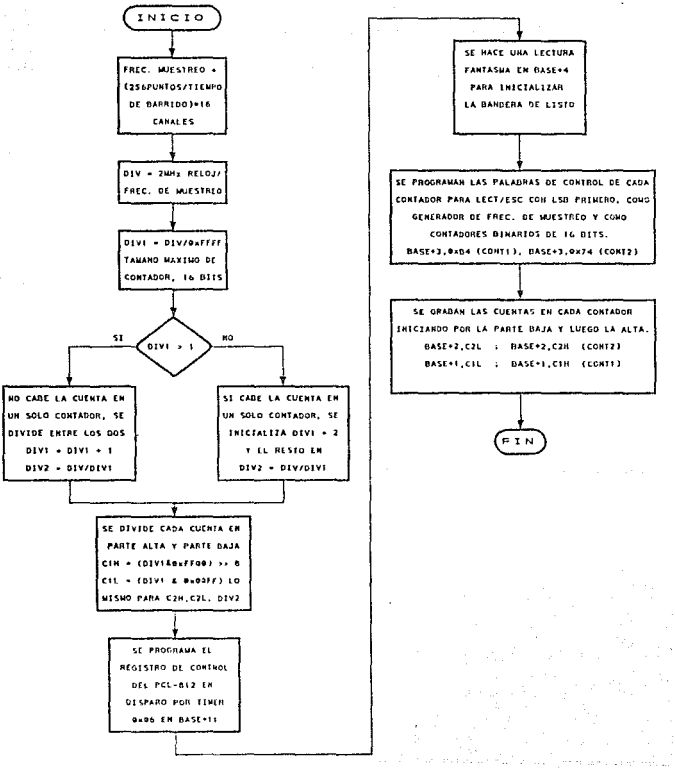
muestreo real de la tarjeta, por lo tanto, debe calcularse como rate=(256/sw_time)*16 canales, ya que la tarjeta multiplexa los 16 canales. El diagrama de flujo de la función se presenta en la página 81.

Como se describe en el apéndice A [21], el PCL-812 utiliza un dispositivo INTEL 8253-5 temporizador/contador programable, el cual contiene 3 contadores de 16 bits; dos de ellos, el contador 1 y 2 están conectados en cascada y tienen como base de tiempo una entrada de 2MHz, por lo que la velocidad de conversión se calcula, como se muestra en el diagrama (div = 2MHz/rate). A continuación se debe hacer la división de las cuentas que va a efectuar cada contador según el valor de div. Una vez divididas las cuentas en C1 y C2, estas deben dividirse a su vez en parte baja y parte alta, ya que los registros de los contadores que deben programarse son de 8 bits (C1H, C1L, C2H, C2L).

Se programa el registro de control (BASE+11) con la condición interna de disparo por timer. Se hace una lectura fantasma (BASE+4) con el objeto de inicializar la bandera DRDY de dato listo. Se programan las palabras de control de cada contador (BASE+3) para configurarlos: número de contador; operación de lectura/escritura, primero el LSB y luego el MSB; el modo de operación como generador de velocidad y como contadores binarios de 16 bits.

Una vez configurados se cargan las cuentas en los respectivos registros: BASE+2,C2L ; BASE+2,C2H (para el contador 2); y BASE+1,C1L ; BASE+1,C1H (para el contador 1).

FUNCION SET_AD_RATE



La función **convierte** tiene como parámetro de entrada el número de canal a convertir, y regresa el dato convertido en una variable entera. El diagrama de flujo se muestra en la página 83.

Primero debe programarse el canal de conversión en BASE+10,canal; en esta dirección se realiza la selección del canal deseado. Luego debe preguntarse por el bit 5 de BASE+5 que es el bit DRDY de dato listo, para asegurarse que ya está lista la conversión. Seguidamente se lee el dato convertido en dos partes, en BASE+5 se lee la parte alta, y en BASE+4 se lee la parte baja. Ambas se guardan en una variable entera *dato*. Como la tarjeta convertidora es de 12 bits se tiene un rango de valores de 0 a 4096, por conveniencia en el despliegue de la señal se hace un ajuste restándole la basal a dato: *dato=dato-0x800*, de manera que quede un rango de -2048 a 2047. Finalmente la función regresa el dato ajustado.

La función adq_16_can esta desarrollada en ensamblador 80x86 y tiene como objetivo realizar la adquisición de un barrido (16 canales de 256 puntos c/u) y guardarlo en un área de memoria apuntada por *adbuff*. Esta función no tiene parámetros de entrada ya que adbuff, así como los modos de disparo que utiliza están definidos como variables globales. El diagrama de flujo se encuentra en la página 85.

El sistema GRETA tiene la opción de convertir bajo tres modos de disparo, según lo configure el usuario: el primero es NO_TRIGGER con el cual la función adq_16_can adquiere inmediatamente despues de que es llamada, la segunda es KB_TRIGGER para la cual la función

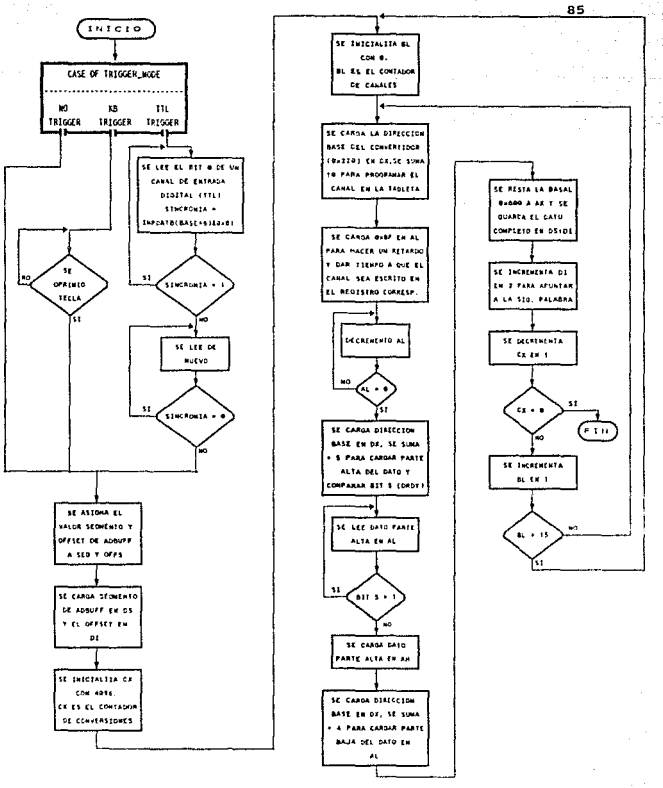
FUNCION CONVIERTE

debe esperar a que se oprima una tecla cualquiera, y la tercera TTL_TRIGGER la cual lee el bit 0 de uno de los canales de entrada digital TTL, en este caso BASE+6, y la conversión inicia cuando se detecta una transición de 0 a 1.

Una vez que se ha cumplido con el modo de disparo, se asigna el valor del segmento y del offset de adbuff a variables enteras sin signo, *seg* y *offs*, respectivamente; esto se hace con las macros FP_SEG y FP_OFF de Borland. Se cargan estos valores a los registros DS (data segment) y DI (data index), respectivamente. Se inicializa el contador de conversiones en CX con 4096; y se inicializa el contador de canales en BL con 0.

Se carga en DX la dirección BASE (0x220) + 10 para programar el canal de conversión, a través del acumulador: OUT DX,AL. Se realiza un retardo de 0x0F para dar tiempo de que el canal sea seleccionado por el multiplexor. Nuevamente se carga la dirección BASE + 5 en DX, se lee la parte alta del dato: IN AL,DX y se prueba el bit 5 de DRDY (dato listo), cuando el bit 5 = 0 se carga la parte alta del dato que ya se leyó en AL a AH. Luego se carga la dirección BASE + 4 en DX para leer la parte baja del dato en: IN AL,DX. Por lo que finalmente tenemos el dato completo en AX al cual se le resta la basal 0x800 y se guarda el dato completo en la dirección de adbuff (guardada en DS:DI): MOV DS:[DI],AX. Se incrementa DI en dos para apuntar a la siguiente palabra (palabras de 16 bits). Se decrementa al contador de conversiones (CX), si éste es igual a 0 se termina la función; si no, incrementa el contador de canales (BL), si éste es menor que 15 brinca a la

FUNCION ADQ_16_CAN



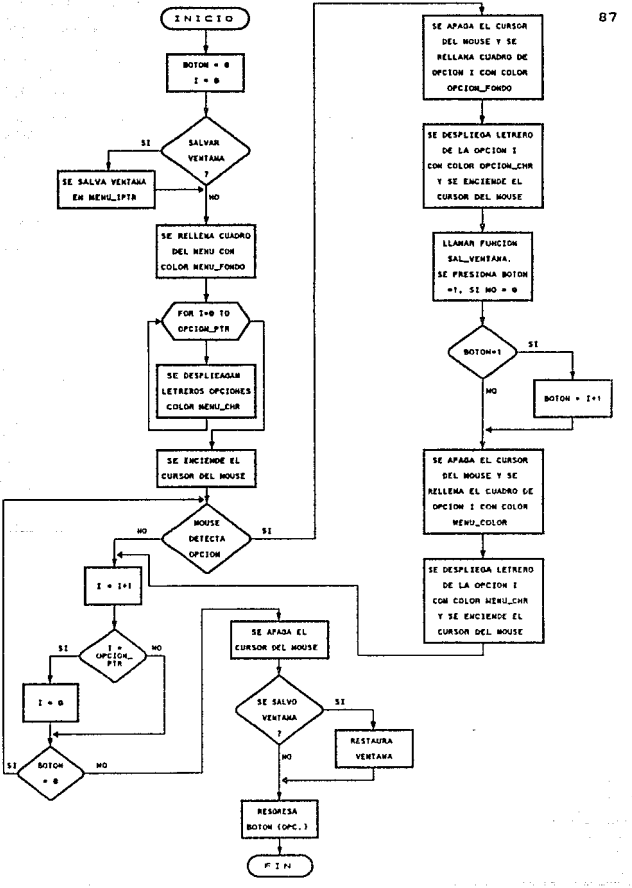
etiqueta *loop16* y repite la conversión del dato 1 de cada canal, guardando los datos secuencialmente en DS:[DI]; cuando BL llega a 15 brinca a la etiqueta *loop256* en donde inicializa el contador de canales (BL) en 0 e inicia la conversión del dato 2 de cada canal; etc., hasta que CX=0 termina la función adquiriendo al final del dato 256 de cada canal.

De la unidad PRIMIFUN se describirá la función **menu**.

La función **menu** no tiene parámetros de entrada, utiliza las variables globales inicializadas en las funciones *parametros_menu*, *parametros_opcion*, y *menu_enter*. Esta función tiene como objetivo el dibujar la ventana de menu con sus opciones y regresar un dato entero (en la variable *boton*) que es el número de la opción seleccionada con el mouse. El diagrama de flujo que la describe se presenta en la página 87.

Primero se inicializan las variables *boton=0* e *i=0*, se pregunta si se debe salvar la ventana de menu, si es el caso se salva en la dirección apuntada por *menu_iptr*; luego se rellena el cuadro del menu con color *menu_fondo*, y se despliegan las *n* opciones con color *menu_chr*. Se enciende el cursor del mouse, y se pregunta si el mouse detectó la ventana de alguna opción, si se detectó se apaga el cursor del mouse y se enciende la ventana de la opción *i*, es decir, se rellena la ventana de la opción con color *opcion_fondo*, y se despliega el letrero de la opción con color *opcion_chr*.

FUNCION MENU



Se llama a la función *sal_ventana*, esta función regresa un valor entero igual a 0 si no se presionó el botón del mouse o si el mouse salió de la ventana, y regresa un valor entero igual 1 si se presionó el botón del mouse; en caso de ser 0 se incrementa el número de opción (*i*) y se inicia de nuevo; si es 1, es decir si se presionó el botón entonces se le asigna a *boton* el número de la opción seleccionada *i*, se restaura la ventana en caso de haber sido salvada y se regresa el valor de *boton* (opción seleccionada).

De la unidad GRAFFUN se describirán las funciones **cursor, correlación, cuenta_intervalos, auto_cor_int** y **FFT**.

La función **cursor** es una de las funciones más elaboradas del sistema GRETA, tiene como objetivo desplegar una señal de un canal determinado y poder escalarla tanto en tiempo como en voltaje, recorrerla a la izquierda o a la derecha sobre el eje del tiempo, medir el voltaje y el tiempo en un punto dado y calcular la integral total o la integral de un segmento de la señal. Tiene como parámetros de entrada un apuntador a enteros *dat* que apunta a la dirección en memoria en donde está almacenada la señal en cuestión, las coordenadas *x,y* de la esquina superior izquierda de la ventana de despliegue, y el número de *canal* de la señal.

El diagrama de flujo de la función se encuentra de la página 90 a la 94 y está dividido en cinco partes: 1. El cuerpo principal de la función, 2. Las opciones de escalamiento, 3. La opción de medición, 4. La opción de integral, y 5. La opción de reset.

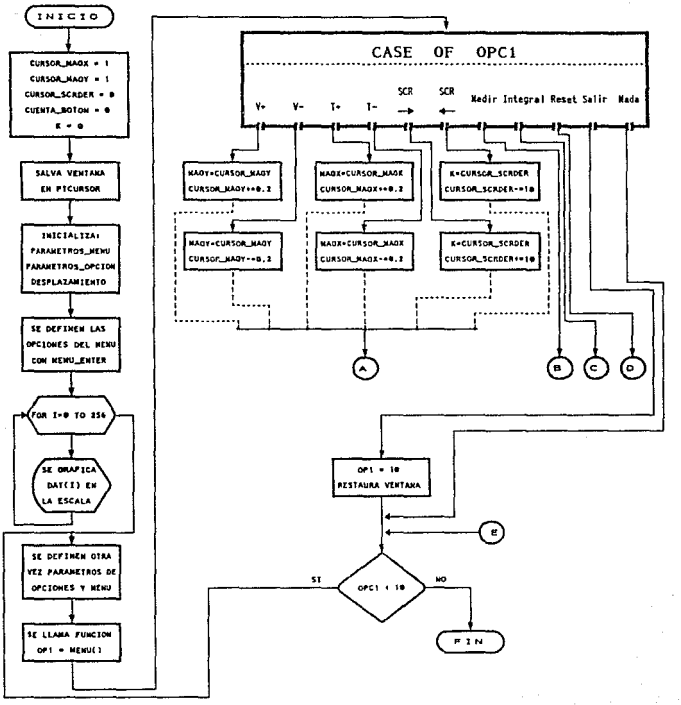
En el cuerpo principal (página 90) se inicializan las variables globales de escalamiento *cursor_magx=1, cursor_magy=1,*

cursor_scrder=0, y las variables locales *cuenta_boton=0* y *k=0*. Se salva la ventana de despliegue en el área de memoria apuntada por *ptcursor*; y se inicializan los parámetros necesarios para hacer el desplegado del menú de opciones, como se describió con la función *menu*.

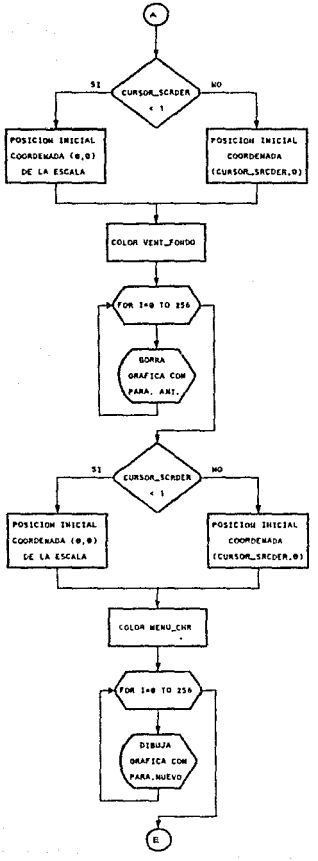
Para las opciones de **escalamiento: V+, V-, T+, T-, SCR→** y **SCR←**, se utilizan las variables locales *magx, magy* y *k* para poder guardar el estado de los parámetros actuales de escala, y así poder inicializar las variables globales mencionadas anteriormente como los parámetros nuevos de escala. Cada vez que el usuario elije alguna de estas opciones con el mouse, el parámetro en cuestión se incrementa o decrementa según sea el caso; para el voltaje lo hace en ±0.2 volts, para el tiempo en ±0.2 segundos y para el corrimiento en ±10 pixeles. Una vez que se ha modificado el parámetro de escala se borra la señal con los parámetros actuales y se vuelve a dibujar la nueva señal con los parámetros nuevos como se muestra en el diagrama de la página 91 correspondiente a la opción CURSOR-A.

Al elegir la opción de **medición** (página 92 CURSOR-B) se enciende el cursor del mouse y el usuario debe mover el mouse hacia la ventana de escala donde se encuentra la señal graficada, en el momento en que el mouse detecta la ventana de escala se apaga el cursor, y aparece un cursor cruz en la posición x del mouse sobre la señal. El usuario puede mover el mouse en el sentido que desee y el cursor cruz se desplazará sobre la señal, en caso de que el mouse salga de la ventana de escala desaparecerá el cursor cruz y

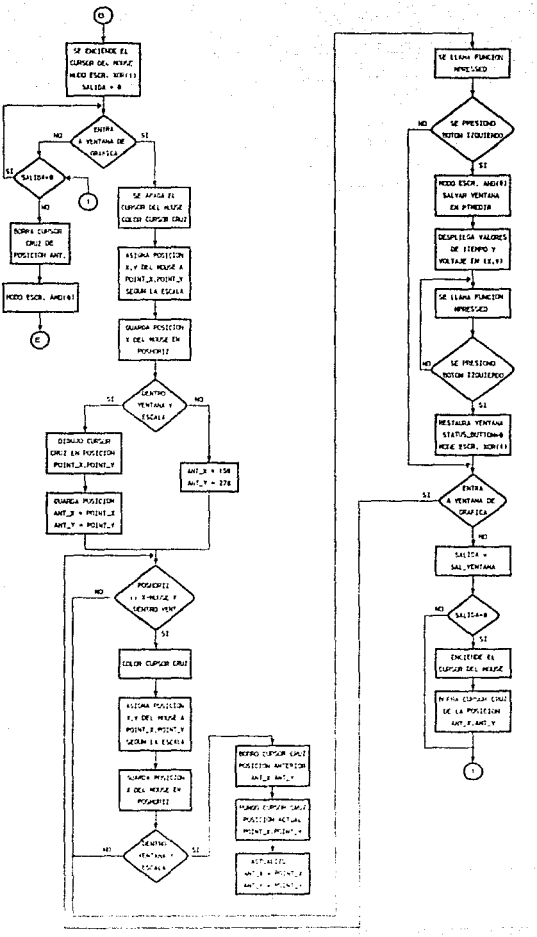
FUNCION CURSOR



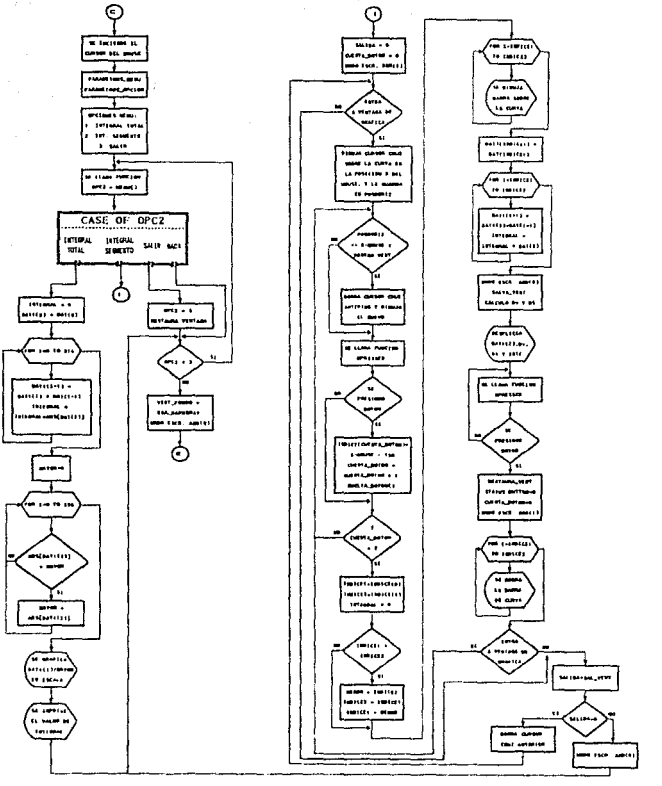
CURSOR-A OPCIONES DE ESCALAMIENTO



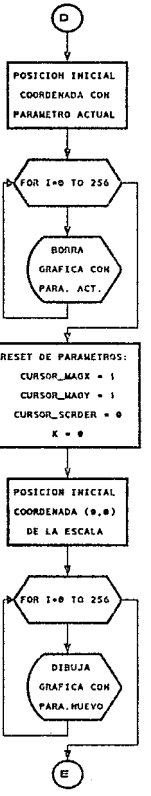
CURSOR-B OPCION DE MEDICION



CURSOR-C OPCION DE INTEGRAL



CURSOR-D OPCION DE RESET



volverá a encenderse el cursor del mouse. Una vez que el usuario posicione el cursor en el punto deseado debe oprimir el botón izquierdo del mouse y aparecerá una ventana donde se desplegará el valor de voltaje y tiempo de el punto seleccionado. La ventana de voltaje y tiempo permanecerá desplegada hasta que el usuario presione nuevamente el botón izquierdo. Se podrán seguir midiendo los puntos que se quiera hasta que se presione el botón derecho del mouse y manteniéndolo oprimido se salga de la ventana de escala, entonces se regresará al menú de opciones.

Al seleccionar la opción **integral** aparecerá un menú con las opciones *integral total* de la señal o la *integral de un segmento* (ver página 93 CURSOR-C). Si se selecciona la opción de **integral total** primero deben inicializarse las variables *dat1(0)=dat(0)*, e *integral=0*; el arreglo *dat1* contendrá en cada una de sus elementos la suma acumulada de los elementos anteriores de *dat*, *dat1(i+1)=dat1(i)+dat(i+1)*; y el float *integral* contendrá la suma acumulada de los valores absolutos de los elementos de *dat*, *integral=integral+abs(dat(i))*; de manera que en *integral* se tendrá el valor numérico de la integral de la señal rectificada. Una vez calculado lo anterior, se despliega una ventana en donde se grafica la curva de los valores acumulados (*dat1*), y el valor numérico de la integral total. La ventana de la integral permanecerá desplegada hasta que el usuario presione el botón izquierdo del mouse con lo que volverá al menú de integrales.

Si se selecciona, sin embargo, la opción de **integral de segmento**, la selección del segmento se hace de manera similar a la

opción medición, con el cursor cruz, con la diferencia que se deben seleccionar dos puntos que serán los extremos del segmento. Al seleccionar el primer punto el cursor cruz permanecerá en esa posición y aparecerá otro cursor cruz para marcar el segundo punto; una vez marcados los dos, se calculan los índices y se dibuja una barra color azul que marca el segmento seleccionado. En base a estos índices se calcula, de manera análoga que con la integral total, el arreglo *dat1* de los elementos acumulados de *dat*, y el valor de la *integral* así como de Δv y Δt del segmento. Una vez calculado lo anterior aparece una ventana donde se despliega la gráfica de los valores acumulados de la señal original y sus correspondientes valores numéricos. Esta ventana permanecerá hasta que el usuario presione el botón izquierdo del mouse con lo cual volverá al menú de integrales. Para salir del menú de integrales deberá seleccionar la opción **salir** con lo que regresará al menú de opciones.

