17
2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

" DESARROLLO DE UN SISTEMA DE COMPUTO PARA ADQUISICION Y ANALISIS DE SEÑALES FISIOLOGICAS "

T E S I S

QUE PARA OBTENER EL TITULO DE

A C T U A R I O

P R E S E N T A

ALFREDO CALLEJAS CHAVERO

DIRECTOR DE TESIS: DRA HORTENSIA GONZALEZ GOMEZ

1997

VNIVERSIDAD NACIONAL
AVÉNOMA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis:
" DESARROLLO DE UN SISTEMA DE COMPUTO PARA ADQUISICION Y ANALISIS DE SEÑALES FISIOLOGICAS "

realizado por CALLEJAS CHAVERO ALFREDO

con número de cuenta 8933217-9 , pasante de la carrera de ACTUARIA

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	DRA. HORTENSIA GONZALEZ GOMEZ
Propietario	DR. JORGE HUMBERTO ARCE RINCON
Propietario	FIS. JUAN JESUS GUTIERREZ GARCIA
Suplente	M. en C. GUADALUPE IBARGÜENGOITIA GONZALEZ
Suplente	ACT. HORTENSIA CANO GRANADOS

Consejo Departamental de Matemáticas

ACT. AGUSTIN ROMAN AGUILAR

A MIS PADRES, HERMANOS, AMIGOS Y PROFESORES QUE CON SU APOYO, SUS CONSEJOS, Y SUS ENSEÑANZAS HICIERON POSIBLE LA CULMINACIÓN DE ESTA ETAPA TAN IMPORTANTE DE MI VIDA.

DESARROLLO DE UN SISTEMA DE CÓMPUTO PARA ADQUISICIÓN Y ANÁLISIS DE SEÑALES FISIOLÓGICAS

ÍNDICE

INTRODUCCIÓN

Desde su aparición, el hombre ha tratado de crear y tener herramientas que le faciliten el desarrollo de sus diversas tareas. Para ello ha tenido que imitar las acciones de muchas partes del cuerpo humano e inclusive de algunos otros animales. En esta tarea de construir herramientas que faciliten su trabajo, tal vez el reto más importante que ha tomado es tratar de construir una herramienta que le permita simular al cerebro humano, llevándolo a desarrollar la **Computadora**.

El desarrollo de la computadora se ha dado sobre todo en los últimos 50 años, durante los cuales se han producido adelantos tecnológicos importantes. Desde su creación, la computadora ha tenido diversas aplicaciones, pero sin duda ha representado siempre un gran apoyo a la ciencia en cuanto a procesamiento automático de datos se refiere. En el pasado, muchos problemas científicos habían quedado sin solución, debido a la tarea imposible de procesar una gran variedad de complicadas fórmulas con métodos manuales. Actualmente, la utilización del procesamiento electrónico de datos permite a los hombres de ciencia contar con los instrumentos necesarios para manejar esos complicados cálculos. Los problemas que habrían requerido toda una vida de cálculos para su solución, se procesan ahora con ayuda de las computadoras en unos cuantos minutos. Así, con los nuevos instrumentos, las investigaciones científicas tienen nuevos horizontes.

Las computadoras se dedican ahora de manera especializada a tareas como análisis de resultados, control inteligente de protocolos experimentales, digitalización, reconocimiento y análisis de imágenes, etc. Pasan entonces a ser parte de los equipos, instrumentos de análisis y archivo de los laboratorios de investigación, con un papel preponderante en el desarrollo de los experimentos. Desde la aparición misma de la computadora, se desarrollaron diversos métodos que permitieran que la información generada por señales fisiológicas pudiera ser capturada por la computadora, para posteriormente realizar análisis diversos con su ayuda. La presente tesis presenta una breve historia acerca de como la computadora fue y es utilizada para realizar adquisición y análisis de datos derivados de señales fisiológicas.

Para que una computadora pueda hacer uso de la información generada por señales fisiológicas, son necesarios un conjunto de instrumentos que la adecuen, además una interface de laboratorio, que generalmente toma la forma de una tarjeta de circuitos, la cual permite la comunicación con la computadora, y así poder tener control por medio de un programa de cómputo de las adquisiciones de datos que se realizan.