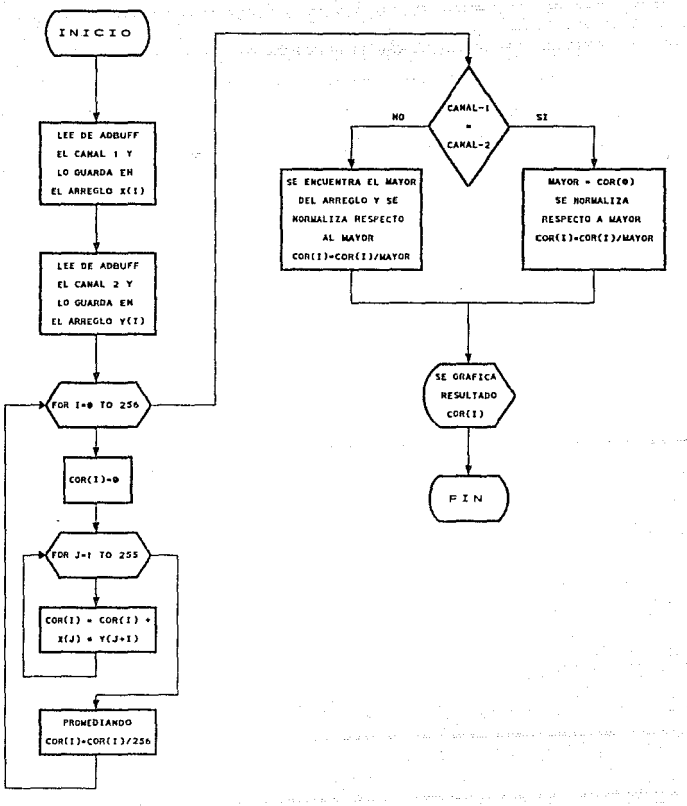
Finalmente, la opción de **reset** (página 94 CURSOR-D) tiene como objetivo regresar la señal al tamaño y posición original. Primero se borra la señal con los parámetros actuales de escala; luego se inicializan los parámetros nuevos con los valores originales: *cursor_magx=1, cursor_may=1, cursor_scrder=0* y *k=0*; y por último se dibuja la señal con los parámetros nuevos.

Para terminar la función *cursor* se debe seleccionar la opción **salir**, con lo cual se restaura la ventana de despliegue.

En la función **correlación** se implementa el método de autocorrelación y correlación cruzada descrito en la sección III.2, en base a la ecuación 15. En este caso se trata de funciones continuas (procesos continuos) por lo que la correlación que hace este algoritmo es de forma de onda, es decir, correlación de amplitudes. El diagrama de flujo de la función *correlación* se encuentra en la página 98.

La función *correlación* tiene como parámetros de entrada el número de los canales a correlacionar, si canal1=canal2 se trata de una autocorrelación de amplitudes; y si canal1≠canal2 se trata de una correlación cruzada de amplitudes. En base al número del canal se lee la señal de *adbuff* en los arreglos X(j) y Y(j). Las señales están compuestas de muestras de j=0,1,2,...,256 por lo que j representa al número de muestras de la señal; i en este caso representa el número de intervalos de correlación (el tamaño de C en la ecuación 15) que va de i=0,1,2,...,256; por lo que estamos considerando en este algoritmo que el intervalo $C=1$, y se está haciendo la correlación punto a punto de las señales. En el arreglo *cor(i)* se obtienen los productos acumulados *X(j) * Y(j+i)* de la correlación para cada intervalo i. Una vez calculada la correlación la normalización se hace en base al siguiente criterio: si se trata de una autocorrelación siempre resulta que el primer punto del arreglo *cor* es el de la máxima correlación por lo que *mayor=cor(0)* y se normaliza el arreglo respecto a *mayor*; si se trata de una correlación cruzada no necesariamente el primer elemento del arreglo es el de máxima correlación por lo que debe buscarse el

FUNCION CORRELACION

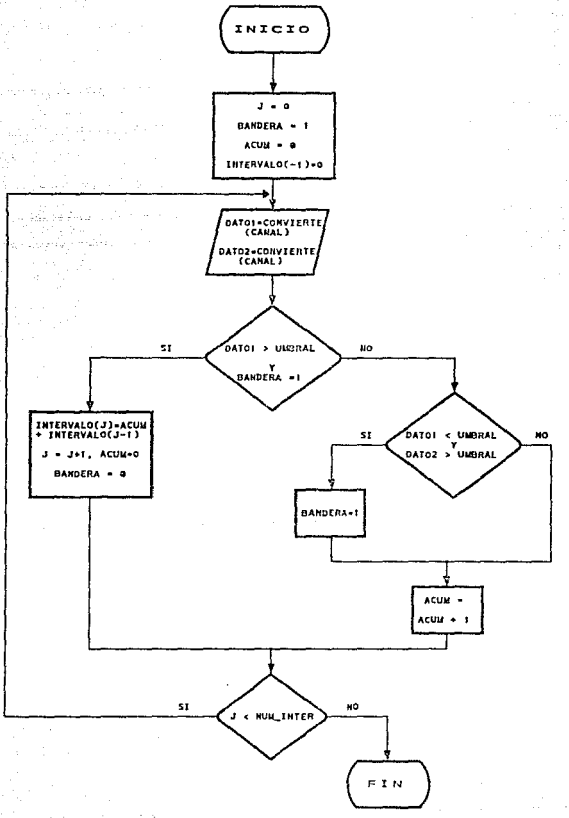


elemento mayor del arreglo y normalizar respecto a *mayor*. Una vez normalizado se grafica el arreglo *cor* en la escala.

Las funciones **cuenta_intervalos** y **auto_cor_int** se utilizan junto con la función **canal_umbral** para obtener la autocorrelación de intervalos. En este caso se trata entonces de procesos puntuales (como trenes de espigas) en donde la información relevante no es la amplitud de las espigas sino los intervalos de tiempo que las separan entre sí. La función **canal_umbral** despliega una señal de 538 puntos del canal indicado por el usuario, y permite establecer en línea, con el uso mouse, el *umbral* respecto al cual se va a convertir la señal continua en un proceso puntual.

La función **cuenta_intervalos** tiene como parámetros de entrada el *número de canal* y el *número de intervalos*, es decir el número de espigas a contar. Esta función tiene como objetivo recorrer cada punto de la señal y compararlo con el *umbral* para establecer el inicio y el final de un cierto intervalo y llevar la cuenta del tamaño de dicho intervalo en el arreglo *intervalo(j)*, todo lo anterior en línea hasta llegar al número de intervalos que se deseen contar. Es importante hacer notar que el algoritmo de **cuenta_intervalos** sólo considera el inicio de un nuevo intervalo cuando la señal cruza el *umbral* de subida y no de bajada. El diagrama de flujo de esta función se muestra en la página 100.

FUNCION CUENTA_INTERVALOS



En la figura 25 se muestra el trazo típico de un tren de espigas con intervalos aleatorios.

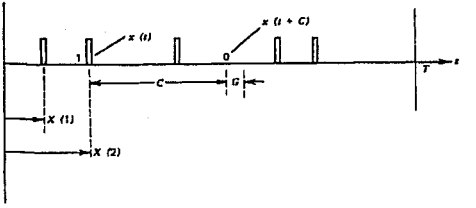
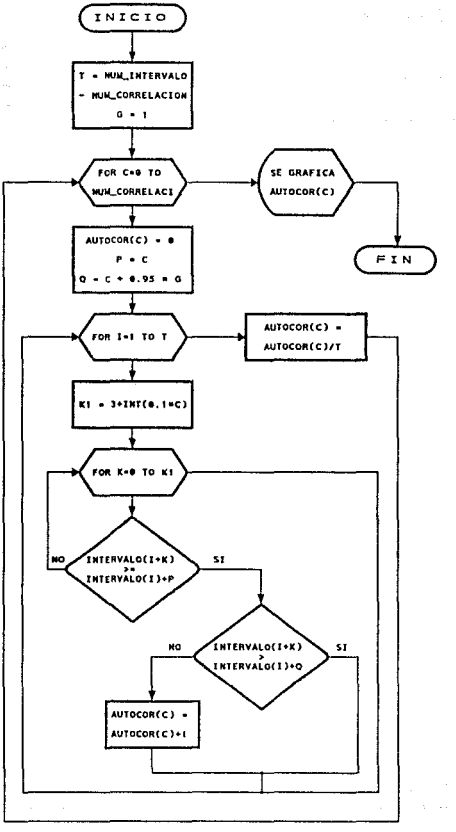


Figura 25. Intervalos aleatorios interespiga. Para formar la función de autocorrelación en el intervalo C se realiza el producto x(t) · x(t+C) para cada espiga. El producto puede tomar solo los valores de 1 o 0. [24]

La sustitución directa de los valores x1, x2,... en la ecuación 15 nos puede dar un resultado erróneo. En la función **auto_cor_int** se aplica un algoritmo para calcular la autocorrelación de intervalos que se describirá en base a la ecuación 15 y a la figura 25 [24]. El diagrama de flujo de la función se muestra en la página 102. La función tiene como parámetros de entrada el número del *canal*, el *número de intervalos* (el número de espigas) y el *número de correlaciones* (el tamaño de la ventana de correlación).

La señal de la figura 25 representa un tren de espigas con intervalos aleatorios interespiga. Generalmente, las amplitudes de las espigas no son de interés particular para el investigador (en el caso de un proceso puntal), a quien sólo le interesa encontrar

FUNCION AUTO_COR_INT



las propiedades de los intervalos interespigas. Por lo que se normalizarán todas las amplitudes a un valor de 1.

Para un intervalo de tiempo *C* dado, el producto de la ecuación 15 puede sólo tener dos valores: $x(t) \cdot x(t+C) = 1$, si la segunda espiga es encontrada dentro de la ventana *G* a una distancia *C* de la primera espiga; o $x(t) \cdot x(t+C) = 0$, si no existe una segunda espiga dentro de la ventana *G* a una distancia *C* de la primera espiga.

Para calcular la función de autocorrelación para un intervalo de tiempo *C* dado, el procedimiento anterior debe ser aplicado a cada una de las espigas del tren. Para cada intervalo *C* se deben realizar las siguientes operaciones: inicialmente consideramos la ventana *G=1*, se deben calcular los límites superior e inferior (*p* y *q*, respectivamente) de la ventana. Para cada espiga, deben examinarse las diez espigas siguientes. Si alguna de ellas aparece en la ventana de tiempo, se suma un 1 al valor acumulado de la función de correlación. De otra manera, el valor acumulado de la función de correlación permanece sin cambio.

El valor acumulado de la función de correlación se va guardando en el arreglo *autocor*, por lo que al finalizar el proceso de correlación, éste debe graficarse.

Finalmente la función **FFT** tiene como parámetros de entrada tres enteros: *m* una potencia de 2, que en el caso particular de las señales que se adquieren con el sistema GRETA es 8 ya que $2^8 = 256$, que es el número total de puntos que conforman una

señal; *isign* que indica si la transformación es directa (*isign=1*), o si la trasformación es inversa (*isign=-1*); y el número de *canal* donde se quiere aplicar el método. El resultado de la transformación se almacena en un arreglo de números complejos *xf*, el cual está definido como se muestra en el encabezado de la unidad GRAFFUN (Apéndice B) por dos números reales: *xf[i].re*, y *xf[i].im*. Para graficar el espectro de potencia se hace frecuencia vs potencia, donde la potencia se calcula como $P=(Re)^2+(Im)^2$. Para iniciar el método la señal que es leída del área de memoria apuntada por *adbuff* debe guardarse en el arreglo *xf* en su parte real, e inicializar la parte imaginaria de dicho arreglo con 0.

El método que se aplica en esta función, como ya se mencionó anteriormente es el llamado *Transformada Rápida de Fourier* (TRF, FFT del inglés Fast Fourier Transform), el cual se describirá brevemente a continuación [3].

La expresión de la *Transformada Discreta de Fourier* (TDF), ecuación 36 de la sección III.2, se puede expresar en forma matricial como sigue:

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(K) \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W(0,0) & W(0,1) & \cdots & W(0,n) & \cdots & W(0,N-1) \\ W(1,0) & W(1,1) & \cdots & W(1,n) & \cdots & W(1,N-1) \\ \vdots & \vdots & & \vdots & & \vdots \\ W(k,0) & W(k,1) & \cdots & W(k,n) & \cdots & W(k,N-1) \\ W(N-1,0) & W(N-1,1) & \cdots & W(N-1,n) & \cdots & W(N-1,N-1) \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(n) \\ x(N-1) \end{bmatrix}$$

donde n y k son los índices de las columnas y renglones de la

matríz, respectivamente. El término $W(k,n)$ es llamado **factor "vuelta"** y generalmente se denota por: $W(k,n) = W_N^i \qquad (i=nk)$.

El cual es un número complejo de magnitud unitaria cuyas partes real e imaginaria son el coseno y el seno del ángulo $(-2\pi i/N)$ en radianes. Por lo tanto:

$$W_N^i = \exp\left(\frac{-j2\Pi i}{N}\right) = \cos\left(\frac{2\Pi i}{N}\right) - j\sin\left(\frac{2\Pi i}{N}\right)$$

Los coeficientes del primer renglón y de la primera columna de la matriz son unitarios. En general, el cálculo de la TDF para N puntos requiere de N^2 multiplicaciones complejas, y algunas de ellas son multiplicaciones por 1.

El algoritmo de la *Transformada Rápida de Fourier*, desarrollado originalmente por Cooley y Tukey en 1965, es un método computacional intensivo significativamente más rápido que el cálculo directo de la TDF. El algoritmo está basado en el principio de calcular una transformación grande por medio de un número de transformaciones pequeñas más fáciles de manejar. Por ejemplo, considérese la descomposición de una transformada de N puntos, donde N es par, en un par de transformadas de N/2 puntos.

Existen dos métodos para llevar a cabo el resultado. El primero llamado **Decimación en Tiempo** (DIT del inglés decimation-in -time), realiza las dos transformaciones de N/2 puntos utilizando

los elementos par e impar de la secuencia de entrada respectivamente. Estas dos transformadas se unen después por otras N/2 transformadas de dos puntos TDF para generar los elementos de salida deseados. El segundo método, llamado **Decimación en Frecuencia** (DIF, del inglés decimation-in-frequency), realiza las mismas dos operaciones, pero en orden inverso. Las muestras de entrada se procesan primeramente en pares por N/2 transformaciones TDF de dos puntos, seguida de las dos transformaciones de N/2 puntos las cuales generan componentes pares e impares directamente a la secuencia de salida.

El algoritmo de la función FFT implementado en el sistema GRETA se basa en el método de *decimación en frecuencia*. La figura 26 ilustra el algoritmo de descomposición DIF, muestra cómo es acarreada la descomposición, una manera de recordar cuál combinación de elementos (muestras) se usan en el primer estado de la descomposición DIF, es arreglar los índices de los elementos en columnas de N/2 elementos, llenando las columnas desde arriba sucesivamente. Por lo que para una transformación de 8 puntos tendríamos la descomposición DIF siguiente:

0	4
1	5
2	6
3	7

indicando que las cuatro transformaciones TDF de dos puntos tienen los elementos pares indicados por los índices de los renglones 0 y 4, 1 y 5, 2 y 6, 3 y 7.

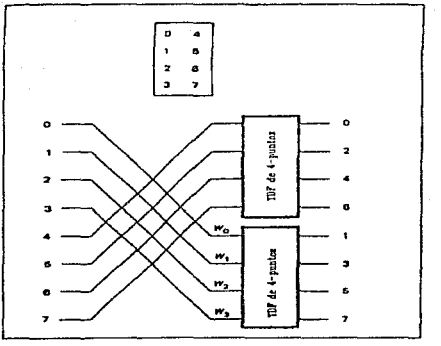


Figura 26. Descomposición de TDF de 8 puntos en 2 transformaciones de 4 puntos utilizando el algoritmo de decimación en frecuencia (DIF). [D]

Se deben señalar dos características importantes, primero que el índice del elemento asociado con una localidad de almacenamiento en particular cambia durante el curso del proceso, en el algoritmo DIF la entrada está fuera de secuencia (revuelta) y la salida está en orden correcto; y segundo, que seguido de el primer estado de transformaciones, algunos de los elementos intermedios son multiplicados por un "factor vuelta" constante Wi (complejo).

Normalmente se combinan la TDF de dos puntos y la multiplicación "vuelta" en un solo proceso llamado **"mariposa"**. En la mariposa DIF, el dato es "volteado" después del cálculo de la mariposa. En ambos algoritmos (el DIF y el DIT) es necesario realizar dos multiplicaciones complejas y dos adiciones complejas.

La ventaja del uso de la descomposición radica en que el tiempo que toma realizar dos trasformaciones de (N-1) puntos es menor al tiempo que toma realizar una trasformación de N puntos; por lo que esto incrementa notablemente el tiempo de ejecución.

La técnica de descomposición tiene una gran aplicación práctica cuando se desea realizar FFT de bloques de datos muy grandes, pues en ocasiones no pueden ser manejados en el espacio de memoria de datos. El bloque es dividido en dos o más sub-bloques los cuales pueden ser alojados en memoria, y cada uno es transformado separadamente aplicando la FFT. Los pares de elementos se procesan con un calculo mariposa para formar secuencialmente la salida del tamaño completo del bloque.

Como hemos visto, el corazón del cálculo de la FFT es el proceso conocido como **mariposa**. La mariposa de base dos (radix-2 butterfly) está definida para dos entradas y dos salidas. Se pueden utilizar mariposas de bases mayores para reducir el tiempo de cálculo de la FFT en algunos casos; aquí nos enfocaremos sólo en la implementación la mariposa de base 2.

La figura 27 muestra tres representaciones gráficas de la mariposa base 2 del algoritmo DIF. Una representación resalta la TDF fija de dos puntos, otra es una representación en símbolos de diagrama de flujo, y la tercera muestra el símbolo convencional de la mariposa. Si la señal compleja muestreada a la entrada de la mariposa se designa como (XPR + jXPI) y (XQR + jXQI) respectivamente, y si WR y WI se utilizan para representar la parte real y la parte imaginaria del factor "vuelta", entonces las

salidas de la mariposa están dadas por:

$$XP(salida) = (XPR + XQR) + j(XPI + XQI)$$

$$XQ(salida) = [(XPR - XQR) \cdot WR - (XPI - XQI) \cdot WI]$$

$$+ j[(XPI - XQI) \cdot WR + (XPR - XQR) \cdot WI]$$

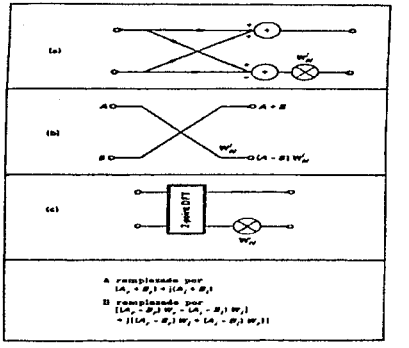


Figura 27. Tres representaciones de la mariposa DIF. a) Representación de diagrama de flujo. b) Símbolo convencional de representación. c) Modelo de TDF de dos puntos. [3]

El procedimiento en Pascal para la implementación de la mariposa base 2 sería el siguiente:[3]

```
procedure butterfly2f (var x:signal; p,q:integer; wr,wi:real);
```

*x es el arreglo de la señal, p y q son índices de arreglos. x[2*p] y x[2*q] son las partes reales, x[2*p+1] y x[2*q+1] son las partes imaginarias. wr y wi son las partes real e imaginaria de Wi.*

```
var
 tr,ti        : real;      { almacenamiento temporal }
begin
 tr:= x[2*p] - x[2*q];
 ti:= x[2*p+1] - x[2*q+1];
 x[2*p]:= x[2*p] + x[2*q];
 x[2*p+1]:= x[2*p+1] + x[2*q+1];
 x[2*q]:= tr * wr - ti * wi;
 x[2*q+1]:= tr * wi + ti * wr;
end; {butterfly2f}
```

El arreglo completo de los valores muestra de la señal, es parámetro de entrada, al igual que los índices designados como *p* y *q*, de las dos muestras a ser procesadas por la mariposa. También son parámetros de entrada *wr* y *wi* que son la parte real e imaginaria del factor "vuelta". El arreglo de entrada es real, con los elementos real e imaginario guardados en localidades consecutivas. Se utilizan internamente dos variables temporales *tr* y *ti*. Finalmente los elementos *p* y *q* de la entrada son sobre escritos con los valores del resultado.

TDF base 2 de Cooley-Tukey. El método consiste en la descomposición de la transformada de N puntos en transformadas de N/2 puntos, como se ilustró anteriormente, pero luego continuar con la descomposición de cada transformada de N/2 puntos a transformadas de N/4 puntos y así sucesivamente hasta conseguir que queden sólo transformadas de 2 puntos. La figura 28 ilustra el proceso de aproximaciones por DIF, aquí se puede ver, como ya se mencionó, que la FFT involucra aproximadamente N $\log_2$ N multiplicaciones complejas comparado con N^2 del cálculo con la implementación directa de TDF.

El arreglo utilizado para almacenar la entrada es ahora sobre escrito después de cada estado.

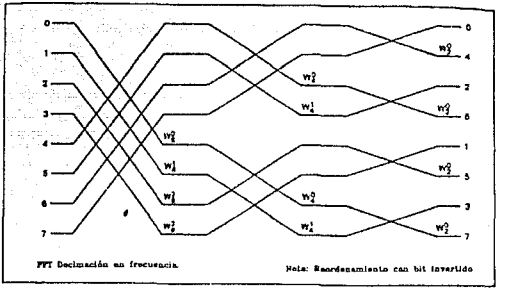


Figura 28. FFT base 2 de Cooley-Tukey formada por la descomposición de la TDF en pares de TDFs de tamaño medio, utilizando el algoritmo DIF. [3]

Otra característica del algoritmo completo de FFT es el rearreglo de los índices de los elementos. Para una descomposición sencilla en dos TDF de tamaño medio la revoltura consiste de elementos con índices pares seguidos de elementos con índices impares. Sin embargo, en una descomposición entera, es decir, para varias descomposiciones sucesivas, resultan repetidas revolturas pares/impares, en un reordenamiento general que corresponde a la *inversión de bit* (bit reversal) de los índices.

La tabla siguiente ilustra los pasos involucrados en el reordenamiento de los elementos de salida del algoritmo DIF:

Posición del elemento en el arreglo	Código binario	Código de inversión de bit	Posición correcta del elemento
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

No siempre es necesario reordenar la salida, como ejemplo cuando sólo se necesita un componente sencillo, o cuando el resultado de la TDF (después de que se realiza algún proceso en el dominio de la frecuencia) debe ser entrada de una rutina asociada con una TDIF. Nótese que la operación de revoltura/ordenamiento es sólo una secuencia de operaciones en la que los contenidos de los pares de localidades de memoria son intercambiados.

Programación del algoritmo. Refiriendonos a la figura 28, la FFT DIF puede ser implementada con tres ciclos anidados. El ciclo externo cubre todos los estados, el siguiente ciclo cubre cada uno de los factores "vuelta" de cada estado, y el ciclo más interno cubre las mariposas asociadas con cada factor "vuelta". El diagrama de flujo del algoritmo se muestra en la página 115.

La tabla siguiente muestra que son necesarias sólo dos variables para controlar el proceso en cada estado, cada variable debe reducirse a la mitad al completar cada estado. Etiquetando el índice de la entrada superior de la mariposa como *P* y el índice de la entrada inferior como *Q*, luego la primera mariposa que involucra

al factor Wi tiene *P=i*, la siguiente *P=i+(periodo "vuelta")*, etc. hasta que todos los elementos son agotados. El término *wing span* (lapso de ala) es utilizado por la variable *(Q-P)*.

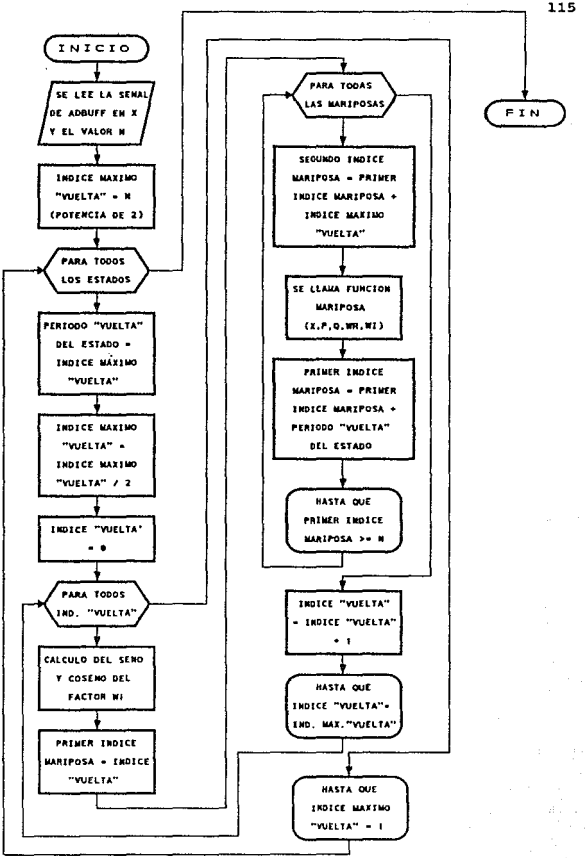
	Valor de inicio	Siguiente estado	Variable
Subíndice "vuelta"	N	x 1/2	N1
Período "vuelta"	N	x 1/2	N1
"vuelta"/ estado	N/2	x 1/2	N2
Wing span	N/2	x 1/2	N2

En seguida se da un ejemplo de un procedimiento en Pascal para la FFT DIF de base 2, utilizando el procedimiento de mariposa descrito anteriormente:[3]

```
procedure fft2df(var x:signal; n:integer);

El tamaño de n debe ser potencia de 2. Las muestras de la
señal están en x, las partes reales son enumeradas con
índices pares.

var
 p    : integer;       {entrada mariposa primer índice}
 q    : integer;       {entrada mariposa segundo índice}
 wr,wi   : real;          {Re(W), Im(W)}
 n1   : integer;       {periodo "vuelta" del estado}
 n2      : integer;       {índice "vuelta" máximo del estado}
 j    : integer;       {índice "vuelta"}

begin
 n2 := n;
 repeat                      {todos los estados}
    n1 := n2;
    n2 := n2 div 2;
    j := 0;
    repeat                   {todos los índices "vuelta"}
```

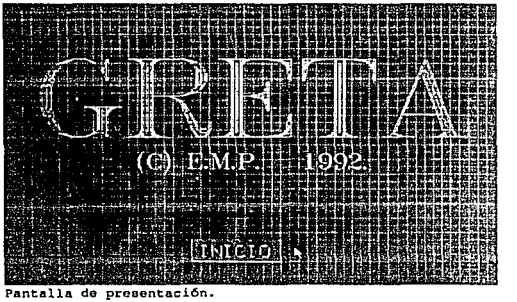
```
        wr := cos(2*pi*j/n1);
        wi := -sin(2*pi*j/n1);
        p := j;
        repeat              {todos los inicios de mariposa}
        q := p + n2;
        butterfly2f(x,p,q,wr,wi);
        p := p + n1;
        until p >= n;
        j := j + 1;
    until j = n2;
 until n2 = 1;
end;
```

FUNCION FFT


INICIO
SE LEE LA SEÑAL DE ADBUFF EN X Y EL VALOR N
INDICE MAXIMO "VUELTA" = N (POTENCIA DE 2)
PARA TODOS LOS ESTADOS
PERIODO "VUELTA" DEL ESTADO = INDICE MÁXIMO "VUELTA"
INDICE MAXIMO "VUELTA" = INDICE MAXIMO "VUELTA" / 2
INDICE "VUELTA" = 0
PARA TODOS IND. "VUELTA"
CALCULO DEL SENO Y COSENO DEL FACTOR W1
PRIMER INDICE MARIPOSA = INDICE "VUELTA"
PARA TODAS LAS MARIPOSAS
SEGUNDO INDICE MARIPOSA = PRIMER INDICE MARIPOSA + INDICE MAXIMO "VUELTA"
SE LLAMA FUNCION MARIPOSA (X,P,Q,WR,WI)
PRIMER INDICE MARIPOSA = PRIMER INDICE MARIPOSA + PERIODO "VUELTA" DEL ESTADO
HASTA QUE PRIMER INDICE MARIPOSA >= N
INDICE "VUELTA" = INDICE "VUELTA" + 1
HASTA QUE INDICE "VUELTA"= IND. MAX."VUELTA"
HASTA QUE INDICE MAXIMO "VUELTA" = 1
FIN

III.5. FUNCIONAMIENTO GENERAL DEL SISTEMA.

El sistema GRETA se corre tecleando la palabra GRETA en el prompt de DOS y la tecla **enter** con lo que aparece la siguiente pantalla:

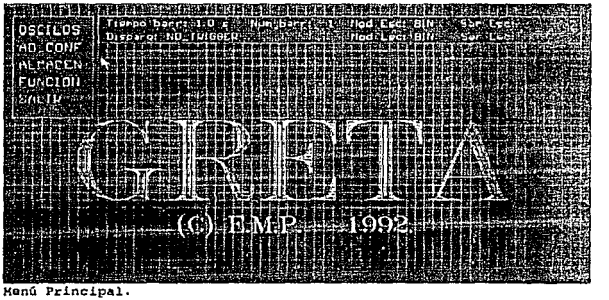


Pantalla de presentación.

El sistema GRETA, como se mencionó anteriormente, hace uso del mouse, por lo que la selección de las opciones de cada menú se hacen oprimiendo el botón izquierdo del mouse.

Para iniciar, el mouse debe posicionarse en el cuadro de **INICIO** con lo cual éste cambiará de color y al oprimir el botón del mouse aparecerá la pantalla con el menú principal, y un recuadro con los parámetros de configuración del sistema, así como los créditos.

El menú principal consta de las opciones: OSCILOS, AD_CONF, ALMACEN, FUNCION y SALIR, cada una de las cuales se describirán en esta sección.



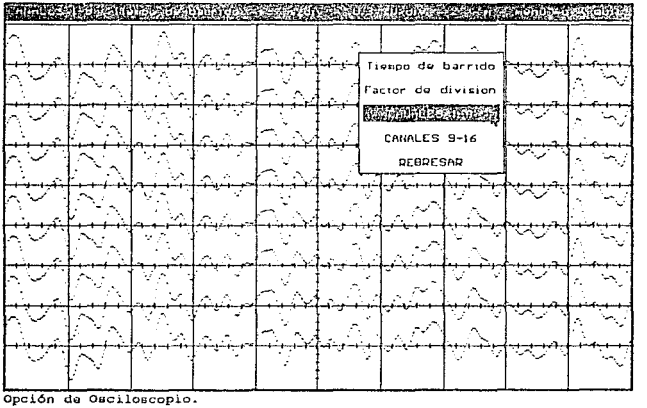
Menú Principal.

El recuadro de parámetros contiene la siguiente información: *Tiempo de barrido* en segundos, que se refiere al tiempo en que va a adquirir 256 puntos por canal; *Disparo*, indica el modo de disparo con el que se va a usar el convertidor; *Num barrido*, es el número de barridos (256 puntos x 16 canales) que va a contener la serie a trabajar; *Modo Esc* y *Modo Lec*, son los modos (BINARIO o ASCII) con los que se van a escribir o a leer los archivos de las series en disco; y finalmente *Ser Esc* y *Ser Lec*, indican los nombres que llevarán en disco las series de escritura y las de lectura, respectivamente.

La opción OSCILOS permite el desplegado de 8 canales a la vez en una pantalla de osciloscopio, la escala en volts/división y segundos/división se selecciona a través de un menú llamado oprimiendo la tecla m con lo cual se desplegarán las opciones de: cambiar el *Tiempo de barrido* (en esta opción es para 640 puntos por

8 canales); *Factor de división*, para seleccionar los volts/división (el valor máximo de voltaje es para señales de ±2 volts); selector de canales: *Canales 1-8* o *Canales 9-16*; y la opción de *Regresar* con lo que se inicializan los parámetros y se sigue viendo el desplegado de las señales.

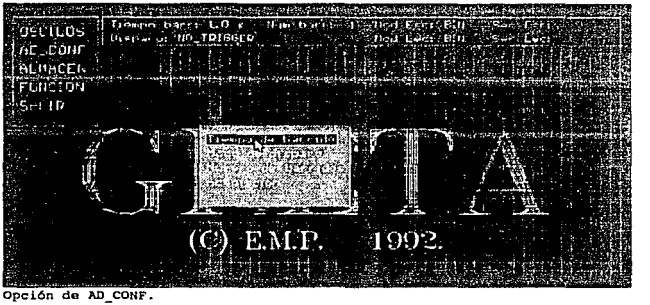
Esta opción es útil para hacer los ajustes necesarios a las señales con las que se trabajarán posteriormente con el sistema; para terminar esta opción debe oprimirse la tecla **q** de quit. Y regresamos así al menú principal.



Opción de Osciloscopio.

Con la opción de **AD_CONF** se configura la tarjeta PCLAB-812. Los parámetros de configuración son: *Tiempo de barrido*, con el que se configura la velocidad de conversión; *Modo de disparo*, que tiene

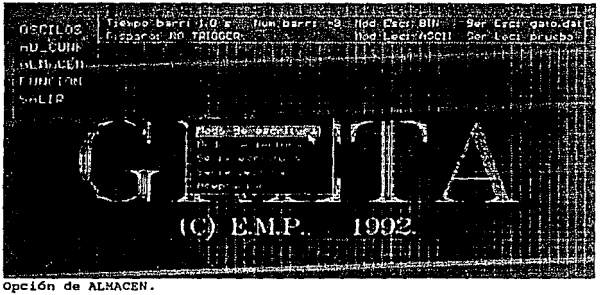
tres opciones: *NO_TRIGGER* = inicio de conversión inmediato, *KB_TRIGGER* = inicio de conversión sólo cuando se oprima una tecla, y *TTL_TRIGGER* = inicio de conversión sólo cuando haya una transición de 0 a 1 de un pulso externo TTL de sincronía; *Numero de barridos*, que indica el número de barridos que tendrá la serie en cuestión; y finalmente *Aceptación* con la que se aceptan los cambios hechos y se regresa al menú principal.



Opción de AD_CONF.

La opción **ALMACEN** permite hacer los cambios de los parámetros que se refieren al almacenaje de las series en disco. Los primeros dos parámetros son: *Modo de escritura* y *Modo de lectura*, estos modos son dos: **BINARIO** o **ASCII**, que es el modo en que los datos de la serie se graban físicamente en el disco; con modo BINARIO se tienen archivos con un tamaño de 16,388 bytes, y con el modo ASCII el tamaño es de 28,679 bytes, sin embargo tienen la ventaja de ser portables a otros sistemas. Con los parámetros *Serie escritura* y

Serie lectura se le da el nombre del archivo en disco con el que se guardarán las series, o el nombre de la serie que se quiera leer del disco; cuando una serie es leída del disco, el modo de lectura debe corresponder al modo de escritura con la que fué creada, de otro modo hará caso omiso y se desplegarán las señales que fueron leídas la última vez. Y finalmente con la opción *Aceptación* se regresa al menú principal.

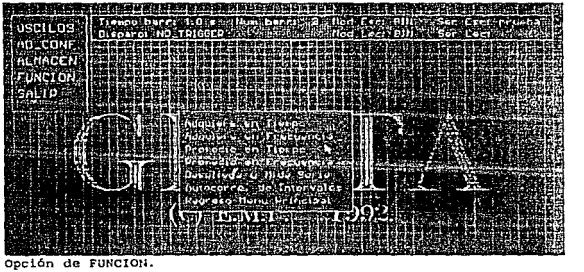


Opción de ALMACEN.

La opción **FUNCION** despliega otro menú en el que tenemos todas las funciones de análisis tanto en tiempo como en frecuencia.

Primero tenemos la **Adquisición en Tiempo**, con esta opción se adquieren *n* barridos con un modo de *disparo* específico, configurados en la opción AD_CONF; y se guardan directamente al disco con el modo y el nombre previamente configurados con la opción ALMACEN. Por ejemplo, si se configuró para una adquisición de 3 barridos, con disparo TTL_TRIGGER y con el nombre de la serie

prueba, los barridos se guardarán en disco, una vez recibida la

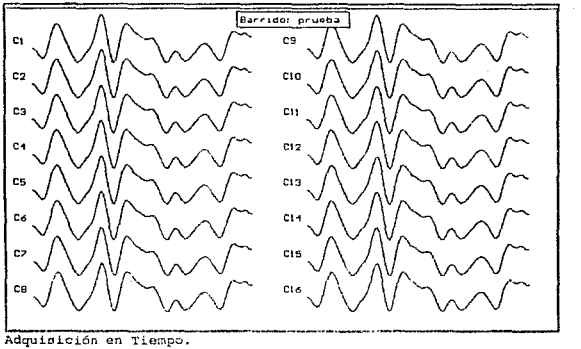


Opción de FUNCION.

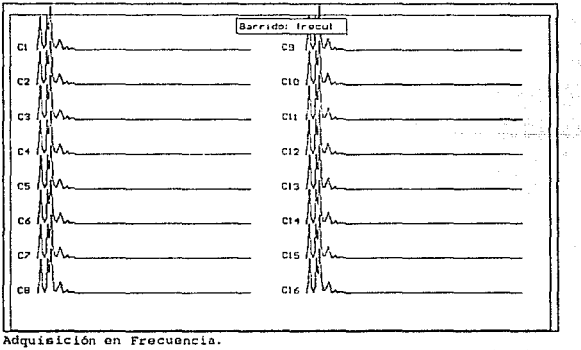
señal de sincronía, como *prueba*, *prueba1* y *prueba2*; antes de iniciar la adquisición pregunta si se quiere desplegar en la pantalla o no. Esta función adquiere utilizando *adq_16_can*, y se despliega con *despliega16* .

La opción de **Adquisición en Frecuencia** realiza la misma operación que en tiempo con la diferencia obvia que antes de gradar en disco se calcula la FFT de cada canal.

Al final de cada archivo se escribe un **0**, si se trata de un archivo en tiempo, y un **1** si se trata de un archivo en frecuencia, de manera que cuando se leen nuevamente de disco se puede diferenciar de que dominio es la información. A continuación se muestran dos pantallas de ejemplo.



Adquisición en Tiempo.

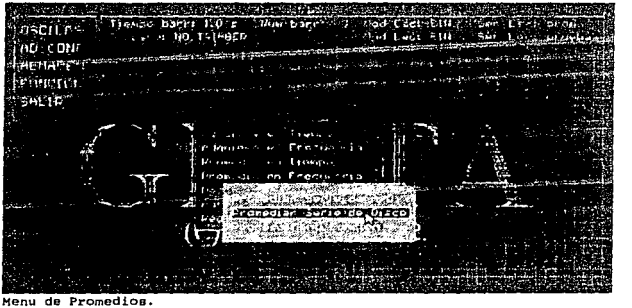


Adquisición en Frecuencia.

En esta función se invocan las rutinas de *adq_16_can*, *FFT* y *despliega16*.

La función de **Promedio en Tiempo** promedia una serie de *n* barridos, definida en *Ser Lec;* y guarda el promedio en disco con el nombre definido en *Ser Esc*, con el modo configurado en *Mod Esc.*

Al seleccionar la opción de promediar en tiempo se despliega otro menú para seleccionar de dónde va a ser tomada la fuente del promedio, ya que la serie de lectura puede ser una serie tomada directamente del convertidor, (por lo que no es necesario definir una serie de lectura), o bien, puede ser leída del disco una serie previamente adquirida. Por ejemplo, si se quiere promediar una serie en tiempo llamada *prueba*, y guardar el resultado en un archivo llamado *prom*, esta opción realiza el promedio sumando uno a uno cada barrido: *prom=(prueba+prueba1+prueba2)/ num_barrido*. También en este caso se pregunta si se desea ver desplegado el promedio o no.

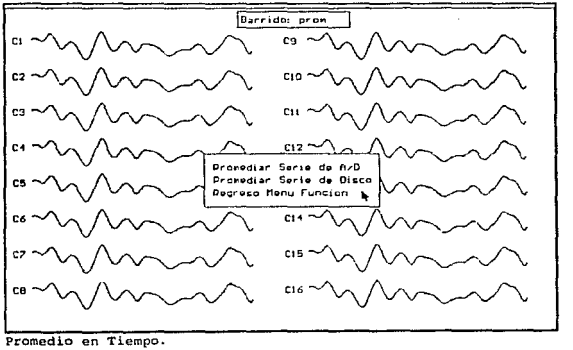


Menu de Promedios.

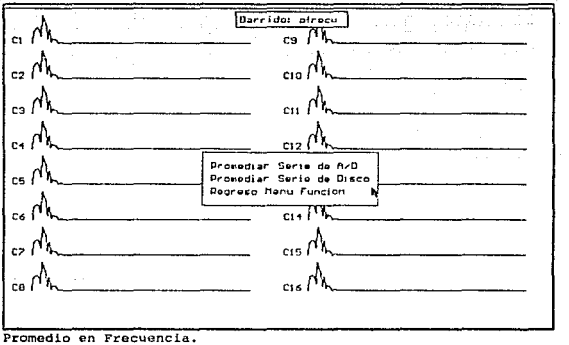
Para esta función se utilizan las rutinas de *adq_16_can* y *despliega16*.

La función **Promedio en Frecuencia** sigue la misma filosofía que el promedio en tiempo, pero en este caso las series promediadas, ya sea del convertidor o del disco, deben ser series en frecuencia. Si la serie se promedia del disco, sólo se leen los archivos ya adquiridos y se promedian; si la serie se promedia directamente del convertidor, debe adquirirse en tiempo aplicar la FFT y luego promediar y guardar en disco el resultado.

A continuación se muestran dos pantallas de ejemplo, en la primera se promedia en tiempo la serie *prueba* y el resultado se guarda en *prom*; y en el segundo se promedia la serie en frecuencia *frecu* y el resultado se guarda en el archivo *pfrecu*.



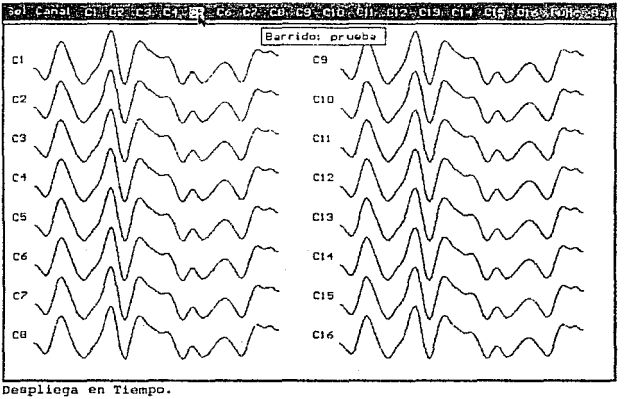
Promedio en Tiempo.



Promedio en Frecuencia.

La función de **Despliega y Mide Serie** permite desplegar la serie definida en la opción ALMACEN, utilizando la función *despliega16*, ya sea en tiempo o en frecuencia, y hacer mediciones a un determinado canal.

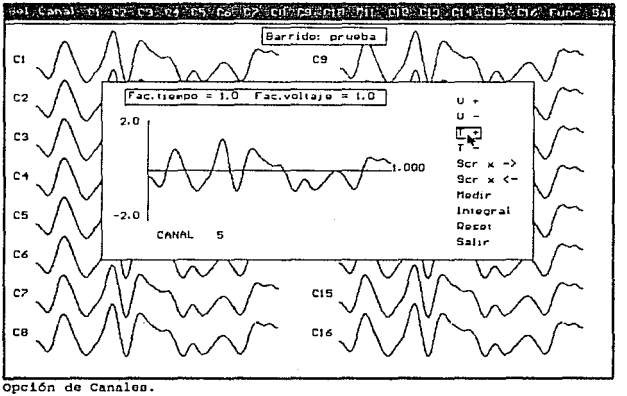
Si la serie a medir es en tiempo, se despliega el primer barrido de la serie y aparece en la cabecera de la pantalla un menú con el número de canal *C1, C2,..., C16,* y las opciones de *FUNción* y *SALir*.



Despliega en Tiempo.

Al seleccionar un canal, se despliega la señal del canal en una ventana de medición, como se muestra en la siguiente figura; esta ventana contiene un menú de opciones de medición; esta es la parte que realiza la función *cursor* descrita en la sección III.4.

Primero se encuentran las opciones de escala **V+**, **V-**, **T+**, **T-**, **SCR →**, y **SCR ←** , con las cuales se escala la señal en voltaje, en tiempo y se recorre sobre el eje horizontal a la derecha o a la izquierda. Esta pantalla se ilustra a en la siguiente página.

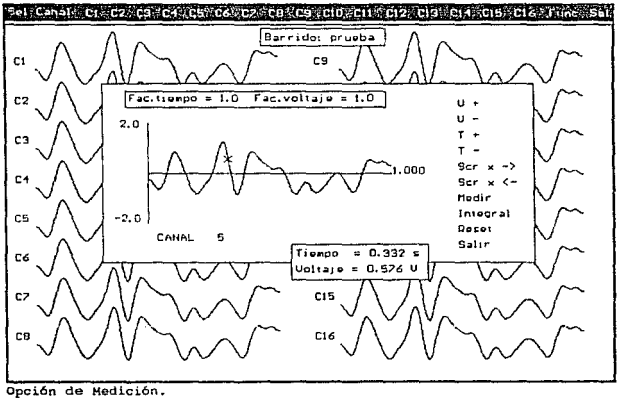


Opción de Canales.

A continuación se encuentra la opción de **Medir**. Al seleccionar esta opción, el usuario debe acercar el mouse hacia la ventana de despliegue de la señal, en el momento en que el mouse entra a la ventana, el cursor se apaga y se enciende un cursor cruz sobre la señal; este cursor cruz se desplaza sobre la señal conforme el usuario mueve el mouse.

Cuando el usuario posiciona el cursor cruz en el punto en que desee tomar una medición, debe presionar el botón izquierdo con lo que se desplegará una ventana con el valor del tiempo y voltaje del punto. Esta ventana de medición permanecerá hasta que el usuario presione nuevamente el botón izquierdo con lo que regresará a la opción de seleccionar otro punto.

Si el usuario desplaza el mouse fuera de la ventana de despliegue, el cursor cruz se apaga y se enciende nuevamente el cursor típico del mouse. El usuario saldrá de la opción de medición presionando el botón derecho, y manteniéndolo oprimido deberá salir de la ventana de despliegue.

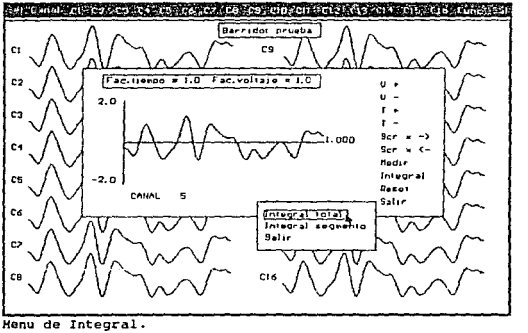


Opción de Medición.

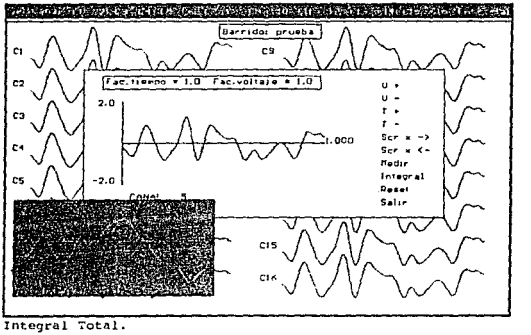
La opción que sigue es la de **Integral**, al seleccionar esta opción aparece un segundo menú en el cual se elige si la integral será **total** o de un **segmento** de la señal.

Si la integral es total se despliega una segunda ventana con la curva de la integral total y el valor numérico de la integral rectificada. La ventana de la integral total permanecerá desplegada hasta que el usuario presione el botón izquierdo del mouse, con la

que regresará a las opciones de *integral*.



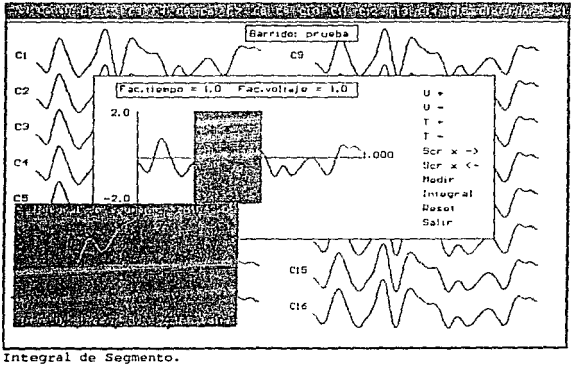
Menu de Integral.



Integral Total.