Los programas utilizados para la adquisición de datos adquieren un papel importante en el desarrollo de un sistema de cómputo para la adquisición y análisis de señales fisiológicas. La elección del "software" que se va a utilizar, es entonces uno de los puntos que deben de tomarse muy en cuenta cuando se desarrolla un sistema de cómputo con este propósito. De entre las opciones comerciales destaca ASYST, del cual hablaremos ampliamente, sus ventajas, sus limitaciones y sus requerimientos.

Para desarrollar un sistema de cómputo se deben de tomar en cuenta diversos aspectos que le permitan ser eficiente y confiable. Aspectos que están íntimamente ligados al desarrollo de los programas que componen el sistema, la relación entre ellos, y la forma en que se comunican con la tarjeta de adquisición, que es la fuente de información del sistema. Una vez realizada la adquisición, el sistema debe ser capaz de facilitar el análisis de los resultados obtenidos, y de permitir darle salida a la información.

El objetivo general de esta Tesis fue el desarrollo de un Sistema de Cómputo Para la Adquisición y Análisis de Señales Fisiológicas basado en un lenguaje comercial de programación. Durante este proceso se cubrieron otros objetivos particulares: una investigación que proporcionara mayor conocimiento de los diversos métodos que se han utilizado y se utilizan para la adquisición de datos por medio de una computadora. El análisis de las características del "software" empleado, ASYST, que consideramos uno de los más apropiados para adquirir datos generados por señales fisiológicas. Finalmente, una revisión al arreglo experimental y los instrumentos necesarios para realizar la captura de datos, así como los principios básicos que deben de seguirse en el desarrollo de un sistema de cómputo dedicado a esta aplicación específica.

CAPÍTULO I

SEÑALES ELECTROFISIOLÓGICAS

1. ELECTROFISIOLOGÍA

En años recientes las computadoras digitales han venido a jugar un papel importante en el análisis de señales electrofisiológicas, por lo que se han implementado diversos procedimientos involucrados en los métodos de adquisición y análisis de señales . En la actualidad la electrofisiologia incluye los estudios de las corrientes eléctricas o potenciales asociados con el movimiento de iones a través de las membranas celulares. Esta disciplina engloba los siguientes temas: los potenciales de acción observados en tejidos excitables, la asociación iónica al flujo de corriente, potenciales pre y post sinápticos, la corriente asociada con neurotransmisiones y neurosecreciones, el estudio de fluctuaciones de corriente a través de un canal individual en la membrana de las células, electrofisiología cardiaca, electroencefalografía y electromiografía (EKG, EEG, EMG).[1]

Desde el punto de vista de procedimientos de análisis, todos esos métodos experimentales involucran la medición de la amplitud y la forma de las ondas eléctricas asociadas con los diversos fenómenos de la membrana. Este uso de la forma de onda para inferir propiedades a un nivel celular, distingue a la electrofisiología de los estudios neurofisiológicos referentes a los niveles de disparo de las neuronas, donde la forma de onda no es de interés primario.

1.1 Una breve historia de la Electrofisiología

Puede considerarse que la electrofisiología surgió a finales del siglo XVIII cuando el italiano Luigi Galvani notó que la aplicación de un potencial eléctrico a un nervio crural de un rana evocó un tirón en el músculo de su pata. A mediados del siglo XIX fue posible observar esto usando el galvanómetro de mercurio, asociando una señal eléctrica con la actividad del nervio o músculo. Para finales del siglo XIX un entendimiento de las propiedades físico-químicas de iones en solución fue desarrollada con varios trabajos de Arrhenius, Nernst y Planck. En 1902, Bernstein dio los fundamentos de la Electrofisiología moderna con su hipótesis de una membrana celular semipermeable y selectiva para iones de K+ , para explicar el impulso nervioso. Estos conceptos fueron fuertemente desarrollados con las mediciones posteriores de potenciales en reposo y en acción en el axón gigante del calamar. Más tarde, Cole desarrolló la técnica de fijación de voltaje (un circuito retroalimentador que permite el control del potencial de membrana celular),