Si la integral es de un segmento de la señal, entonces al seleccionar la opción, el usuario deberá seguir las mismas indicaciones que con la opción de medición, pero ahora deberá marcar sobre la curva dos puntos que serán los límites del segmento sobre el cual se calculará la integral.

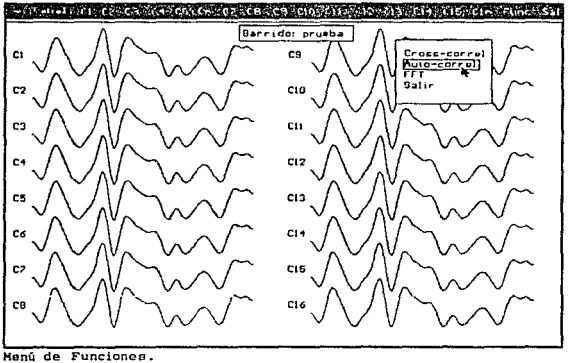
Una vez marcados los dos puntos, se desplegará una segunda ventana con la curva de la integral del segmento, el valor de la integral rectificada, y los valores de Δt y Δv del segmento. De la misma manera que en el caso anterior, la ventana permanecerá hasta que el usuario presione el botón izquierdo del mouse con la que se regresará al menú de integrales. Para salir de la opción *Integral* se deberá seleccionar **salir**.



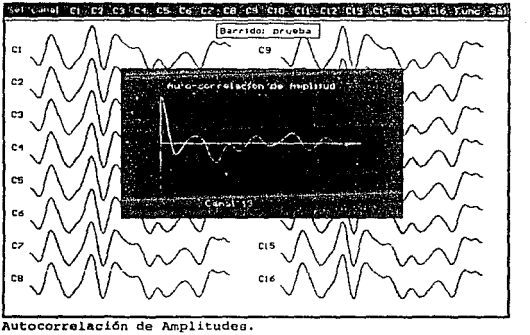
Integral de Segmento.

Finalmente, resta la opción de **Reset** con la que se recuperará la señal original, en el caso de haberla escalado. Para regresar al menú de la opción de *Desplegar y Medir Serie* deberá seleccionarse la opción **Salir.**

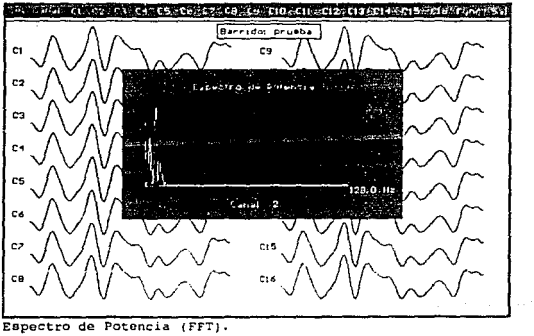
La siguiente opción de *Desplegar y Medir Serie* es la de **FUNCiones**, con esta opción se puede hacer el cálculo de: **Correlación Cruzada y Autocorrelación de Amplitudes** y **FFT**, de un canal seleccionado. Cada una de estas opciones pregunta por el número del canal con el que se va a operar; en el caso de la correlación cruzada pregunta por el número de los dos canales a correlacionar.



Menú de Funciones.



Autocorrelación de Amplitudes.

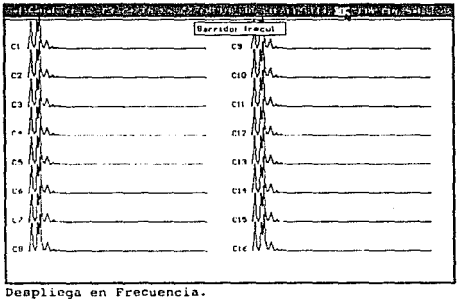


Espectro de Potencia (FFT).

En cada uno de los casos anteriores, las ventanas de despliegue de curvas permanecerán en la pantalla hasta que el usuario presione el botón izquierdo del mouse con lo que volverá al menú de funciones. Para salir del menú de funciones deberá seleccionarse la opción **Salir**.

La última opción del menú de *Despliega y Mide Serie* es la opción de **SALir**; al oprimir esta opción cambiará el desplegado del barrido; es decir, si se está trabajando con la serie *prueba*, y el barrido actual es *prueba*, al oprimir la opción *SALir* se desplegará el barrido *prueba1* y así sucesivamente hasta que el número de barridos contenidos en el serie se termine; en cuyo caso, se regresará al menú de *FUNCIONES* del menú principal.

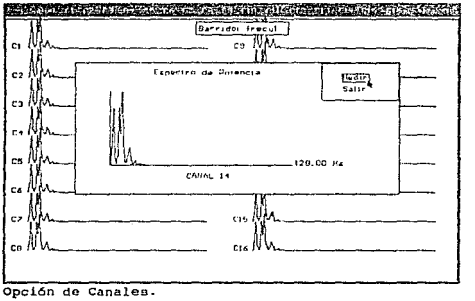
Cuando lo que queremos desplegar y medir son series en frecuencia, al seleccionar la opción *Despliega y Mide Serie*, se despliega el primer barrido de la serie como se muestra:



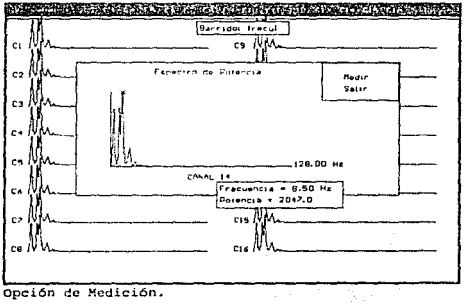
Despliega en Frecuencia.

La diferencia con el menú de series en tiempo es que no se cuenta con la opción de FUNCiones.

Ahora bien, al seleccionar alguno de los canales se despliega en una ventana el espectro de potencia con un menú que lo único que permite hacer es mediciones de frecuencia y potencia:



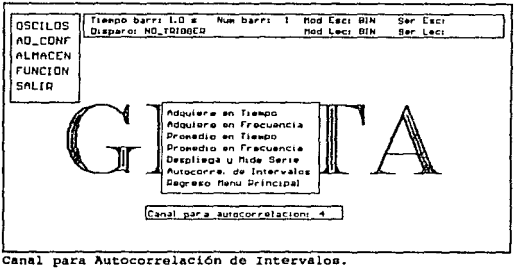
Opción de Canales.



Opción de Medición.

La última función de la opción *FUNCIONES* del menú principal, es la opción de **Autocorrelación de intervalos.**

Al seleccionar esta opción primero se pregunta por el número del canal a autocorrelacionar.

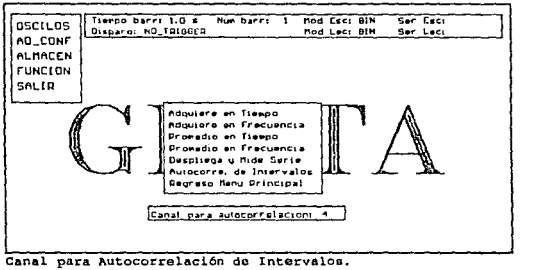


Canal para Autocorrelación de Intervalos.

Una vez seleccionado el canal aparecerá una ventana en donde se despliega en línea la señal del canal con 512 puntos. Para seleccionar el nivel del umbral, primero se presiona una vez el botón izquierdo del mouse, con lo que se detiene la señal y se enciende el cursor del mouse; el usuario podrá mover el mouse en la posición deseada para fijar el umbral, al presionar nuevamente el botón aparecerá una línea color verde que indica la posición del umbral.

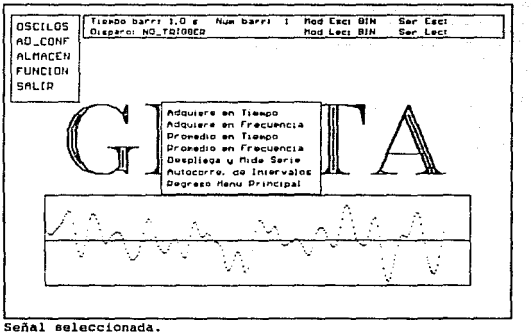
La última función de la opción *FUNCIONES* del menú principal, es la opción de **Autocorrelación de intervalos.**

Al seleccionar esta opción primero se pregunta por el número del canal a autocorrelacionar.

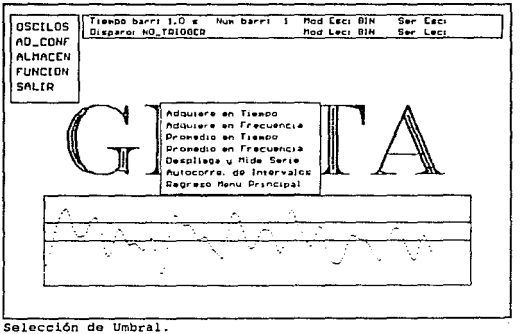


Canal para Autocorrelación de Intervalos.

Una vez seleccionado el canal aparecerá una ventana en donde se despliega en línea la señal del canal con 512 puntos. Para seleccionar el nivel del umbral, primero se presiona una vez el botón izquierdo del mouse, con lo que se detiene la señal y se enciende el cursor del mouse; el usuario podrá mover el mouse en la posición deseada para fijar el umbral, al presionar nuevamente el botón aparecerá una línea color verde que indica la posición del umbral.

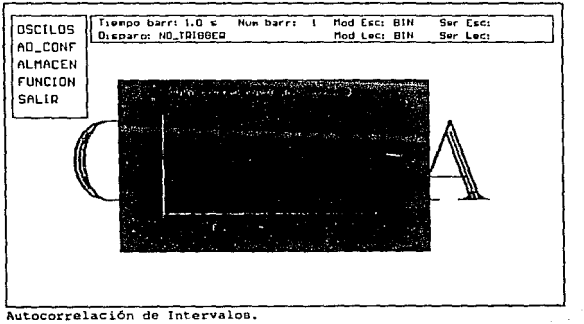


Señal seleccionada.



Selección de Umbral.

Se podrá repetir esta labor tantas veces sea necesario hasta que el umbral quede en la posición deseada; una vez hecho esto se presiona el botón derecho del mouse con lo que se inicia el cálculo de la autocorrelación de intervalos. El algoritmo cuenta 512 espigas, y guarda en un arreglo los 512 intervalos, la autocorrelación la realiza con una ventana de 256 puntos.

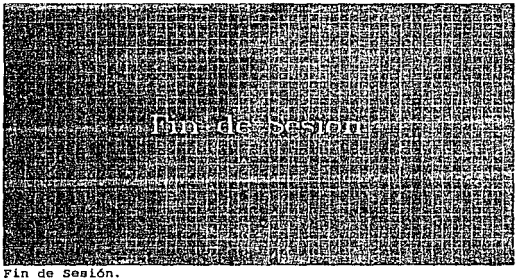


Autocorrelación de Intervalos.

La ventana de autocorrelación de intervalos permanecerá desplegada hasta que el usuario presione el botón izquierdo del mouse, con lo que regresará al menú de FUNCIONES del menú principal.

Para regresar al menú principal se deberá elegir la opción **Regreso al Menu Principal.**

Finalmente, una vez en el menú principal, debera seleccionarse la opción **SALIR** para terminar la sesión con el sistema GRETA.

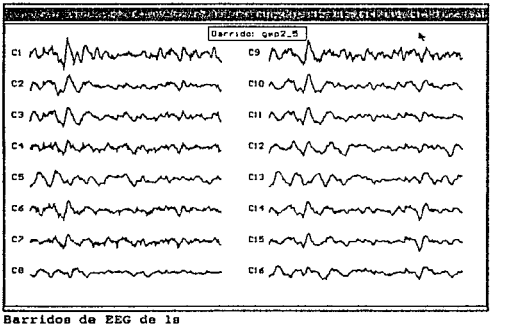


Fin de Sesión.

IV. PRUEBAS Y APLICACIONES.

Análisis en tiempo.

Para el análisis en tiempo se adquirieron 256 barridos de EEG, con pulso de sincronía y con un tiempo de barrido de 1s. Estos registros de EEG corresponden a los potenciales evocados visuales lingüísticos de un sujeto sano.

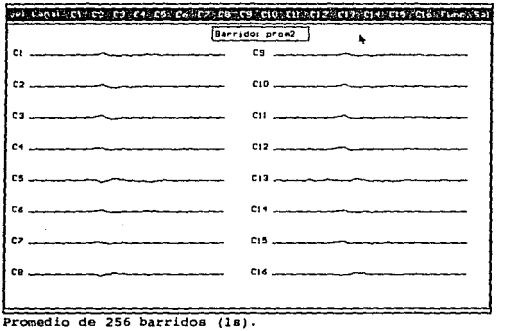


Barridos de EEG de 1s

Para el EEG se tiene por convención la siguiente relación entre canales y derivaciones de electrodos:

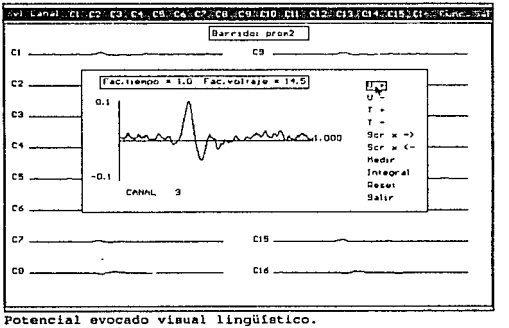
CANAL	DERIVACION	CANAL	DERIVACION
1	Fp1	9	Fp2
2	F3	10	F4
3	C3	11	C4
4	P3	12	P4
5	O1	13	O2
6	F7	14	F8
7	T3	15	T4
8	T5	16	T6

Los potenciales evocados como ya se mencionó anteriormente son señales inmersas en el EEG, por lo tanto para hacerlas resaltar se utiliza el método de promedios en tiempo. En este caso, los potenciales evocados son señales sincronizadas con el estímulo, por lo que al promediar los 256 barridos se eliminan los componentes que están fuera de fase y sólo destacan los que están en fase o sincronizados.

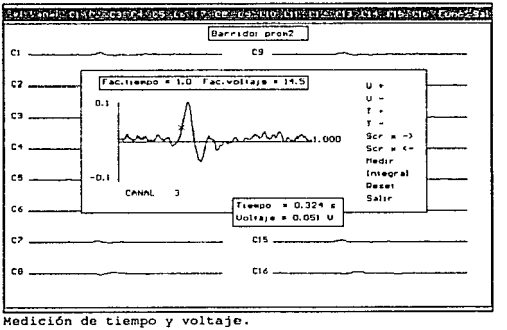


Promedio de 256 barridos (1s).

Con la función cursor se puede amplificar y medir cualquiera de estos potenciales del canal elegido: voltajes, tiempos, latencias, integrales, etc.



Potencial evocado visual lingüístico.
(Derivación C3).



Medición de tiempo y voltaje.

Análisis en Frecuencia.

Para ejemplificar las aplicaciones de análisis en frecuencia se adquirieron 5 barridos de EEG, sin sincronía y con un tiempo de barrido de 4s para poder caracterizar el rango de frecuencias completo del EEG, que es de alrededor de 32 Hz.

Como se mencionó en la sección 1.3. el EEG está dividido en bandas de frecuencia en cada una de las cuales se encuentran caracterizadas las distintas ondas que lo componen:

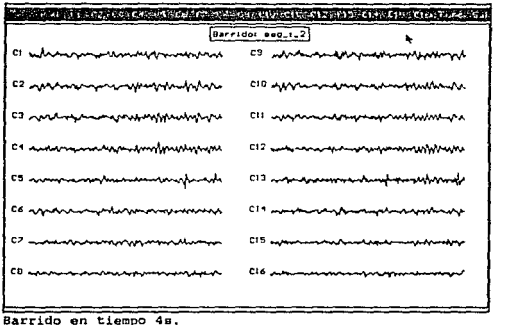
Rango delta:	0.5 - 3.5 Hz
Rango teta :	4.0 - 7.5 Hz
Rango alfa :	8.0 - 13.5 Hz
Rango beta :	14.0 - 22 o más Hz

En el siguiente ejemplo daremos una muestra típica de la caracterización de las ondas alfa.

El ritmo alfa está asociado al estado de vigilia. Se registra en sujetos despiertos, con los ojos cerrados, sin actividad sensorial, psíquica o motriz. Es el ritmo del adulto normal, que aparece en salvas más o menos amplias, con morfología variable, y en él suele ser más frecuente el aspecto sinusoidal; otras veces su aspecto es puntiagudo o "en dientes de cepillo". En realidad, dentro de la banda alfa existen variantes morfológicas y también variaciones de frecuencia, desde las formas más lentas, vecinas al ritmo teta, hasta las fronterizas con el ritmo beta.

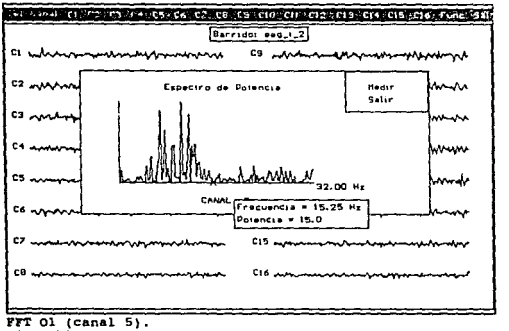
El origen del ritmo alfa se asocia a las regiones posteriores de la corteza (región occipital) con proyección hacia las zonas temporales. Cuando el sujeto abre los ojos existe la intervención de estímulos visuales, que provocan la desaparición del ritmo alfa (bloqueo del ritmo alfa).

A continuación se muestra un barrido en tiempo que corresponde a la adquisición de 4s por canal con ojos abiertos, en él que puede observar la ausencia de ritmo alfa.

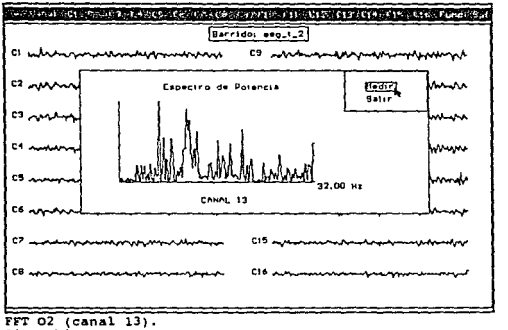


Barrido en tiempo 4s.
Ojos abiertos.

Haciendo ahora el análisis del espectro de potencia de las zonas occipitales (canal 5 = O1, y canal 13 = O2), se muestra en la siguiente página que el espectro de estos dos canales, en especial en el canal 13 (O2), está distribuido a lo largo de toda la escala de 32 Hz por lo que se comprueba que con ojos abiertos no hay ritmo alfa, o por lo menos es muy pequeño.

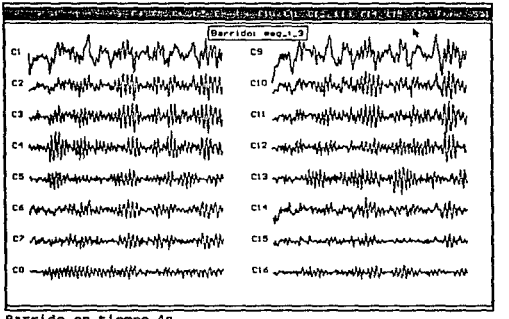


FFT O1 (canal 5).
Ojos abiertos.



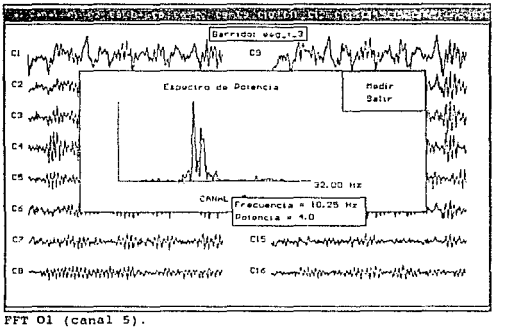
FFT O2 (canal 13).
Ojos abiertos.

Ahora se muestra un barrido en tiempo que corresponde a la adquisición de 4s por canal con ojos cerrados, en él que puede observar la presencia de ritmo alfa.

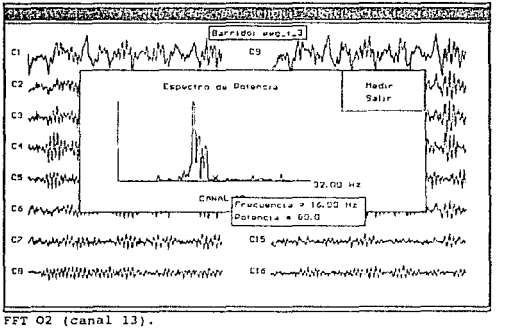


Barrido en tiempo 4s.
Ojos cerrados.

Haciendo también para este caso el análisis del espectro de potencia de las zonas occipitales (canal 5 = O1, y canal 13 = O2), se muestra en la siguiente página que en estos dos canales se caracteriza claramente la banda del ritmo alfa (8 - 13.5 Hz); aunque en el caso de este sujeto la banda está ligeramente por arriba (hasta los 15.5 o 16 Hz), sin embargo, es normal ya que de sujeto a sujeto existen pequeñas variaciones.

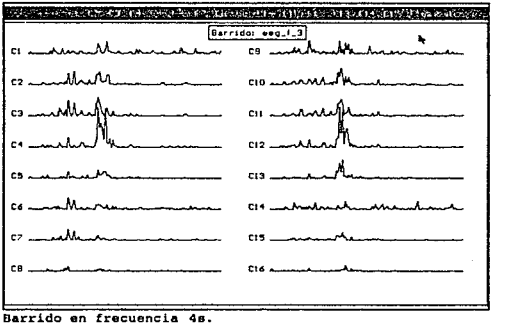


FFT O1 (canal 5).
Ojos cerrados.



FFT O2 (canal 13).
Ojos cerrados.

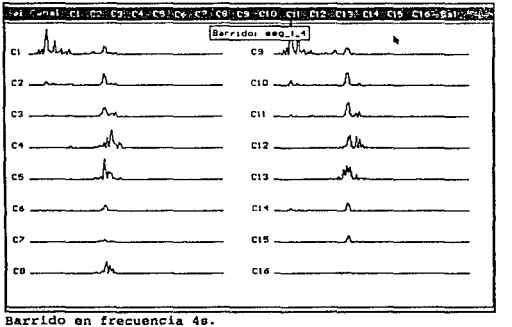
Con el sistema GRETA también es posible hacer el análisis de frecuencia directo, es decir, se adquieren directamente las series en frecuencia. Para aplicar esta opción, se adquirieron 5 barridos, sin sincronía y con un tiempo de barrido de 4s, en frecuencia con ojos abiertos y ojos cerrados, para lo cual se obtuvieron los siguientes resultados:



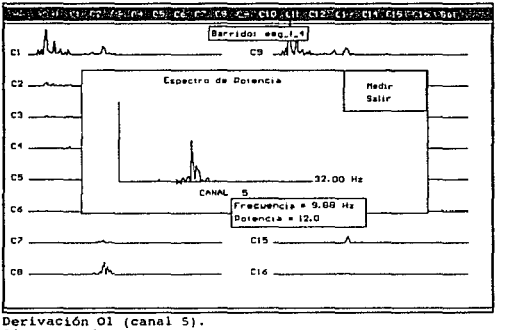
Barrido en frecuencia 4s.
Ojos abiertos.

Aquí se observa que el espectro se encuentra repartido a lo largo de el rango de 32 Hz, aunque es evidente un componente alto en potencia de ritmo alfa.

Enseguida se muestran los resultados con ojos cerrados, donde prácticamente el único componente que se ve es el del ritmo alfa.



Barrido en frecuencia 4s.
Ojos cerrados.

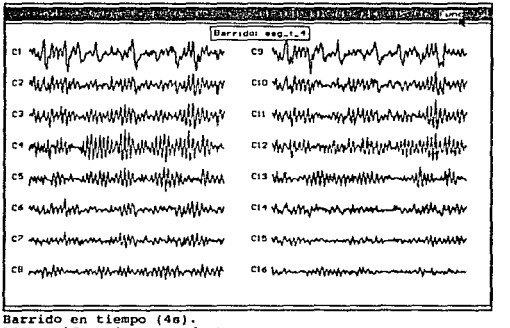


Derivación O1 (canal 5).
Ojos cerrados.

Correlación de Amplitudes.

En el contexto del análisis del EEG la correlación de amplitudes puede dar datos respecto a la periodicidad y fase de dos registros hechos en diferentes derivaciones.

Utilizando un barrido con un tiempo de muestreo de 4s tomado durante la condición de vigilia ojos cerrados, por simple inspección se puede observar que el ritmo alfa es más prominente en las zonas occipitales que en las zonas frontales, ver figura siguiente.

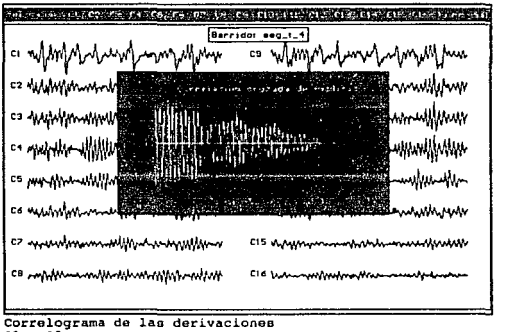


Barrido en tiempo (4s).
Ritmo alfa (ojos cerrados).

Es de esperarse que, si este ritmo se genera en la parte occipital, durante su propagación hacia partes más lejanas, sufra cambios de fase y hasta de frecuencia. La correlación entónces, nos dice si un registro es igual a otro en términos de su morfología,

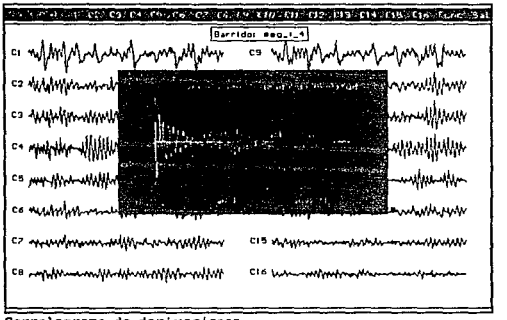
fase y frecuencia.

El sistema GRETA provee un método eficiente para generar el correlograma entre dos canales. En la siguiente figura, se muestra la correlación cruzada entre las derivaciones O1 y O2 que corresponden al lugar donde se genera el alfa. Es importante notar que este correlograma indica una alta sincronización.

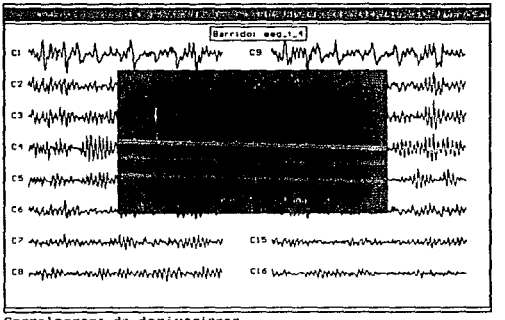


Correlograma de las derivaciones
O1 y O2.

En las siguientes dos figuras se muestran correlogramas de zonas centrales y frontales de ambos hemisferios donde se ve la progresiva pérdida de sincronía de fase del ritmo alfa en su propagación hacia zonas lejanas.



Correlograma de derivaciones
C3 y C4.



Correlograma de derivaciones
Fp1 y Fp2.

V. CONCLUSIONES.

Dentro del campo del análisis de señales existen varios métodos que se consideran entidades básicas para la implementación de programas para análisis digital. En este trabajo, se han elegido los más importantes teniendo en cuenta las necesidades específicas de laboratorios de neurofisiología, cronobiología (análisis de sueño), entre otros.

El sistema GRETA está pensado como un punto de inicio en el cual ya estén resueltos y funcionando métodos como FFT, correlación, etc., dentro de un entorno de adquisición y almacenaje eficientes. Dadas las necesidades específicas de cada laboratorio, GRETA puede ser modificado fácilmente y adaptado para un protocolo experimental muy específico. Estas modificaciones pueden hacerse a nivel código fuente, gracias a que el diseño de este sistema es modular y con jerarquías bien definidas. Cuando se hace una modificación, el usuario tiene acceso a todas las librerías de control del convertidor A/D, gráficas, etc.

En el caso particular de las aplicaciones en los laboratorios donde se desarrollo este sistema, cabe mencionar que se utilizan generalmente programas originales desarrollados específicamente para los experimentos en curso en este lugar, esto se debe a que los programas comerciales de análisis, por ser de propósito general, pierden la capacidad de adaptarse a problemas específicos con protocolos de adquisición complicados. GRETA está diseñado con la idea de resolver estos protocolos experimentales. Además, GRETA

es capáz de generar y adquirir datos de otros muchos programas que actualmente se utilizan en estos laboratorios, lo cual permite que GRETA entre directamente a un experimento que ya está corriendo, evitando hacer modificaciones a los programas y/o algoritmos con los cuales GRETA va a cooperar.

Generalmente en este tipo de experimentos se requiere de análisis en línea, lo cual limita en gran medida la aplicación de programas comerciales de análisis como DADISP, SIGMAPLOT, etc.; GRETA está diseñado con la filosofía de operar en línea para algunos protocolos experimentales.

REFERENCIAS.

1. **Abraira, V., e Ibarz, J.M.** *Spectral estimation of temporal series at unequal intervals*. Computer and Biomedical Research, 19, 203-212 (1986).

2. **Barber, C.** *Evoked Potentials*. University Park Press. Baltimore, 1980, 640.

3. **Bateman, A. y Yates, W.** *Digital Signal Processing Design*. Computer Science Press. New York, 1989, 385.

4. **Borland International.** *Trubo C++ Ver 2.0*. Reference Guide, E.U.A., 1990.

5. **Cohen, A.** *Biomedical Signal Processing. Volume I Time and Frequency Domains Analysis*. CRC Press. Boca Raton, Florida, 1986, 167.

6. **Cohen, A.** *Biomedical Signal Processing. Volume II Compression and Automatic Recognition*. CRC Press. Boca Raton, Florida, 1968, 195.

7. **Duffin, J.** *A cross-correlator for neuronal spike trains*. J. Electrophysiol. Tech., 13, 61-71 (1986).

8. **Frostig, Z. y Frostig, R.D.** *Analysis of frequency components in time series data*. Journal of Neuroscience Methods, 22, 79-87 (1987).

9. **Houk, J.C., Dessem, D.A., Miller, L.E., y Sybirska, E.H.** *Correlation and spectral analysis of relations between single unit discharge and muscle activities*. Journal of Neuroscience Methods, 21, 201-224 (1987).

10. **Huang, C., y White, L.E. Jr.** *High-frequency components in epileptiform EEG*. Journal of Neuroscience Methods, 30, 197-201 (1989).

11. **Jasper, H.H.** *The ten twenty electrode system of the International Fereration*. Electroenceph. Clin. Neurophysiol., 10, 371-375 (1958).

12. **Kelly, T.M., y Chapple, W.D.** *An inexpensive, microcomputer-based system for recording movements in real time*. Journal of Neuroscience Methods, 23, 35-42 (1988).

13. **Kirkwood, P.A.** *On the use and interpretation of cross-correlation measurements in the mammalian central nervous system*. Journal of Neuroscience Methods, 1, 107-132 (1979).

14. **Langdon, R.B.** *Software that detects and analyzes bioelectrical events by assessing the fit of modeled subintervals*. Journal of Neuroscience Methods, 12, 331-341 (1985).

15. **McIntyre, D.C., y Chew, G.L.** *Power spectra analysis of electroencephalographic activity in kindled rats*. Experimental Neurology, 92, 261-266 (1986).

16. **Moore, G.P., Perkel, D.H., y Segundo, J.** *Statistical analysis and functional interpretation of neuronal spike data*. Annual Review of Physiology, 28, 493-522 (1966).

17. **Moore, G.P., Segundo, J., Perkel, D.H., y Levitan, H.** *Statistical signs of synaptic interaction in neurons*. Biophysical Journal, 10, 876-900 (1970).

18. **Norcia, A., Sato, T., Shinn, P., y Mertus, J.** *Methods for the identification of evoked response components in the frequency and combined time/frequency domains*. Electroencephalography an Clinical Neurophysiology, 65, 212-226 (1986).

19. **Novak, J., y Wheeler, B.** *A high-speed multichannel neural data acquisition system for IBM PC compatibles*. Journal of Neuroscience Methods, 26, 239-247 (1989).

20. **O'Flynn, M.** *Probabilities, Random Variables and Random Processes*. Harper & Row Publishers. New York, 1982, 523.

21. **PC-LabCard.** *User's Manual. Model PCL-812 Enhanced Multi-Lab Card.* Lab & Engineering Add-on's for PC/XT/AT.

22. **Schlindwein, F.S., Smith, M.J., y Evans, D.H.** *Spectral analysis of Doppler signals and computation of the normalised first moment in real time using a digital signal processor*. Medical & Biological Engineering & Computing, 26, 228-232 (1988).

23. **Simpson, R., y Houts, R.C.** *Fundamentals of analog and digital communication systems*. Allyn and Bacon, Inc. Boston, 1971, 394.

24. **Souček, B., y Carlson, A.** *Computers in Neurobiology and Behavior*. Wiley-Interscience. New York, 1975, 324.

25. **Tam, D.C., Ebner, T.J., y Knox, C.K.** *Conditional cross-interval correlation analysis with applications to simultaneously recorded cerebellar Purkinje neurons*. Journal of Neuroscience Methods, 23, 23-33 (1988).

APENDICE A. CARACTERISTICAS GENERALES DEL PCLAB-812.

La tarjeta PC-LABCARD-812 es una tarjeta de adquisición de datos (A/D y D/A) que se instala en el slot de una computadora XT o AT. Puede programarse utilizando lenguajes de alto nivel lo que la hace muy versátil.

Características generales.

- 16 canales de entrada analógica en configuración simple
- Un convertidor A/D de 12 bits (HADC574Z), de aproximaciones sucesivas; el cual tiene una frecuencia máxima de muestreo de 30KHz en modo DMA.
- Switch para seleccionar los rangos de entrada de voltaje. Bipolar: ±1V, ±2V, ±5V, ±10V.
- Tres modos de disparo en conversión A/D:
  * Disparo por software.
  * Disparo programable.
  * Disparo por pulso externo.
- La característica de poder transferir conversiones A/D por medio de un programa de control, o transferencia vía DMA.
- Un timer/counter programable (INTEL 8253-5), provee una salida que puede ser utilizada como pulso de disparo para el convertidor A/D, el valor en frecuencia de esta señal puede ir desde 35 minutos/pulso hasta 500 KHz (2μsec). La base de tiempo del temporizador es de 2 MHz. Uno de los contadores de 16 bits está reservado para alguna aplicación que el usuario desee.
- Dos convertidores D/A de 12 bits, con un rango de salida de 0 a 5 volts. Pueden generarse otros rangos de salida utilizando una fuente externa de voltaje como referencia.
- 16 entradas digitales compatibles TTL, y 16 salidas digitales.

A.2

Especificaciones técnicas de la tarjeta PCL-812.

Entradas analógicas (convertidor A/D).

Canales	: 16 en configuración simple.
Resolución	: 12 bits.
Rango de entrada	: Bipolar : ±10V, ±5V, ±2V, ±1V. Todos los rangos de entrada son seleccionados por medio de un switch.
Sobrevoltaje	: Continuo ±30 volts.
Tipo de conversión	: Aproximaciones sucesivas.
Convertidor	: HADC574Z.
Velocidad de conversión	: 30 KHz máximo.
Exactitud	: 0.015% , ±1 bit.
Linealidad	: ±1 bit.
Modo de disparo	: Por software, con el timer programable que tiene la misma tarjeta o por un pulso externo.
Transferencia de datos	: Por un programa de control, control de interrupciones o vía DMA.
Disparo externo	: Compatible TTL, carga 0.4mA máx. a 0.5V (bajo), o 0.05mA máx. a 2.7V (alto).

Salidas analógicas (conversión D/A).

Canales	: Dos.
Resolución	: 12 bits.
Rango de salida	: 0 a +5V con referencia de -5V. ±10V con referencia externa de DC o AC.
Voltaje de referencia	: Interno: -5V (±0.05V) Externo: DC o AC, ±10V máx.
Tipo de conversión	: De 12 bits monolítico multiplicador.
Dispositivos analógicos	: AD7541AKN o equivalente.
Linealidad	: ± ½ bit.
Manejador de salida	: ±5mA máx.
Tiempo de conversión	: 30 microsegundos.

Entradas digitales.

Canales	: 16 bits.
Niveles	: Compatibles TTL.
Voltaje de entrada	: Bajo 0.8V máx., alto 2.0V min.
Carga de entrada	: Bajo 0.4mA máx. a 0.5V. Alto 0.05mA máx. a 2.7V.

Salidas digitales.

Canales	: 16 bits.
Niveles	: Compatibles TTL
Voltaje de salida	: Bajo 8 mA a 0.5V máx. Alto 0.4mA a 2.4V.

Temporizador/Contador programable.

Dispositivo	: INTEL 8253-5.
Contadores	: Tres canales, 16 bits. Dos canales permanentemente conectados al reloj de 2MHz para su utilización por programa, y uno libre para aplicaciones del usuario.
Base de tiempo	: 2 MHz.
Salida de frecuencia	: de 35 minutos/pulso a 0.5 MHz.

Canales de interrupción.

Niveles de interrupción	: IRQ2 a IRQ7, seleccionables por switch.
Habilitación	: Vía S0, S1 y S2 del registro de CONTROL.

Canal de DMA.

Niveles	: 1 o 3, seleccionables por switch.
Habilitación	: Vía S0, S1 y S2 del registro de CONTROL.

Especificaciones generales.

Consumo de potencia	: + 5V :típico 500mA, máx. 1A +12V :típico 50mA, máx. 100mA -12V :típico 14mA, máx. 20mA
Conector E/S	: Conectores de 20 pines.
Dirección base E/S	: Definido por los DIP switches de las líneas A8-A4. Dirección de fábrica 220H.
Temperatura de operación	: de 0 a + 50 C°.
Temperatura de almacenaje	: de -20 a +65 C°.
Peso	: 8.6 oz. (242.89 gm).

ESTRUCTURA Y FORMATO DE LOS REGISTROS.

La tarjeta PCL-812 requiere de 16 direcciones consecutivas de espacio E/S. La característica más importante para programar la tarjeta es entender el significado de los 16 registros de puertos E/S direccionables a partir de la dirección base.

Mapa de direcciones de puertos E/S.

La siguiente tabla contiene la localización de cada uno de los registros y manejadores relativos a la dirección base y sus usos.

Localización	Lectura	Escritura
BASE + 0	Contador 0	Contador 0
+ 1	Contador 1	Contador 1
+ 2	Contador 2	Contador 2
+ 3	N/U	Control de contador
+ 4	Byte bajo A/D	Byte bajo CH1 D/A
+ 5	Byte alto A/D	Byte alto CH1 D/A
+ 6	Byte bajo E/D	Byte bajo CH2 D/A
+ 7	Byte alto E/D	Byte alto CH2 D/A
+ 8	N/U	Inicialización de requerimiento de interrupción.
+ 9	N/U	N/U
+10	N/U	Selector de canal MUX
+11	N/U	Control del PCL-812
+12	N/U	Disparo A/D software
+13	N/U	Byte bajo S/D
+14	N/U	Byte alto S/D
+15	N/U	

*N/U = No se usa.

Registros de datos (A/D).

Los registros de datos (A/D) utilizan las direcciones BASE+4 y +5.

Formato:

1. Byte bajo A/D y dato

BASE+4	D7	D6	D5	D4	D3	D2	D1	D0
	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

2. Byte alto A/D y dato.

BASE+5	D7	D6	D5	D4	D3	D2	D1	D0
	0	0	0	DRDY	AD11	AD10	AD9	AD8

Tamaño:

AD11 a AD0 - Dato Analógico a Digital. AD0 es el bit menos significativo (LSB) y AD11 es el bit más significativo (MSB) del dato.

DRDY - Señal de dato listo. 0 = dato listo, 1 = dato no listo. Una lectura el registro (BASE+5) pondrá este bit en 1 y es inicializado cuando se completa una conversión A/D.

Registro selector MUX.

El registro selector MUX es un registro de sólo escritura que utiliza la dirección BASE+10. Los 4 primeros bits proveen el número del canal. El multiplexor cambia de switch cuando se escribe a este registro.

Formato:

BASE+10	D7	D6	D5	D4	D3	D2	D1	D0
Canal	X	X	X	X	CL3	CL2	CL1	CL0

Tamaño:

CL3 a CL0 - Número de canal.

Registros de E/S digital.

La PCL-812 ofrece 16 canales de entradas y 16 canales de salidas digitales. Estos canales de E/S utilizan las direcciones de entrada BASE+6 y BASE+7. Los puertos de salida están en BASE+13 y BASE+14.

Formato:

BASE+6	D7	D6	D5	D4	D3	D2	D1	D0
(puerto lectura)								
Byte bajo E/D	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

BASE+13	D7	D6	D5	D4	D3	D2	D1	D0
(puerto escritura)								
Byte bajo S/D	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

BASE+7	D7	D6	D5	D4	D3	D2	D1	D0
(puerto lectura)								
Byte alto E/D	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

BASE+14	D7	D6	D5	D4	D3	D2	D1	D0

(puerto escritura)
Byte alto S/D DO15 DO14 DO13 DO12 DO11 DO10 DO9 DO8

Registros de salidas D/A.

Los registros de salidas D/A son registros de escritura y utilizan las direcciones BASE+4, +5, +6, y +7.

Formato:

BASE+4 (D/A # 1 Byte bajo)	D7	D6	D5	D4	D3	D2	D1	D0
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
BASE+5 (D/A # 1 Byte alto)	D7	D6	D5	D4	D3	D2	D1	D0
	X	X	X	X	DA11	DA10	DA9	DA8
BASE+6 (D/A # 2 Byte bajo)	D7	D6	D5	D4	D3	D2	D1	D0
	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
BASE+7 (D/A # 2 Byte alto)	D7	D6	D5	D4	D3	D2	D1	D0
	X	X	X	X	DA11	DA10	DA9	DA8

Tamaño:
DA11 a DA0 - Dato digital a analógico. DA0 es el bit menos significativo (LSB) y DA11 es el bit más significativo (MSB) del dato D/A.

Registro de Control del PCL-812.

El registro de control del PCL-812 es un registro de sólo escritura que utiliza la dirección BASE+11. Este registro provee de información de los modos de operación de la tarjeta.

Formato:

BASE+11 (PLC-812 control status)	D7	D6	D5	D4	D3	D2	D1	D0
	X	X	X	X	X	S2	S1	S0

A. Bajo condiciones internas de disparo:

S1	S2	S3	
0	0	0	: Deshabilita disparos por software y por timer.
0	0	1	: Habilita solo disparo por software
0	1	0	: Habilita disparo por timer usando sólo transferencia DMA.
1	1	0	: Habilita disparo por timer usando

transferencia por programa o transferencia por interrupciones. El jumper JP4 debe ser puesto en la posición "X".

B. Bajo condiciones de fuentes externas de disparo:

S1	S2	S3	
0	0	X	: Habilitando solo disparo externo.
0	1	0	: Habilitando disparo externo usando sólo transferencia DMA.
1	1	0	: Habilitando transferencia por programa o transferencia por interrupciones con fuente de disparo externa.

*Nota : Debe ponerse el jumper JP1 en EXT antes de utilizar cualquiera de estos tipos de disparo.

Registros temporizador/contador de intervalos programables.

La tarjeta PCL-812 utiliza un temporizador/contador de intervalos programable INTEL 8253 versión 5. El 8253 es un dispositivo temporizador/contador muy popular que consiste de tres contadores de bajada independientes de 16 bits. Cada contador tiene una entrada de reloj, una compuerta de control y una salida. Pueden ser programados para contar desde 2 hasta 65535.

La máxima frecuencia de entrada de reloj es de 5MHz para la versión 5 del 8253. La PCL-812 provee una frecuencia de entrada de 2MHz a través de un oscilador de cristal.

Los contadores 1 y 2 están conectados en cascada y operan con una configuración fija de divisores. La entrada del contador 2 está conectada a la frecuencia de 2MHz y la salida del contador 2 está conectada a al entrada del contador 1. La salida del contador 1 está internamente configurada para proveer un pulso de disparo al convertidor A/D. El contador 0 no está reservado para uso interno de la PCL-812, y se puede accesar a través del conector 5.

Los contadores programables del 8253 utilizan cuatro registros localizados en la dirección BASE+0, +1, +2, y +3. La función de cada registro es:

* BASE+0 Contador 0 Lectura/Escritura
* BASE+1 Contador 1 Lectura/Escritura
* BASE+2 Contador 2 Lectura/Escritura
* BASE+3 Palabra de Control de Contador

Como el contador 8253 utiliza una estructura de 16 bits, cada

A.8

Lectura/Escritura de datos es dividida en el byte menos significativo y el byte más significativo. Es importante asegurarse en hacer las operaciones de Lectura/Escritura en pares y mantener el orden correcto del byte.

Formato:

BASE+3	D7	D6	D5	D4	D3	D2	D1	D0
	SC1	SC0	RW1	RW0	M2	M1	M0	BUD

Tamaño:

* SC1 y SC0 - Selector de contador

SC1	SC0	Contador
0	0	0
0	1	1
1	0	2
1	1	Ilegal

* RW1 y RW0 - Selector de operación Lectura/Escritura

RW1	RW0	Operación
0	0	Contador Lach
0	1	Lectura/Escritura LSB
1	0	Lectura/Escritura MSB
1	1	Lectura/Escritura primero LSB y luego MSB

* M2, M1 y M0 - Selector de Modo de operación

M2	M1	M0	Modo
0	0	0	0 - Interrumpido al término de la cuenta.
0	0	1	1 - Un disparo programable.
X	1	0	2 - Generador de velocidad.
X	1	1	3 - Generador de onda cuadrada.
1	0	0	4 - Disparo por software.
1	0	1	5 - Disparo por hardware.

* BUD - Selector de cuentas binarias o BUD

BUD	Tipo
0	Contador Binario de 16 bits.
1	Contador BUD (Binary Coded Decimal) de 4 décadas.

Si los módulos son programados en binario la cuenta puede ser de cualquier número entre 0 a 65535. Si los módulos son programados en BUD las cuentas pueden ser cualquier número de 0 a 9999.

APENDICE B. LISTADO DE LAS PRINCIPALES FUNCIONES.

```
/*  PROGRAMA PRINCIPAL DEL SISTEMA DE ANALISIS GRETA */

#include <string.h>
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include "primifun.h"
#include "graffun.h"
#include "adfun.h"
#include "osciLo.h"


/* Variables globales */

Mresult     * result;
char        buffer[80];

int  far * ptini;
int  far * ptmp;


int         opini;
int         opmp;

void menu_principal(void);

void main()
{

  initgraph(&gdriver,&gmode,"");
  result=Mreset();
  if(!(result->present))
  {
     setcolor(EGA_YELLOW);
     settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
     settextjustify(CENTER_TEXT,CENTER_TEXT);
     moveto(getmaxx()/2,getmaxy()/2);
     sprintf(buffer,"No hay mouse intalado. Adios!");
     outtext(buffer);
     exit(-1);
     restorecrtmode();
  }

  Mshow(FALSE);
  presentacion();
  titulo();

  settextjustify(LEFT_TEXT,BOTTOM_TEXT);
  settextstyle(SMALL_FONT,HORIZ_DIR,8);
  parametros_menu(250,400,350,430,CYAN,LIGHTCYAN,WHITE,1,ptini);
  parametros_opcion(LIGHTGRAY,DARKGRAY,WHITE,1);
  desplaza(0,-8,0);
  menu_enter("INICIO");
  opini=menu();
```

B.2

```
  menu_principal();
  clrscr();
  closegraph();
  restorecrtmode();
}


/*************************************************/
/*** funcion que despliega el menu principal   ******/
/*** y ejecuta todas sus opciones              ******/
/*************************************************/
void menu_principal(void)
{
  int  far *ptmp;
  int  far *ptconf;
  int  far *pttrig;
  int  far *ptalma;
  int  far *ptesc;
  int  far *ptlec;
  int  far *ptfun;
  int  far *ptver;
  int  far *ptprot;


  int       opmp;
  int       opconf;
  int       optrig;
  int       opalma;
  int       opesc;
  int       oplec;
  int       opfun;
  int       opver;
  int       opprot;
  float     mayor;

  int       i,j,k,l;
  int       esc,lee;
  float far *f;


  ad_init();
  set_ad_rate(sw_time);
  set_trigger_mode(trigger_mode);
do{
  par_config();

  settextjustify(LEFT_TEXT,BOTTOM_TEXT);
  settextstyle(SMALL_FONT,HORIZ_DIR,6);
  parametros_menu(8,15,95,150,EGA_GREEN,EGA_LIGHTGREEN,EGA_WHITE,0,ptmp);
  parametros_opcion(EGA_DARKGRAY,EGA_LIGHTGRAY,WHITE,10);
  desplaza(-3,-3,1);
  menu_enter("OSCILOS");
  menu_enter("AD_CONF");
  menu_enter("ALMACEN");
  menu_enter("FUNCION");
  menu_enter("SALIR");

  opmp=menu();
  switch (opmp)
  {
     case 1: /* OSCILOS */
           presentacion();
```

```
        osciloscopio();
        presentacion();
        titulo();
        break;


case 2: /* AD_CONFIG */
        ptconf=salva_vent(200,150,355,250);
        do{
         settextjustify(LEFT_TEXT,BOTTOM_TEXT);
         settextstyle(SMALL_FONT,HORIZ_DIR,5);
         parametros_menu(200,150,355,250,EGA_DARKGRAY,EGA_WHITE,
                         EGA_WHITE,0,ptconf);
         parametros_opcion(EGA_BLUE,EGA_WHITE,EGA_WHITE,8);
         desplaza(-3,-3,1);
         menu_enter("Tiempo de barrido");
         menu_enter("Modo de disparo");
         menu_enter("Numero de barridos");
         menu_enter("Aceptacion");

         opconf=menu();
         switch (opconf)
         {
            case 1: /* Tiempo de barrido */
                  parametros_prompt(EGA_BLUE,EGA_YELLOW,EGA_WHITE);
                  moveto(180,200);
                  sw_time=readfloat("Tiempo de barrido: ");
                  set_ad_rate(sw_time);
                  settextjustify(LEFT_TEXT,BOTTOM_TEXT);
                  break;

            case 2: /* Modo de disparo */
                  parametros_menu(230,200,330,260,EGA_GREEN,EGA_YELLOW,
                                  EGA_YELLOW,0,pttrig);
                  parametros_opcion(EGA_BLUE,EGA_WHITE,EGA_WHITE,5);
                  pttrig=salva_vent(230,200,330,260);
                  desplaza(-3,-3,1);
                  menu_enter("NO_TRIGGER");
                  menu_enter("KB_TRIGGER");
                  menu_enter("TTL_TRIGGER");
                  optrig=menu();
                  switch (optrig)
                  {
                     case 1: trigger_mode=0;
                           restaura_vent(230,200,pttrig);
                           break;

                     case 2: trigger_mode=1;
                           restaura_vent(230,200,pttrig);
                           break;

                     case 3: trigger_mode=2;
                           restaura_vent(230,200,pttrig);
                           break;
                  }
                  set_trigger_mode(trigger_mode);
                  break;

            case 3: /* Numero de barridos */
                  parametros_prompt(EGA_BLUE,EGA_YELLOW,EGA_WHITE);
                  moveto(180,200);
                  num_barrido=readfloat("Numero de barridos: ");
```

```
                settextjustify(LEFT_TEXT,BOTTOM_TEXT);
                break;

            case 4: /* Aceptacion AD_Config */
                opconf=4;
                restaura_vent(200,150,ptconf);
                break;
        }
       }while(opconf<4);
       break;

case 3: /* ALMACEN */
       ptalma=salva_vent(200,150,355,250);
       do{
        settextjustify(LEFT_TEXT,BOTTOM_TEXT);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        parametros_menu(200,150,355,250,EGA_BLUE,EGA_WHITE,
                        EGA_YELLOW,0,ptalma);
        parametros_opcion(EGA_LIGHTGRAY,EGA_WHITE,EGA_WHITE,8);
        desplaza(-3,-3,1);
        menu_enter("Modo de escritura");
        menu_enter("Modo de lectura");
        menu_enter("Serie escritura");
        menu_enter("Serie lectura");
        menu_enter("Aceptacion");

        opalma=menu();
        switch (opalma)
        {
            case 1: /* Modo de escritura */
                parametros_menu(230,190,300,235,EGA_DARKGRAY,EGA_YELLOW,
                                EGA_WHITE,0,ptesc);
                parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,5);
                ptesc=salva_vent(230,190,300,235);
                desplaza(-3,-3,1);
                menu_enter("BINARIO");
                menu_enter("ASCII");
                opesc=menu();
                switch (opesc)
                {
                    case 1: disk_mode_write=0;
                            restaura_vent(230,190,ptesc);
                            break;

                    case 2: disk_mode_write=1;
                            restaura_vent(230,190,ptesc);
                            break;

                }
                set_disk_mode_write(disk_mode_write);
                break;


            case 2: /* Modo de lectura */
                parametros_menu(230,190,300,235,EGA_DARKGRAY,EGA_YELLOW,
                                EGA_WHITE,0,ptlec);
                parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,5);
                ptlec=salva_vent(230,190,300,235);
                desplaza(-3,-3,1);
                menu_enter("BINARIO");
                menu_enter("ASCII");
                oplec=menu();
```

```
                switch (oplec)
                {
                   case 1: disk_mode_read=0;
                           restaura_vent(230,190,ptlec);
                           break;

                   case 2: disk_mode_read=1;
                           restaura_vent(230,190,ptlec);
                           break;

                }
                set_disk_mode_read(disk_mode_read);
                break;

           case 3: /* Nombre de Serie escritura */
                parametros_prompt(EGA_RED,EGA_YELLOW,EGA_WHITE);
                moveto(180,200);
                strcpy(serie_esc,readtext("Nombre Serie escritura: "));
                settextjustify(LEFT_TEXT,BOTTOM_TEXT);
                break;

           case 4: /* Nombre de Serie lectura */
                parametros_prompt(EGA_RED,EGA_YELLOW,EGA_WHITE);
                moveto(180,200);
                strcpy(serie_lec,readtext("Nombre Serie lectura: "));
                settextjustify(LEFT_TEXT,BOTTOM_TEXT);
                break;

           case 5: /* Aceptacion de Almacen */
                opalma=5;
                restaura_vent(200,150,ptalma);
                break;
        }
       }while(opalma<5);
       break;

case 4: /* FUNCION */
       ptfun=salva_vent(200,150,400,290);
       do{
        settextjustify(LEFT_TEXT,BOTTOM_TEXT);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        parametros_menu(200,150,400,290,EGA_GREEN,EGA_WHITE,
                        EGA_YELLOW,0,ptfun);
        parametros_opcion(EGA_LIGHTGRAY,EGA_WHITE,EGA_WHITE,8);
        desplaza(-3,-3,1);
        menu_enter("Adquiere en Tiempo");
        menu_enter("Adquiere en Frecuencia");
        menu_enter("Promedio en Tiempo");
        menu_enter("Promedio en Frecuencia");
        menu_enter("Despliega y Mide Serie");
        menu_enter("Autocorre. de Intervalos");
        menu_enter("Regreso Menu Principal");

        opfun=menu();
        switch (opfun)
        {
           case 1: /* Adquiere en Tiempo */
                tipo=0;
                suelta_boton();
                ptver=salva_vent(179,199,381,221);
                ventana(180,200,320,220,EGA_DARKGRAY,EGA_DARKGRAY);
                setcolor(EGA_WHITE);
```

```
            moveto(179,199);
            lineto(381,199);
            lineto(381,221);
            lineto(179,221);
            lineto(179,199);
            outtextxy(188,212,"Ver adquisicion : ");
            parametros_menu(320,200,380,220,EGA_DARKGRAY,EGA_DARKGRAY,
                                    EGA_WHITE,0,ptver);
            parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,8);

            desplaza(0,-8,0);
            menu_enter("SI");
            menu_enter("NO");
            opver=menu();
            switch (opver)
            {
               case 1: ver=1;
                       restaura_vent(179,199,ptver);
                       break;

               case 2: ver=0;
                       restaura_vent(179,199,ptver);
                       break;

            }

            for(i=0; i<num_barrido;i++)
            {
               adq_16_can();
               esc=write_disk(serie_esc);
               if (esc==1)
               {
                esc=0;
                break;
               }
               if (ver)
               {
                strcpy(serie_actual,serie_esc);
                despliega16(0);
               }
               archivo_sec(serie_esc);
            }
            if (ver)
            {
               presentacion();
               titulo();
               par_config();
            }
            break;


         case 2: /* Adquiere en Frecuencia */
            tipo=1;
            suelta_boton();
            ptver=salva_vent(179,199,381,221);
            ventana(180,200,320,220,EGA_DARKGRAY,EGA_DARKGRAY);
            setcolor(EGA_WHITE);
            moveto(179,199);
            lineto(381,199);
            lineto(381,221);
            lineto(179,221);
            lineto(179,199);
```

```
outtextxy(188,212,"Ver adquisicion : ");
parametros_menu(320,200,380,220,EGA_DARKGRAY,EGA_DARKGRAY,
                        EGA_WHITE,0,pVer);
parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,8);
desplaza(0,-8,0);
menu_enter("SI");
menu_enter("NO");
opver=menu();
switch (opver)
{
   case 1: ver=1;
         restaura_vent(179,199,ptver);
         break;

   case 2: ver=0;
         restaura_vent(179,199,ptver);
         break;

}

fft_init();
for(l=0; l<num_barrido;l++)
{
   adq_16_can();
   f=adfft;
   for(j=0;j<16;j++)
   {
    FFT(8,1,j+1);
    f+=j;
    for (k=0;k<128;k++)
    {
       *(f)=(xf[k].re)*(xf[k].re)+(xf[k].im)*(xf[k].im);
       f+=16;
       *(f)=(xf[k].re)*(xf[k].re)+(xf[k].im)*(xf[k].im);
       f+=16;
    }
    f=adfft;
   }
   mayor=0;
   for(i=0;i<4096;i++)
    if (*(adfft+i)>mayor) mayor=*(adfft+i);
   mayor=mayor/2048;
   if (mayor)
   for(i=0;i<4096;i++)
    *(adbuff+i)=(int) ceil(*(adfft+i)/mayor);
   else
   for (i=0;i<4096;i++)
    *(adbuff+i)= (int) ceil(*(adfft+i));

   esc=write_disk(serie_esc);
   if (esc==1)
   {
    esc=0;
    break;
   }
   if (ver)
   {
    strcpy(serie_actual,serie_esc);
    despliega16(0);
   }
   archivo_sec(serie_esc);
}
```

```
                fft_finit();

                if (ver)
                {
                   presentacion();
                   titulo();
                   par_config();
                }
                break;

          case 3: /* Promedio en tiempo */
                tipo=0;
                ptprot=salva_vent(230,230,435,305);
                do{
                   parametros_menu(230,230,435,305,EGA_DARKGRAY,EGA_WHITE,
                                   EGA_WHITE,0,ptver);
                   parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,8);
                   menu_enter("Promediar Serie de A/D");
                   menu_enter("Promediar Serie de Disco");
                   menu_enter("Regreso Menu Funcion");
                   opprot=menu();
                   switch (opprot)
                   {
                    case 1: suelta_boton();
                          ptver=salva_vent(269,269,471,291);
                          ventana(270,270,410,290,EGA_CYAN,EGA_CYAN);
                          setcolor(EGA_LIGHTCYAN);
                          moveto(269,269);
                          lineto(471,269);
                          lineto(471,291);
                          lineto(269,291);
                          lineto(269,269);
                          setcolor(EGA_WHITE);
                          outtextxy(278,282,"Ver Promedio : ");
                          parametros_menu(410,270,470,290,EGA_CYAN,EGA_CYAN,
                                          EGA_WHITE,0,ptver);
                          parametros_opcion(EGA_BLUE,EGA_LIGHTBLUE,
                                            EGA_WHITE,8);
                          desplaza(0,-8,0);
                          menu_enter("SI");
                          menu_enter("NO");
                          opver=menu();
                          switch (opver)
                          {
                             case 1: ver=1;
                                   restaura_vent(269,269,ptver);
                                   break;

                             case 2: ver=0;
                                   restaura_vent(269,269,ptver);
                                   break;

                          }

                          prom_init();
                          for(i=0; i<4096; i++)
                             *(prombuff+i) = 0;

                          for(i=0; i<num_barrido;i++)
                          {
                            adq_16_can();
                            for (j=0; j<4096; j++)
```

```
            *(prombuff+j) += *(adbuff+j);
         }
         for(i=0; i<4096; i++)
            *(adbuff+i) = ceil(*(prombuff+i)/num_barrido);
         prom_finit();
         esc=write_disk(serie_esc);
         if(esc==1)
         {
            esc=0;
            break;
         }
         if (ver)
         {
            strcpy(serie_actual,serie_esc);
            despliega16(0);
         }

         break;


      case 2: suelta_boton();
         ptver=salva_vent(269,269,471,291);
         ventana(270,270,410,290,EGA_CYAN,EGA_CYAN);
         setcolor(EGA_LIGHTCYAN);
         moveto(269,269);
         lineto(471,269);
         lineto(471,291);
         lineto(269,291);
         lineto(269,269);
         setcolor(EGA_WHITE);
         outtextxy(278,282,"Ver Promedio : ");
         parametros_menu(410,270,470,290,EGA_CYAN,EGA_CYAN,
                         EGA_WHITE,0,ptver);
         parametros_opcion(EGA_BLUE,EGA_LIGHTBLUE,
                           EGA_WHITE,8);
         desplaza(0,-8,0);
         menu_enter("SI");
         menu_enter("NO");
         opver=menu();
         switch (opver)
         {
            case 1: ver=1;
                  restaura_vent(269,269,ptver);
                  break;

            case 2: ver=0;
                  restaura_vent(269,269,ptver);
                  break;

         }

         prom_init();
         for(i=0; i<4096; i++)
            *(prombuff+i) = 0;

         for(i=0; i<num_barrido;i++)
         {
           lee=read_disk(serie_lec);
           if (lee==1) break;

           for (j=0; j<4096; j++)
              *(prombuff+j) += *(adbuff+j);
```

```
                        archivo_sec(serie_lec);
                    }
                    if (lee==1)
                    {
                        lee=0;
                        break;
                    }
                    for(i=0; i<4096; i++)
                        *(adbuff+i) = ceil(*(prombuff+i)/num_barrido);
                    prom_finit();
                    esc=write_disk(serie_esc);
                    if(esc==1)
                    {
                        esc=0;
                        break;
                    }

                    if (ver)
                    {
                        strcpy(serie_actual,serie_esc);
                        despliega16(0);
                    }

                    break;

                case 3: opprot=3;
                                restaura_vent(230,230,ptprot);
                    if (ver)
                    {
                        presentacion();
                        titulo();
                        par_config();
                    }
                    break;

                }
            }while(opprot<3);

            break;

        case 4: /* Promedio en Frecuencia */
            tipo=1;
            ptprot=salva_vent(230,230,435,305);
            do{
                parametros_menu(230,230,435,305,EGA_DARKGRAY,EGA_WHITE,
                                EGA_WHITE,0,ptver);
                parametros_opcion(EGA_RED,EGA_WHITE,EGA_WHITE,8);
                menu_enter("Promediar Serie de A/D");
                menu_enter("Promediar Serie de Disco");
                menu_enter("Regreso Menu Funcion");
                opprot=menu();
                switch (opprot)
                {
                 case 1: suelta_boton();
                        ptver=salva_vent(269,269,471,291);
                        ventana(270,270,410,290,EGA_CYAN,EGA_CYAN);
                        setcolor(EGA_LIGHTCYAN);
                        moveto(269,269);
                        lineto(471,269);
                        lineto(471,291);
                        lineto(269,291);
                        lineto(269,269);
```

```
setcolor(EGA_WHITE);
outtextxy(278,282,"Ver Promedio : ");
parametros_menu(410,270,470,290,EGA_CYAN,EGA_CYAN,
                        EGA_WHITE,0,ptver);
parametros_opcion(EGA_BLUE,EGA_LIGHTBLUE,
                                EGA_WHITE,8);
desplaza(0,-8,0);
menu_enter("SI");
menu_enter("NO");
opver=menu();
switch (opver)
{
   case 1: ver=1;
         restaura_vent(269,269,ptver);
         break;

   case 2: ver=0;
         restaura_vent(269,269,ptver);
         break;

}

prom_init();
for(I=0; i<4096; i++)
   *(prombuff+i) = 0;

fft_init();
for(l=0; l<num_barrido;l++)
{
   adq_16_can();
   f=adfft;
   for(j=0;j<16;j++)
   {
     FFT(8,1,j+1);
     f+=j;
     for (k=0;k<128;k++)
     {
        *(f)=(xf[k].re)*(xf[k].re)+
             (xf[k].im)*(xf[k].im);
        f+=16;
        *(f)=(xf[k].re)*(xf[k].re)+
             (xf[k].im)*(xf[k].im);
        f+=16;
     }
     f=adfft;
   }
   mayor=0;
   for(i=0;i<4096;i++)
      if (*(adfft+i)>mayor) mayor=*(adfft+i);
   mayor=mayor/2048;
   if (mayor)
   for(i=0;i<4096;i++)
     *(adbuff+i)=(int) ceil(*(adfft+i)/mayor);
   else
   for(i=0;i<4096;i++)
      *(adbuff+i)=(int)ceil(*(adfft+i));

   for (j=0; j<4096; j++)
     *(prombuff+j) += *(adbuff+j);
}
fft_finit();
```

```
        for(i=0; i<4096; i++)
           *(adbuff+i) = ceil(*(prombuff+i)/num_barrido);
        prom_finit();

        esc=write_disk(serie_osc);
        if(esc==1)
        {
           esc=0;
           break;
        }
        if (ver)
        {
           strcpy(serie_actual,serie_osc);
           despliega16(0);
        }

        break;


     case 2: suelta_boton();
        ptver=salva_vent(269,269,471,291);
        ventana(270,270,410,290,EGA_CYAN,EGA_CYAN);
        setcolor(EGA_LIGHTCYAN);
        moveto(269,269);
        lineto(471,269);
        lineto(471,291);
        lineto(269,291);
        lineto(269,269);
        setcolor(EGA_WHITE);
        outtextxy(278,282,"Ver Promedio : ");
        parametros_menu(410,270,470,290,EGA_CYAN,EGA_CYAN,
                        EGA_WHITE,0,ptver);
        parametros_opcion(EGA_BLUE,EGA_LIGHTBLUE,
                          EGA_WHITE,8);
        desplaza(0,-8,0);
        menu_enter("SI");
        menu_enter("NO");
        opver=menu();
        switch (opver)
        {
           case 1: ver=1;
                 restaura_vent(269,269,ptver);
                 break;

           case 2: ver=0;
                 restaura_vent(269,269,ptver);
                 break;

        }

        prom_init();
        for(i=0; i<4096; i++)
           *(prombuff+i) = 0;

        for(i=0; i<num_barrido;i++)
        {
          lee=read_disk(serie_lec);
          if (lee==1) break;

          for (j=0; j<4096; j++)
             *(prombuff+j) += *(adbuff+j);
          archivo_sec(serie_lec);
```

```
                    }
                    if (lee==1)
                    {
                       lee=0;
                       break;
                    }
                    for(i=0; i<4096; i++)
                       *(adbuff+i) = cell(*(prombuff+i)/num_barrido);
                    prom_finit();
                    esc=write_disk(serie_esc);
                    if(esc==1)
                    {
                       esc=0;
                       break;
                    }

                    if (ver)
                    {
                       strcpy(serie_actual,serie_esc);
                       despliega16(0);
                    }

                    break;

              case 3: opprot=3;
                                restaura_vent(230,230,ptprot);
                    if (ver)
                    {
                       presentacion();
                       titulo();
                       par_config();
                    }
                    break;

                 }
              }while(opprot<3);
              break;

        case 5: /* Despliega y Mide Serie */
              for(i=0; i<num_barrido;i++)
              {
                 lee=read_disk(serie_lec);
                 if (lee==1)
                 {
                  lee=0;
                  break;
                 }
                 strcpy(serie_actual,serie_lec);
                 despliega16(1);
                 archivo_sec(serie_lec);
              }
              presentacion();
              titulo();
              par_config();
              break;

        case 6: /* Autocorrelacion de Intervalos */
              set_ad_rate(16.0);

              parametros_prompt(EGA_DARKGRAY,EGA_WHITE,EGA_WHITE);
              moveto(180,310);
              canal=readfloat("Canal para autocorrelacion: ");
```

```
                    settextjustify(LEFT_TEXT,BOTTOM_TEXT);

                    canal_umbral(canal);
                    cuenta_intervalos(canal,512);
                    auto_cor_int(canal,512,256);
                    break;

              case 7: /* Regreso al Menu Principal */
                    opfun=7;
                    restaura_vent(200,150,ptfun);
                    break;
             }
            }while(opfun<7);
            break;

     case 5: /* SALIR MENU PRINCIPAL */
            presentacion();
            setcolor(EGA_YELLOW);
            settextstyle(TRIPLEX_FONT,HORIZ_DIR,9);
            settextjustify(CENTER_TEXT,CENTER_TEXT);
            setusercharsize(1,1,1,1);
            outtextxy(getmaxx()/2,getmaxy()/2,"Fin de Sesion");
            opmp=5;
            break;
   }
}while(opmp<5);
sleep(1);
sd_finit();
}


/*********************************************************************/

/* Encabezado de funciones del convertidor PCLAB-812 */
/* Unidad  ADFUN.H */

/* Definicion de constantes */

static const NO_TRIGGER = 0;          /* no hay trigger */
static const KB_TRIGGER = 1;          /* el trigger se inicia por teclado */
static const TTL_TRIGGER= 2;          /* el trigger es un pulso externo de
                                         sincronia*/
static const BIN = 0;                 /* tipo binario de almacenaje y lectura */
static const ASCII = 1;               /* tipo ASCII de almacenaje y lectura */

/* Definicion de variables globales */

extern  int       canal;              /* inicializacion de canales */
extern  int       base;               /* direccion base del convertidor */
extern  float     sw_time;            /* tiempo de barrido */
extern  int       num_barrido;        /* numero de barridos a adquirir */
extern  int       rate;               /* frecuencia de muestreo o intervalo de
                                         muestreo */
extern  int       dato_conv;
extern  int       fin;
extern  int far *adbuff;              // apuntador entero a area de memoria
                                      // donde van a guradarse 256 ptos X 16
                                         canales
extern  int far *prombuff;            // apuntador entero a area de memoria
                                      // donde van a guradarse 256 ptos X 16
canales                               // promediados
extern  float far*adfft;              // apuntador real a area de memoria
```

```
                                        // donde van a guardarse 256 ptos X16 canales
                                        // series FFT
extern  int trigger_mode;               // tipo de trigger
extern  int     disk_mode_read;         // tipo de archivo a leer, binario o ASCII
extern  int     disk_mode_write;        // tipo de archivo a escribir, binario o
                                        // ASCII
extern  long    tipo;                   // tipo de serie 0=tiempo, 1=frecuencia
extern  char    serie_esc[80];          // nombre de la serie de escritura
extern  char    serie_lec[80];          // nombre de la serie de lectura
extern  char    serie_actual[80];       // nombre auxiliar de la serie

/* Definicion de funciones */

extern void ad_init(void);
extern void ad_finit(void);
extern void prom_init(void);
extern void prom_finit(void);
extern void fft_Init(void);
extern void fft_finit(void);
extern void set_ad_rate(float sw_time);
extern int convierte(int canal);
extern int convierte_asm(int canal);
extern void adq_16_can(void);
extern void set_trigger_mode(int mode);
extern void set_disk_mode_read(int mode);
extern void set_disk_mode_write(int mode);
extern int read_disk(char const *namefile);
extern int write_disk(char const  *namefile);
extern void archivo_sec(char * nombre);


/* Funciones Principales de la Unidad ADFUN.C */
#pragma inline
#include <dos.h>
#include <alloc.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <graphics.h>
#include "adfun.h"
#include "primifun.h"



//*********************************
//* Funcion que inicializa los    **
//* contadores para la velocidad **
//* de muestreo de la tarjeta     **
//*********************************
void set_ad_rate(float sw_time)
{
      float       div,div1,div2;
      int         error;
      int         rate;                     /*frecuencia de muestreo*/
      unsigned char c1h,c11,c2h,c21;
      unsigned w1,w2;

      rate=(256/sw_time)*16;
      div=2E6/rate;
      div1=div/0xFFFF;
      if(div1>1)
      {
```

```
         div1+=1;
         div2=div/div1;
      }
      else
      {
         div1=2;
         div2=div/div1;
      }
      w1=div1;
      w2=div2;

      c1h=(w1 & 0xFF00)>>8;
      c1l=(w1 & 0x00FF);

      c2h=(w2 & 0xFF00)>>8;
      c2l=(w2 & 0x00FF);

      outportb(base+11,0x06);        // Enable pacer trigger only
      inportb(base+4);               // Dummy read, clear ready flag
      outportb(base+3,0xB4);         // Control cntr 2
      outportb(base+3,0x74);         // Control cntr 1

      outportb(base+2,c2l);
      outportb(base+2,c2h);
      outportb(base+1,c1l);
      outportb(base+1,c1h);
}



//*****************************
//*** funcion que hace uso ****
//*** del convertidor      ****
//*****************************
int convierte(int canal)
{
  int  fin;

  outportb(base+10,canal);
  do
  {
     fin=inportb(base+5) & 16;
  }while (fin!=0);

  dato_conv=(inportb(base+5) << 8) + (inportb(base+4) );
  dato_conv-=0x800;
  return(dato_conv);
}
```

```
//*****************************
//** funcion que adquiere 16 ***
//** canales de 256 ptos c/u ***
//*****************************
void adq_16_can(void)
{
   unsigned int   seg;
   unsigned int   offs;

   switch (trigger_mode)
   {
      case 0 : break;

      case 1 : do{
               }while (!kbhit());
               getch();
               break;

      case 2 : do
               {
                  sincronia=inportb(base+6) & 0x01;        // leeo el bit 0,
                }while(sincronia);                         // pulso TTL
               do
               {
                  sincronia=inportb(base+6) & 0x01;
               }while(!sincronia)
               break;
   }


   seg  = FP_SEG((int far *)adbuff);
   offs = FP_OFF((int far *)adbuff);

   asm   PUSH DS
   asm   PUSH DI
   asm   MOV  AX,seg          /* carga el segmento de adbuff en  DS */
   asm   MOV  DS,AX
   asm   MOV  DI,offs         /* carga el offset de adbuff en DI */
   asm   MOV  CX,0x1000       /* inicio contador con 4096 conversiones */
loop256:
   asm   MOV  BL,0            /* primer canal a convertir (cont. de canales) */
loop16:
   asm   MOV  DX,0x220        /* cargo la dir. base del convertidor */
   asm   ADD  DX,10           /* sumo +10 a la dir. base = set chanel */
   asm   MOV  AL,BL
   asm   OUT  DX,AL           /* selecciono canal en AL */

   asm   MOV  AL,0x0F         /* cargo un 0F an AL para hacer un retardo */
wait:
   asm   DEC  AL              /* y dar tiempo de que el canal sea escrito  */
   asm   JNZ  wait            /* en el registro correspondiente */

   asm   MOV  DX,0x220        /* cargo dir. base del convertidor */
   asm   ADD  DX,5            /* leer parte alta (bit 5 DRDY) */
wait5:
   asm   IN   AL,DX
   asm   CMP  AL,0x0F         /* pruebo si el bit 5 esta en 1= no listo */
   asm   JG   wait5           /* brinca mientras no este listo */

   asm   MOV  AH,AL           /* cargo parte alta del dato en AH */
   asm   MOV  DX,0x220        /* dir. base convertidor */
   asm   ADD  DX,4            /* dir. parte baja del dato */
```

B.18

```
   asm   IN    AL,DX             /* cargo parte baja del dato en AL */

   asm   SUB   AX,0x800          /* resto la basal */
   asm   MOV   DS:[DI],AX        /* guardo el dato completo en DS:DI (dir. de */
                                 /* adbuff) */
   asm   INC   DI                /* DI apunta a la siguiente palabra */
   asm   INC   DI

   asm   DEC   CX                /* decremento contador de conversiones */
   asm   JZ    fin               /* terminan las conversiones ? */

   asm   INC   BL                /* incremento en 1 el num. de canal */
   asm   CMP   BL,0x10           /* compara el num. de canal con 15 (ultimo canal)*/
   asm   JNZ   loop16            /* repite hasta que llega al canal 15 (0-15) */

   asm   JMP   loop256           /* otra linea de 16 canales */

fin:
   asm   POP DI
   asm   POP DS

}


/*************************************************************************/

/* Definiciones de Primitivos Graficos */
/* Unidad PRIMIFUN.H */

#include <dos.h>

#define call_mouse int86(0x33,&inreg,&outreg)

/* definicion de constantes */

#define   FALSE     0
#define   TRUE      1
#define   OFF       0
#define   ON        1
#define   SI        1
#define   NO        0
#define   ButtonL   0
#define   ButtonR   1
#define   ButtonM   2
#define   ELENA_BLACK  EGA_DARKGRAY

/* definicion de variables globales */

extern int      promt_fondo;
extern int      promt_marco;
extern int      promt_chr;

extern int      menu_x;
extern int      menu_y;
extern int      menu_tx;
extern int      menu_ty;
extern int      menu_fondo;
extern int      menu_marco;
extern int      menu_chr;
extern int      menu_issave;
extern int far *menu_iptr;
```

```
extern int  vent_x;
extern int  vent_y;
extern int  vent_tx;
extern int  vent_ty;
extern int  vent_fondo;
extern int  vent_marco;
extern int  vent_issave;
extern int far *vent_iptr;

extern int      opcion_fondo;
extern int      opcion_marco;
extern int      opcion_chr;
extern int      opcion_ptr;
extern int      opcion_distancia;
extern int      opcion_despx;       /*desplazamiento en x*/
extern int      opcion_despy;       /*desplazamiento en y*/
extern int      opcion_sentido;     /*sentido  del  desplazamiento  0=horz,*/
                                    /* 1=vert.*/
extern int      ver;
extern char     str[20];
extern int      gdriver;
extern int      gmode;
extern int      Mview;

/* definicion de tipos */

typedef struct
{
      int button_status,
          button_count,
          xaxis,yaxis;
} Mstatus;

typedef struct
{
      int present,
          buttons;
} Mresult;


typedef struct {
             int x1;
             int y1;
             int x2;
             int y2;
             char msg[25];
             }opos;

extern opos posvent[25];
extern Mstatus status;
extern union REGS inreg,outreg;

/* definicion de funciones */

extern float readfloat(char *texto);
extern char far * readtext(char far *texto);
extern int far * salva_vent(int x1,int y1,int x2,int y2);
extern void restaura_vent(int x,int y,int far *ptres);
extern void parametros_prompt(int fondo,int marco,int chr);
extern void parametros_menu(int x,int y,int tx,int ty,
                            int fondo,int marco,int chr,int save,int far *ptmenu);
extern void parametros_opcion(int fondo,int marco,int chr,int distancia);
```

```
extern void desplaza(int x,int y,int sentido);
extern void menu_enter(char  *texto);
extern int menu(void);
extern void ventana(int x,int y,int tx,int ty,int fondo,int marco);
extern int  entra_ventana(int x1,int y1,int x2, int y2);
extern int  sal_ventana(int x1,int y1,int x2,int y2);
extern void detecta_boton(void);
extern void suelta_boton(void);
extern void Mshow(int showstat);
extern Mstatus Mpos();
extern Mstatus Mpressed(int button);
extern Mresult  *Mreset();
extern void plot(int x,int y,int color);
extern void presentacion(void);
extern void titulo(void);
extern void par_config(void);


/* Funciones de Primitivos Graficos */
/* Funciones Principales de la Unidad PRIMIFUN.C */

#pragma inline
#include <graphics.h>
#include <alloc.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <dos.h>
#include "primifun.h"
#include "adfun.h"


/*********************************************************/
/*** funcion que pone los parametros de una ventana  *****/
/*** de menu cualquiera, numero de opciones, color del ***/
/*** fondo de la ventana, color del texto, color del *****/
/*** fondo de la opacion seleccionada, color del texto ***/
/*** de la opcion seleccionada, color del marco de ambas */
/*** ventanas; haciendo uso del mouse para seleccionar  **/
/*********************************************************/

int menu(void)
{
      int boton=0;
      int i;

      suelta_boton();
      if(menu_issave) menu_iptr=salva_vent(menu_x,menu_y,menu_tx,menu_ty);
      setfillstyle(SOLID_FILL,menu_fondo);
      bar(menu_x,menu_y,menu_tx,menu_ty);
      setcolor(menu_marco);
      rectangle(menu_x,menu_y,menu_tx,menu_ty);

      setcolor(menu_chr);

      for(i=0;i<opcion_ptr;i++)
         outtextxy(posvent[i].x1+2,posvent[i].y2-4,posvent[i].msg);

      Mshow(TRUE);
```

```
      i=0;
      do
      {
            if (entra_ventana(posvent[i].x1,posvent[i].y1,
                posvent[i].x2,posvent[i].y2))
            {
              Mshow(FALSE);
              setfillstyle(SOLID_FILL,opcion_fondo);
              bar(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);
              setcolor(opcion_marco);
              rectangle(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);

              setcolor(opcion_chr);
              moveto(posvent[i].x1+2,posvent[i].y2-4);
              outtext(posvent[i].msg);

              Mshow(TRUE);
              boton=sal_ventana(posvent[i].x1,posvent[i].y1,
                                posvent[i].x2,posvent[i].y2);
              if(boton)boton=i+1;
              Mshow(FALSE);
              setfillstyle(SOLID_FILL,menu_fondo);
              bar(posvent[i].x1,posvent[i].y1,posvent[i].x2,posvent[i].y2);

              setcolor(menu_chr);
              moveto(posvent[i].x1+2,posvent[i].y2-4);
              outtext(posvent[i].msg);

              Mshow(TRUE);
            };
            i++;
            if(i==opcion_ptr)i=0;
      }while(!boton);
      Mshow(FALSE);
      if(menu_isave) restaura_vent(menu_x,menu_y,menu_iptr);
      return(boton);
}


/*****************************************************************/

/* Definicion de Graficos de Alto Nivel */
/* Unidad GRAFFUN.H */

/*  definicion de constantes */

#define AD_RANGO  2
#define     FFT_MAX     256   /* numero maximo de puntos por transformada */

/* definicion de tipos */

typedef  struct               /* se define el tipo complex  */
     {
        double   re;
        double   im;
     } complex;

typedef complex fastdat[FFT_MAX+10];       /* se define el tipo fastdat como un*/

/* definicion de variables globales */
```

```
extern int   dat[256];          /* arreglo de datos a graficar */
extern float dat1[256];         /* arreglo de datos auxiliar, para calculo de
                                   integral */
extern float x[256],y[256];     /* arreglo para canales a correlacionar */
extern float cor[256];          /* arreglo de correlacion */
extern float autocor[256];      /*  arreglo  de  resultado  auto-correlacion*/
                                /* intervalo*/
extern float intervalo[512];    /* arreglo para el calculo de histograma */
extern float cursor_magx;       /* factor de ajuste en x */
extern float cursor_magy;       /* factor de ajuste en y */
extern int   cursor_scrder;     /* scroll derecha */
extern int   cursor_scrizq;     /* scroll izquierda */
extern float mayor;             /* dato mayor de la correlacion */
extern int   umbral;            /* umbral calculo correlacion intervalo */
extern fastdat xf;              /* arreglo de numeros complejos calcual FFT */

/* definicion de funciones */

extern void cursor(int far *dat,int x,int y,int num_canal);
extern void despliega16(int seleccion);
extern void despliega_1(int num_canal);
extern void correlacion(int canal1,int canal2);
extern void canal_umbral(int canal);
extern void cuenta_intervalos(int canal,int num_intervalo);
extern void auto_cor_int(int canal,int num_intervalo,int num_correlacion);
extern void dvd(complex a,complex b,complex *r);/* aritmetica de complejos*/
extern void mul(complex a,complex b,complex *r);
extern void add(complex a,complex b,complex *r);
extern void FFT(int m,int isign,int canal);
extern void despliegaFFT1(int canal);
extern void cursor_fft(int canal);


/* Libreria de Graficos de Alto Nivel */
/* Funciones Principales de Unidad GRAFFUN.C */

#include <graphics.h>
#include <alloc.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include "primifun.h"
#include "adfun.h"
#include "graffun.h"


/*******************************************************/
/***** funcion que despliega un cursor para medir    **/
/***** las caracteristicas de una senal              **/
/*******************************************************/

void cursor(int far *dat,int x,int y,int num_canal)
{

   int      op1;
   int      op2;
   int      i,k;
   int far *ptcursor;
   int far *ptmedir;
   int far *ptintegral;
```

```
int far *ptmenucur;
int far *ptmenuint;
int     entrada,salida;
int     poshoriz;
float   magy,magx;
int     point_y,point_x;
int     ant_x,ant_y;
char    buffer[80];
float   mayor;
int     menor;
float   datmax,datmin;
int     cuenta_boton;
int     indice[2];
int     indice1,indice2;
float   dv;
float   integral;


cursor_magx=1;
cursor_magy=1;
cursor_scrder=0;
cuenta_boton=0;
k=0;


ptcursor=salva_vent(x,y,556,328);
ventana(x,y,556,328,EGA_DARKGRAY,EGA_YELLOW);

settextstyle(SMALL_FONT,HORIZ_DIR,5);
parametros_menu(446,101,555,327,EGA_RED,EGA_RED,EGA_YELLOW,0,ptmenucur);
parametros_opcion(EGA_BLUE,EGA_LIGHTGRAY,EGA_WHITE,10);
desplaza(20,5,1);
menu_enter("V +");
menu_enter("V -");
menu_enter("T +");
menu_enter("T -");
menu_enter("Scr x ->");
menu_enter("Scr x <-");
menu_enter("Medir");
menu_enter("Integral");
menu_enter("Reset");
menu_enter("Salir");

line(150,150,150,278);
line(150,214,406,214);
moveto(150,214-dat[0]/32);  /* ojo */
for (i=0;i<256;i++)
   lineto(i+150,214-dat[i]/32);

sprintf(buffer,"%2.1f",cursor_magy*AD_RANGO);
moveto(120,155);
outtext(buffer);

sprintf(buffer,"-%2.1f",cursor_magy*AD_RANGO);
moveto(113,275);
outtext(buffer);

sprintf(buffer,"%1.3f",cursor_magx*sw_time);
moveto(408,214);
outtext(buffer);

if (num_canal > 0 )
```

```
   {
      sprintf(buffer,"CANAL  %2d",num_canal);
      moveto(160,300);
      outtext(buffer);
   }

do
{
   setfillstyle(SOLID_FILL,EGA_BLUE);
   bar(125,110,400,130);
   setcolor(EGA_LIGHTBLUE);
   rectangle(125,110,400,130);

   settextstyle(SMALL_FONT,HORIZ_DIR,5);
   settextjustify(LEFT_TEXT,BOTTOM_TEXT);
   setcolor(EGA_WHITE);
   moveto(130,122);
    sprintf(buffer,"Fac.tiempo    =    %2.1f        Fac.voltaje    =
          %2.1f",cursor_magx,cursor_magy);
   outtext(buffer);

   settextstyle(SMALL_FONT,HORIZ_DIR,5);
   parametros_menu(446,101,555,327,EGA_RED,EGA_RED,EGA_YELLOW,0,ptmenucur);
   parametros_opcion(EGA_BLUE,EGA_LIGHTGRAY,EGA_WHITE,10);
   desplaza(20,5,1);
   menu_enter("V +");
   menu_enter("V -");
   menu_enter("T +");
   menu_enter("T -");
   menu_enter("Scr x ->");
   menu_enter("Scr x <-");
   menu_enter("Medir");
   menu_enter("Integral");
   menu_enter("Reset");
   menu_enter("Salir");

   opl=menu();
   switch (opl)
       {
        case 1: /* incremento voltaje */
              magy=cursor_magy;
              cursor_magy+=0.2;

              if(cursor_scrder<1)
                 moveto(150,214-(int)((dat[0]/32)*magy)); /* ojo */
              else
                 moveto(150+cursor_scrder,214-(int)((dat[0]/32)*magy));
              moveto(150+cursor_scrder,214-(int)((dat[0]/32)*magy));
              setcolor(vent_fondo);

              sprintf(buffer,"%2.1f",AD_RANGO/magy);
              moveto(120,155);
              outtext(buffer);

              sprintf(buffer,"-%2.1f",AD_RANGO/magy);
              moveto(113,275);
              outtext(buffer);

              for (i=0;i<256;i++)
              {
                   point_y=214-(int)((dat[i]/32)*magy);
                   point_x=(int)(i*cursor_magx)+150+cursor_scrder;
```

```
            if(point_y<150) point_y=150;
            if(point_y>278) point_y=278;
            if(point_x<150) point_x=150;
            if(point_x>406) point_x=406;
            lineto(point_x,point_y);
        }

        setcolor(menu_chr);
        line(150,150,150,278);
        line(150,214,406,214);

        sprintf(buffer,"%2.1f",AD_RANGO/cursor_magy);
        moveto(120,155);
        outtext(buffer);

        sprintf(buffer,"-%2.1f",AD_RANGO/cursor_magy);
        moveto(113,275);
        outtext(buffer);

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy)); /* ojo */
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        for (i=0;i<256;i++)
        {
            point_y=214-(int)((dat[i]/32)*cursor_magy);
            point_x=(int)(i*cursor_magx)+150+cursor_scrder;
            if(point_y<150) point_y=150;
            if(point_y>278) point_y=278;
            if(point_x<150) point_x=150;
            if(point_x>406)
            {
             setcolor(vent_fondo);
             point_x=406;
            }
            lineto(point_x,point_y);
        }
        break;

case 2: /* decremento voltaje */
        magy=cursor_magy;
        cursor_magy-=0.2;

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*magy)); /* ojo */
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*magy));
        setcolor(vent_fondo);

        sprintf(buffer,"%2.1f",AD_RANGO/magy);
        moveto(120,155);
        outtext(buffer);

        sprintf(buffer,"-%2.1f",AD_RANGO/magy);
        moveto(113,275);
        outtext(buffer);

        for (i=0;i<256;i++)
        {
            point_y=214-(int)((dat[i]/32)*magy);
            point_x=(int)(i*cursor_magx)+150+cursor_scrder;
            if(point_y<150) point_y=150;
```

```
            if(point_y>278) point_y=278;
            if(point_x<150) point_x=150;
            if(point_x>406) point_x=406;
            lineto(point_x,point_y);
        }

        setcolor(menu_chr);
        line(150,150,150,278);
        line(150,214,406,214);

        sprintf(buffer,"%2.1f",AD_RANGO/cursor_magy);
        moveto(120,155);
        outtext(buffer);

        sprintf(buffer,"-%2.1f",AD_RANGO/cursor_magy);
        moveto(113,275);
        outtext(buffer);

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        for (i=0;i<256;i++)
        {
            point_y=214-(int)((dat[i]/32)*cursor_magy);
            point_x=(int)(i*cursor_magx)+150+cursor_scrder;
            if(point_y<150) point_y=150;
            if(point_y>278) point_y=278;
            if(point_x<150) point_x=150;
            if(point_x>406)
            {
             setcolor(vent_fondo);
             point_x=406;
            }
            lineto(point_x,point_y);
        }
        break;

    case 3: /* incremento tiempo */
        magx=cursor_magx;
        cursor_magx+=0.2;

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        setcolor(vent_fondo);

        sprintf(buffer,"%1.3f",sw_time/magx);
        moveto(408,214);
        outtext(buffer);

        for (i=0;i<256;i++)
        {
            point_x=(int)(i*magx)+150+cursor_scrder;
            point_y=214-(int)((dat[i]/32)*cursor_magy);
            if(point_x<150) point_x=150;
            if(point_x>406) point_x=406;
            if(point_y<150) point_y=150;
            if(point_y>278) point_y=278;
            lineto(point_x,point_y);
        }
```

```
        setcolor(menu_chr);
        line(150,150,150,278);
        line(150,214,406,214);

            sprintf(buffer,"%1.3f",sw_time/cursor_magx);
        moveto(408,214);
        outtext(buffer);

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        for (i=0;i<256;i++)
        {
             point_x=(int)(i*cursor_magx)+150+cursor_scrder;
             point_y=214-(int)((dat[i]/32)*cursor_magy);
             if(point_x<150) point_x=150;
             if(point_x>406)
             {
              setcolor(vent_fondo);
              point_x=406;
             }
             if(point_y<150) point_y=150;
             if(point_y>278) point_y=278;
             lineto(point_x,point_y);
        }
        break;

case 4: /* decremento tiempo */
        magx=cursor_magx;
        cursor_magx-=0.2;

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        setcolor(vent_fondo);

        sprintf(buffer,"%1.3f",sw_time/magx);
        moveto(408,214);
        outtext(buffer);

        for (i=0;i<256;i++)
        {
             point_x=(int)(i*magx)+150+cursor_scrder;
             point_y=214-(int)((dat[i]/32)*cursor_magy);
             if(point_x<150) point_x=150;
             if(point_x>406) point_x=406;
             if(point_y<150) point_y=150;
             if(point_y>278) point_y=278;
             lineto(point_x,point_y);
        }

        setcolor(menu_chr);
        line(150,150,150,278);
        line(150,214,406,214);

        sprintf(buffer,"%1.3f",sw_time/cursor_magx);
        moveto(408,214);
        outtext(buffer);

        if(cursor_scrder<1)
```

```
            moveto(150,214-(int)((dat[0]/32)*cursor_magy));
          else
            moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
          for (i=0;i<256;i++)
          {
               point_x=(int)(i*cursor_magx)+150+cursor_scrder;
               point_y=214-(int)((dat[i]/32)*cursor_magy);
               if(point_x<150) point_x=150;
               if(point_x>406)
               {
                setcolor(vent_fondo);
                point_x=406;
               }
               if(point_y<150) point_y=150;
               if(point_y>278) point_y=278;
               lineto(point_x,point_y);
          }
          break;

     case 5: /* scroll derecha */
          k=cursor_scrder;
          cursor_scrder+=10;

          if(k<1)
             moveto(150,214-(int)((dat[0]/32)*cursor_magy));
          else
             moveto(150+k,214-(int)((dat[0]/32)*cursor_magy));
          setcolor(vent_fondo);
          for (i=0;i<256;i++)
          {
               point_x=(int)(i*cursor_magx)+150+k;
               point_y=214-(int)((dat[i]/32)*cursor_magy);
               if(point_x<150) point_x=150;
               if(point_x>406) point_x=406;
               if(point_y<150) point_y=150;
               if(point_y>278) point_y=278;
               lineto(point_x,point_y);
          }

          setcolor(menu_chr);
          line(150,150,150,278);
          line(150,214,406,214);

          if(cursor_scrder<1)
             moveto(150,214-(int)((dat[0]/32)*cursor_magy));
          else
             moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
          for (i=0;i<256;i++)
          {
               point_x=(int)(i*cursor_magx)+150+cursor_scrder;
               point_y=214-(int)((dat[i]/32)*cursor_magy);
               if(point_x<150) point_x=150;
               if(point_x>406)
               {
                setcolor(vent_fondo);
                point_x=406;
               }
               if(point_y<150) point_y=150;
               if(point_y>278) point_y=278;
               lineto(point_x,point_y);
          }
          break;
```

```
case 6: /* scroll izquierda */
        k=cursor_scrder;
        cursor_scrder-=10;

        if(k<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+k,214-(int)((dat[0]/32)*cursor_magy));
        setcolor(vent_fondo);
        for (i=0;i<256;i++)
        {
             point_x=(int)(i*cursor_magx)+150+k;
             point_y=214-(int)((dat[i]/32)*cursor_magy);
             if(point_x<150) point_x=150;
             if(point_x>406) point_x=406;
             if(point_y<150) point_y=150;
             if(point_y>278) point_y=278;
             lineto(point_x,point_y);
        }

        setcolor(menu_chr);
        line(150,150,150,278);
        line(150,214,406,214);

        if(cursor_scrder<1)
           moveto(150,214-(int)((dat[0]/32)*cursor_magy));
        else
           moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
        for (i=0;i<256;i++)
        {
             point_x=(int)(i*cursor_magx)+150+cursor_scrder;
             point_y=214-(int)((dat[i]/32)*cursor_magy);
             if(point_x<150) point_x=150;
             if(point_x>406)
             {
              setcolor(vent_fondo);
              point_x=406;
             }
             if(point_x<150) point_x=150;
             if(point_y<150) point_y=150;
             if(point_y>278) point_y=278;
             lineto(point_x,point_y);
        }
        break;

case 7: /* medicion de parametros */
          Mshow(TRUE);
          salida=0;
          setwritemode(1);
          do
          {

           if (entra_ventana(150,150,406,278))
           {
               Mshow(FALSE);
               setcolor(EGA_CYAN);
               point_x=(int)(((status.xaxis-150)*cursor_magx)+150)
                                  +cursor_scrder;
               point_y=214-(int)((dat[status.xaxis-150]/32)*cursor_magy);
               poshoriz=status.xaxis;

               status=Mpos();
```

```
if(point_x<150) point_x=150;
if(point_x>406) point_x=406;
if(point_y<150) point_y=150;
if(point_y>278) point_y=278;
if ((point_x>150)&&(point_x<406)&&
   (point_y>=150)&&(point_y<=278)&&
   (status.xaxis>150)&&(status.xaxis<406)&&
   (status.yaxis>150)&&(status.yaxis<278))
{
  line(point_x-4,point_y+4,point_x+4,point_y-4);
  line(point_x-4,point_y-4,point_x+4,point_y+4);

  ant_x=point_x;
  ant_y=point_y;
}
else
{
  ant_x=150;
  ant_y=278;
}

do
{
    if ((poshoriz != status.xaxis)&&
        (entra_ventana(150,150,406,278)))
    {
      setcolor(EGA_CYAN);
      point_x=(int)(((status.xaxis-150)*cursor_magx)+150)
                    +cursor_scrder;
      point_y=214-(int)((dat[status.xaxis-150]/32)
                    *cursor_magy);
      poshoriz=status.xaxis;

      status=Hpos();
      if(point_x<150) point_x=150;
      if(point_x>406) point_x=406;
      if(point_y<150) point_y=150;
      if(point_y>278) point_y=278;
      if ((point_x>150)&&(point_x<406)&&
         (point_y>=150)&&(point_y<=278)&&
         (ant_x>150)&&(ant_x<406)&&
         (ant_y>=150)&&(ant_y<=278) &&
         (status.xaxis>150)&&(status.xaxis<406)&&
         (status.yaxis>150)&&(status.yaxis<278))
      {
         line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
         line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);

         line(point_x-4,point_y+4,point_x+4,point_y-4);
         line(point_x-4,point_y-4,point_x+4,point_y+4);

                  ant_x=point_x;
         ant_y=point_y;
      }

    }

    status=Hpressed(ButtonL);
    if (status.button_status==1)
    {
     setwritemode(0);
     ptmedir=salva_vent(300,305,445,350);
```

```
                suelta_boton();
                ventana(300,305,445,350,EGA_GREEN,EGA_LIGHTGREEN);

                setcolor(EGA_WHITE);
                sprintf(buffer,"Tiempo=%2.3fs",((float)
                        (status.xaxis-150)*aw_time)/256);
                moveto(305,320);
                outtext(buffer);

                sprintf(buffer,"Voltaje = %2.3f V",((float)
                        (dat[status.xaxis-150])*AD_RANGO)/2048);
                moveto(305,340);
                outtext(buffer);

                do
                {
                    status=Mpressed(ButtonL);
                }while(status.button_status!=1);
                suelta_boton();
                restaura_vent(300,305,ptmedir);
                status.button_status=0;
                setwritemode(1);
              }
            }while(entra_ventana(150,150,406,278));

            salida=sal_ventana(150,150,406,278);
            if(!salida)
            {
              Mshow(TRUE);
              if((ant_x>150)&&(ant_x<406)&&
                (ant_y>=150)&&(ant_y<=278))
              {
                line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
                line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);
              }
            }
          }
        }while(!salida);
        if((ant_x>150)&&(ant_x<406)&&
         (ant_y>=150)&&(ant_y<=278))
        {
         line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
         line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);
        }
        vent_fondo=EGA_DARKGRAY;
        setwritemode(0);
        break;

case 8:  /*marca un intervalo y calcula Dv y Dt */

        Mshow(TRUE);
        settextstyle(SMALL_FONT,HORIZ_DIR,5);
        parametros_menu(320,305,470,380,EGA_CYAN,
                        EGA_LIGHTCYAN,EGA_WHITE,1,ptmenuint);
        parametros_opcion(EGA_BLUE,EGA_LIGHTBLUE,EGA_WHITE,8);
        menu_enter("Integral total");
        menu_enter("Integral segmento");
        menu_enter("Salir");

        do
        {
         op2=menu();
```

```
switch(op2)
{
   case 1:
         integral=0.0;
         dat1[0]=(float)dat[0];
         for(i=0;i<256;i++)
         {
            dat1[i+1]=dat1[i]+(float)dat[i+1]; /* Sumas de */
            integral+=fabs((float)dat[i]);     /* Reimann */
         }

         ptintegral=salva_vent(10,300,266,450);
         ventana(10,300,266,450,LIGHTGRAY,EGA_WHITE);

         line(10,364,266,364);
         moveto(10,364);
         mayor = 0.0;
         for (i=0;i<256;i++)
         {
            if ((fabs(dat1[i])) > mayor)
             mayor = fabs(dat1[i]);
         }


         for (i=0;i<256;i++)
            lineto(i+10,364-(int)(dat1[i]*64/mayor) );

         sprintf(buffer,"Integral Total = %4.2f",integral);
         moveto(50,445);
         outtext(buffer);

         do
         {
           status=Mpressed(ButtonL);
         }while(!status.button_status);
         suelta_boton();
         restaura_vent(10,300,ptintegral);

         break;

   case 2:
         salida=0;
         setwritemode(1);
         do
         {
            Mshow(TRUE);
            if (entra_ventana(150,150,406,278))
            {
              Mshow(FALSE);
              setcolor(EGA_CYAN);
              point_x=(int)(((status.xaxis-150)*
                      cursor_magx)+150)+cursor_scrder;
              point_y=214-(int)((dat[status.xaxis-
                      150]/32)*cursor_magy);
              poshoriz=status.xaxis;

              status=Mpos();
              if(point_x<150) point_x=150;
              if(point_x>406) point_x=406;
              if(point_y<150) point_y=150;
              if(point_y>278) point_y=278;
              if ((point_x>150)&&(point_x<406)&&
```

```
   (point_y>=150)&&(point_y<=278)&&
   (status.xaxis>150)&&(status.xaxis<406)&&
   (status.yaxis>150)&&(status.yaxis<278))
{
   line(point_x-4,point_y+4,point_x+4,point_y-4);
   line(point_x-4,point_y-4,point_x+4,point_y+4);

   ant_x=point_x;
   ant_y=point_y;
}
else
{
   ant_x=150;
   ant_y=278;
}

do
{
   if ((poshoriz != status.xaxis)&&
       (entra_ventana(150,150,406,278)))
   {
    setcolor(EGA_CYAN);
    point_x=(int)(((status.xaxis-150)
            *cursor_magx)+150)+cursor_scrder;
    point_y=214-(int)((dat[status.xaxis-150]/32)
            *cursor_magy);
    poshoriz=status.xaxis;

    status=Mpos();
    if(point_x<150) point_x=150;
    if(point_x>406) point_x=406;
    if(point_y<150) point_y=150;
    if(point_y>278) point_y=278;
    if ((point_x>150)&&(point_x<406)&&
       (point_y>=150)&&(point_y<=278)&&
       (ant_x>150)&&(ant_x<406)&&
       (ant_y>=150)&&(ant_y<=278) &&
       (status.xaxis>150)&&(status.xaxis<406)&&
       (status.yaxis>150)&&(status.yaxis<278))
    {
       line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
       line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);
       line(point_x-4,point_y+4,
            point_x+4,point_y-4);
       line(point_x-4,point_y-4,
            point_x+4,point_y+4);

       ant_x=point_x;
       ant_y=point_y;
    }
   }

   status=Mpressed(ButtonL);
   if (status.button_status==1)
   {
    indice[cuenta_boton]= status.xaxis-150;
    cuenta_boton++;
    suelta_boton();
   }

   if (cuenta_boton==2)
   {
```

```
integral=0;
indice1=indice[0];
indice2=indice[1];
if (indice1 > indice2)
{    menor=indice2;
     indice2=indice1;
     indice1=menor;
}
setcolor(EGA_BLUE);
for(i=indice1;i<=indice2;i++)
   line(i*cursor_magx+150+cursor_scrder,150,
        i*cursor_magx+150+cursor_scrder,278);

dat1[indice1]=(float)dat[indice1];
for(i=indice1;i<=indice2;i++)
{
   dat1[i+1]=dat1[i]+(float)dat[i+1];
   integral+=fabs((float)dat[i]);
   /* Sumas de Reimann del intervalo */
}

            setwritemode(0);
ptintegral=salva_vent(10,278,266,450);
ventana(10,278,266,450,LIGHTGRAY,EGA_WHITE);

line(10,364,266,364);
moveto(10,364);
mayor = 0.0;

for (i=indice1;i<=indice2;i++)
{
   if ((fabs(dat1[i])) > mayor)
      mayor = fabs(dat1[i]);
}

moveto(indice1+10,364);
for (i=indice1;i<=indice2;i++)
   lineto(i+10,364-(int)(dat1[i]*64/mayor));
datmax=dat[indice1];
datmin=dat[indice1];
for (i=indice1;i<=indice2;i++) /* encuentra
                    mayor datos originales */
{
   if ( dat[i] > datmax)
      datmax = dat[i];
}

for (i=indice1;i<=indice2;i++) /* encuentra
                    menor datos originales */
{
   if ( dat[i] < datmin)
      datmin = dat[i];
}
if ( (datmax<0) && (datmin<0) )
    /* ambos signos negativos */
   dv= ((fabs(datmin)-fabs(datmax))
            *AD_RANGO)/2048;
else
{
   if ( (datmax>0) && (datmin>0) )
       /* ambos signos positivos */
      dv=((datmax-datmin)*AD_RANGO)/2048;
```

```
                        else
                           dv=((datmax + fabs(datmin))
                                 *AD_RANGO)/2048;
                           /* signos contrarios */
                     }

                     sprintf(buffer,"Dt = %2.3f seg  Dv = %2.3f
                           volts ",((float)(indice2-indice1)*
                              sw_time)/256,dv);
                     moveto(20,445);
                     outtext(buffer);

                     sprintf(buffer,"Integral de segmento = %2.1f
                           ",integral);
                     moveto(30,290);
                     outtext(buffer);

                     do
                     {
                        status=Mpressed(ButtonL);
                     }while(status.button_status!=1);

                     restaura_vent(10,278,ptintegral);
                     status.button_status=0;
                     cuenta_boton=0;
                     setwritemode(1);
                     setcolor(EGA_BLUE);
                     for(i=indice1;i<=indice2;i++)
                        line(i*cursor_magx+150+cursor_scrdor,
                                 150,i*cursor_magx+150
                                 +cursor_scrdor,278);
                  }
               }while(entra_ventana(150,150,406,278));

            salida=sal_ventana(150,150,406,278);
            if(!salida)
            {
               Mshow(TRUE);
               if((ant_x>150)&&(ant_x<406)&&
                  (ant_y>=150)&&(ant_y<=278))
               {
                  line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
                  line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);
               }
            }
            }
         }while(!salida);
         if((ant_x>150)&&(ant_x<406)&&
            (ant_y>=150)&&(ant_y<=278))
         {
            line(ant_x-4,ant_y+4,ant_x+4,ant_y-4);
            line(ant_x-4,ant_y-4,ant_x+4,ant_y+4);
         }
         vent_fondo=EGA_DARKGRAY;
         setwritemode(0);
         break;

   case 3:
         op2=3;
         break;
 }
}while(op2<3);
```

```
            vent_fondo=EGA_DARKGRAY;
            setwritemode(0);
            break;

case 9: /* recupera senal original (reset) */

      setcolor(vent_fondo);
      sprintf(buffer,"%2.1f",(float)(AD_RANGO/cursor_magy));
      moveto(120,155);
      outtext(buffer);

      sprintf(buffer,"-%2.1f",(float)(AD_RANGO/cursor_magy));
      moveto(113,275);
      outtext(buffer);

      sprintf(buffer,"%1.3f",sw_time/cursor_magx);
      moveto(408,214);
      outtext(buffer);

      if(cursor_scrder<1)
         moveto(150,214-(int)((dat[0]/32)*cursor_magy));
      else
         moveto(150+cursor_scrder,214-(int)((dat[0]/32)*cursor_magy));
      for (i=0;i<256;i++)
      {
           point_x=(int)(i*cursor_magx)+150+cursor_scrder;
           point_y=214-(int)((dat[i]/32)*cursor_magy);
           if(point_x<150) point_x=150;
           if(point_x>406)
           {
            setcolor(vent_fondo);
            point_x=406;
           }
           if(point_y<150) point_y=150;
           if(point_y>278) point_y=278;
           lineto(point_x,point_y);
      }

      cursor_magx=1;
      cursor_magy=1;
      cursor_scrder=0;
      k=0;

      moveto(150,214);
      setcolor(menu_chr);
      line(150,150,150,278);
      line(150,214,406,214);

      sprintf(buffer,"%2.1f",(float)(AD_RANGO/cursor_magy));
      moveto(120,155);
      outtext(buffer);

      sprintf(buffer,"-%2.1f",(float)(AD_RANGO/cursor_magy));
      moveto(113,275);
      outtext(buffer);

      sprintf(buffer,"%1.3f",sw_time/cursor_magx);
      moveto(408,214);
      outtext(buffer);

      moveto(150,214-(dat[0]/32));
```

```
                for (i=0;i<256;i++)
                    lineto(i+150,214-(dat[i]/32));
                break;

        case 10: /* salir */
                restaura_vent(x,y,ptcursor);
                op1=10;
                break;
        }
  }while(op1<10);
}



/************************************************/
/**** Funcion que hace la correlacion cruzada ***/
/**** o la autocorrelacion de amplitudes  *******/
/************************************************/
void correlacion(int canal1,int canal2)
{
   int far   *d;
   int       i,j;
   int far   *ptcorr;
   char      buffer[80];

   d=adbuff;
   d+=(int)canal1-1;
   for (i=0;i<256;i++)
   {
      x[i]=*(d);
      d+=16;
   }

   d=adbuff;
   d+=(int)canal2-1;
   for (i=0;i<256;i++)
   {
      y[i]=*(d);
      d+=16;
   }


   for(i=0;i<256;i++)
   {
      cor[i]=0;
      for(j=1;j<255;j++)
       cor[i]=cor[i]+x[j]*y[j+i];
      cor[i]=cor[i]/256;
    }


   if (canal1==canal2)
   {
      mayor=cor[0];
      for(i=0;i<256;i++)
       cor[i]=cor[i]/mayor;
   }
   else
   {
      mayor=0.0;
      for(i=0;i<256;i++)
```

```
      {
       if( (fabs(cor[i])) > mayor)
          mayor= fabs(cor[i]);
      }

      for(i=0;i<256;i++)
      cor[i]=cor[i]/mayor;
   }

   ptcorr=salva_vent(150,100,506,328);
   ventana(150,100,506,328,EGA_LIGHTGRAY,EGA_YELLOW);
   setcolor(EGA_DARKGRAY);
   line(200,145,200,283);
   line(200,214,456,214);

   if(canal1==canal2)
   {
      sprintf(buffer,"Auto-correlacion de Amplitud");
      moveto(210,120);
      outtext(buffer);
      sprintf(buffer," Canal %2d",canal1);
      moveto(250,300);
      outtext(buffer);
   }
   else
   {
      sprintf(buffer,"Correlacion cruzada de Amplitud");
      moveto(230,120);
      outtext(buffer);
      sprintf(buffer,"Canal %2d con Canal %2d",canal1,canal2);
      moveto(270,300);
      outtext(buffer);
   }

   setcolor(EGA_YELLOW);
   moveto(200,214-ceil(cor[0]*70));
   for (i=0;i<256;i++)
      lineto(i+200,214-ceil(cor[i]*70));

   do
   {
      status=Mpressed(ButtonL);
   }while(status.button_status!=1);
   suelta_boton();
   restaura_vent(150,100,ptcorr);
}
```

```
/*********************************************************/
/*** funcion que selecciona canal y establece umbral     **/
/*** para el calculo de la autocorrelacion de intervalos **/
/*********************************************************/
void canal_umbral(int canal)
{
   int       cont;
   int       dato;
   int       posy;
   int       bandera=0;
   int       bandera2=0;
   int far *ptinter;


   ptinter=salva_vent(50,292,590,430);
   ventana(50,292,590,430,EGA_DARKGRAY,EGA_WHITE);
   setcolor(EGA_WHITE);
   line(51,361,589,361);

   do
   {
      for(cont=0;cont<=538;cont++)
      {
       dato=convierte(canal-1);
       plot(cont+50,361-(int)(dato>>5),15);
       status=Mpressed(ButtonL);
       if(status.button_status==1)
       {
            suelta_boton();
          Mshow(TRUE);
          status.button_status=0;
          do{
             status=Mpressed(ButtonL);
          }while(status.button_status!=1);
          suelta_boton();
          Mshow(FALSE);
          status=Mpos();
          posy=status.yaxis;
          setcolor(EGA_GREEN);
          line(51,posy,589,posy);
          bandera=1;
       }
       status=Mpressed(ButtonL);
       status=Mpos();
       if(status.button_status==2)
       {
          bandera2=1;
          break;
       }
       status.button_status=0;
      }

      if (bandera2==1) break;
      ventana(50,292,590,430,EGA_DARKGRAY,EGA_WHITE);
      if (bandera==1)
      {
       setcolor(EGA_GREEN);
       line(51,posy,589,posy);
      }
      setcolor(EGA_WHITE);
      line(51,361,589,361);
   }while(1);
```

```
   suelta_boton();
   umbral=(361-posy)*32;
   bandera=0;
   bandera2=0;
   restaura_vent(50,292,ptinter);
}


/*************************************************************/
/*** funcion que cuenta el numero y el tamano de intervalos *****/
/*** para el calculo de la autocorrelacion de intervalos ********/
/*************************************************************/
void cuenta_intervalos(int canal,int num_intervalo)
{
   int  j=0;
   int  bandera=1;
   int  acum=0;
   int  dato1,dato2;

   intervalo[-1]=0;
   do
   {
      dato1=convierte(canal-1);
      dato2=convierte(canal-1);
      if( (dato1>umbral) && (bandera==1) )
      {
       intervalo[j]=acum+intervalo[j-1];
       j+=1;
       acum=0;
       bandera=0;
      }
      else
      {
       if( (dato1< umbral) && (dato2>umbral) )
          bandera=1;
       acum+=1;
      }
   }while(j<num_intervalo);
}


/************************************************/
/*** funcion que realize la auto-correlacion de ***/
/*** intervalos a un arreglo de 512 ptos. con una */
/*** ventana de correlacion de 256 ptos.        ***/
/************************************************/
void auto_cor_int(int canal,int num_intervalo,int num_correlacion)
{
   char     buffer[80];
   int      t,i,c,k,k1,g;
   float    d,p,q,escala;
   double   f;

   int far *ptcorint;

   t=num_intervalo-num_correlacion;
   g=1;
```

```
/******** algoritmo correlacion ****/

for(c=0;c<num_correlacion;c++)
{
   autocor[c]=0;
   p=c;
   q=c+.95*g;
   for(i=1;i<t;i++)
   {
    k1=3+ceil(.1*c);
    for (k=0;k<k1;k++)
       if ( (intervalo[i+k]) >= (intervalo[i]+p) )  goto UNO;
    goto DOS;
    UNO: if ( (intervalo[i+k]) > (intervalo[i]+q) ) goto DOS;
    autocor[c]+=1;
    DOS:;
   }
   autocor[c]=autocor[c]/t;
}

/************ desplegado resultado  *******/

ptcorint=salva_vent(130,100,486,328);
ventana(130,100,486,328,EGA_LIGHTGRAY,EGA_YELLOW);
setcolor(EGA_DARKGRAY);
line(180,145,180,283);
line(180,278,436,278);

sprintf(buffer,"Auto-correlacion de Intervalos");
moveto(190,120);
outtext(buffer);
sprintf(buffer," Canal %2d",canal);
moveto(230,300);
outtext(buffer);

setcolor(EGA_YELLOW);
moveto(180,278-ceil(autocor[0]*128));

escala=(float)num_correlacion/256;
for(c=1;c<num_correlacion;c++)
   lineto((int)(c/escala)+180,278-ceil(autocor[c]*128));
do
{
   status=Mpressed(ButtonL);
}while(status.button_status!=1);
suelta_boton();
restaura_vent(130,100,ptcorint);

}


/*******************************************************/
/*    Esta es la funcion que calcula FFT.              */
/*    Se tradujo de pascal y no se optimizaron los     */
/*    saltos, de modo que se utilizaran goto's         */
/*    isign=1 fft directa, isign=-1 fft inversa        */
/*    m potencia de 2                                  */
/*******************************************************/
void  FFT(int m,int isign,int canal)
{
     #define pi 3.14159265358979
```

```
      /*** Definicion de variables locales a la funcion FFT ***/

      complex        u,w,t;
      complex        ncmplx;
      int            n,l,le,le1;
      int            i;
      char           opcion;
      double         xy;
      int            je,k,nml,nv2;
      int            ip,j,index;
      int  far       *d;

//incializa xf con el canal correspondiente de ADBUFF

      for(i=0;i<258;i++)
      {
         xf[i].re=0;
         xf[i].im=0;
      }

      d=adbuff;
      d+=(int)canal-1;
      for (i=0;i<256;i++)
      {
         xf[i].re=*(d);
         xf[i].im=0.0;
         d+=16;
      }

      n=l;
      for(index=0;(index++)<m;n*=2);
      for(l=1;(m+1)>l;l++)
      {
            je=m+1-l;
            le=1;
            for(index=0;(index++)!=je;le*=2);
            le1=le/2;
            u.re=1.0;
            u.im=0.0;
            w.re=(cos(pi/le1));
            w.im=(sin(pi*isign/le1));
            for(j=1;(le1+1)>j;j++)
            {
                  i=j;
                  while(i<n)
                  {
                        ip=i+le1;
                        add(xf[i],xf[ip],&t);
                        xf[ip].re=(xf[i].re-xf[ip].re);
                        xf[ip].im=(xf[i].im-xf[ip].im);
                        mul(xf[ip],u,&xf[ip]);
                        xf[i]=t;
                        i=i+le;
                  }
                  mul(u,w,&u);
            }
      }
      nv2=n/2;
      nml=n-1;
      j=1;
      for(i=1;nml>i;i++)
      {
```

```
            if(i>=j) goto 125;
            t=xf[j];
            xf[j]=xf[i];
            xf[i]=t;
125:  k=nv2;
126:  if(k>=j)goto 130;
            j=j-k;
            k=k/2;
            goto 126;
130:  j=j+k;
      }
      ncmplx.re=n;
      ncmplx.im=0.0;
      if(isign>0)
      {
            for(i=0;n>i;i++)
                  dvd(xf[i],ncmplx,&xf[i]);
      }
}


/*************************************************************/

/*** Definicion de funciones de osciloscopio ***/
/*** Unidad OSCILO.H **/

/** Definicion de constantes */
#define RANGO  2

/* Variables globales */
extern int   os_dato;                  // dato convertido
extern int os_fin;
extern  float seg_div;                 // segundos por division
extern  float vol_div;                 // volts por division
extern  float os_factor;               // factor de division de voltaje
extern  int os_rate;                   // frecuencia de muestreo
extern  int os_num_canal;              //variable que selecciona si canal 1-8 o
                                       // canal 9-16


/* Encabezado de funciones */
extern int os_convierte(int canal);
extern void os_set_sample(void);
extern void os_set_ad(int rate);
extern void os_pantalla(void);
extern void os_menu_principal(void);
extern void os_encabezado(void);

extern int os_menu(int nopciones,int c_barra1,int c_letrero1,
                   int c_barra2,int c_letrero2,int marco);
extern void osciloscopio(void);
extern float os_readfloat(char *texto);
```