[1] John Dempster "Computerr Analysis of Electrophysiological Signals" Press Inc, San Diego CA. , USA 1993

permitiendo por primera vez la medición directa de corriente iónica fluyendo a través de la membrana celular (1949). Hodgkin & Huxley usaron la técnica para deducir el papel jugado por los iones de Na+ y K+ en el potencial de acción (1952). Por ese mismo tiempo el desarrollo del electrodo como micropipeta de vidrio permitió mediciones de potencial intracelular siendo extendida al músculo esquelético. Con el desarrollo del "patch clamp" en el año de 1976, fue posible estudiar la función detallada de un canal iónico, - el mecanismo básico por el cual los iones atraviesan la membrana celular-.

La primera técnica de registro usada por Hodgkin & Huxley en su trabajo en el axón de calamar fue un film fotográfico. Señales de corriente y voltaje fueron desplegadas en una pantalla del osciloscopio y grabadas en cinta de 35 mm usando una cámara motorizada con un obturador sincronizado al barrido del osciloscopio. Las grabaciones eran posteriormente procesadas y medidas manualmente desde impresiones o imágenes agrandadas. El análisis de cada experimento obtenido era substancialmente más largo que su ejecución, un día de experimentos producía semanas de trabajo de análisis. Esta manera de trabajar fue la norma de muchos años, desde mediados del los 60's y como hasta los finales de los 70's. Desde entonces sin embargo, la colección y análisis de señales se ha ido incrementando al usar la computadora digital. Este uso entusiasta de la computadora ha contrastado con otras técnicas de registro fisiológico, las cuales han tenido pequeños cambios en los últimos 40 años.

Una característica notable de los electrofisiólogos a lo largo de su vida científica, ha sido su predisposición para aplicar los descubrimientos en sistemas de computadoras en sus trabajos de laboratorio, teniendo muchas razones para esto. Muchas señales electrofisiológicas son eventos eléctricos breves, de pequeña amplitud, se obtienen superponiéndolos en un gran fondo, además se tiene dificultad para adquirir datos usando los dispositivos más convencionales de grabación. Desde esta perspectiva la técnica ha puesto límites en cuanto a tecnología disponible, particularmente en sistemas con gran impedancia de entrada, amplificación de bajo ruido y diferentes medios de grabación utilizados (como el caso hasta hace unos años de la cinta magnética). Esta orientación de descubrimientos en el campo de las nuevas tecnologías y las limitaciones de los métodos existentes, crearon una gran expectación para explorar la posibilidad de la computadora, no obstante del costo y las dificultades que ello involucraba. En suma, la interpretación de las señales obtenidas demandaba un análisis detallado y cuantitativo, algunas veces involucraban modelos matemáticos, el largo número de mediciones que se requerían para ello debieron ser mucha carga de trabajo. Así pues el potencial de las computadoras para automatizar esas mediciones era una de sus grandes atracciones.

1.2 Los Principios de la Computadora Digital en el Laboratorio según Dempster[2]

El desarrollo de computadoras digitales puede ser dividido en una serie de fases muy ligadas a la tecnología electrónica disponible. De 1940 a 1952 el periodo puede ser considerado como una fase experimental del desarrollo de la computadora . Ecker & Mauchly crearon la ENIAC (Electronic Numerical Integrator and Calculator) en el año de

[2] John Dempster " Computer Analysis of Electrophysiological Signals" Press Inc, San Diego CA. , USA 1993

1946 en la Universidad de Pennsylvania. Esta usaba 18000 bulbos (tubos al vacío), pesaba 30 toneladas, media 35 m de longitud, y en uso consumia 140 kilowatts de energía. En la siguiente década numerosas máquinas individuales fueron manejadas con habilidad en laboratorios de investigación, principalmente como calculadoras mecánicas.

Para 1951 se desarrollo la primera computadora digital comercialmente producida para propósitos generales llamada UNIVAC 1 (Universal Automatic Computer), la cual fue comisionada para el censo de los Estados Unidos en 1950. La UNIVAC 1 fue la primera computadora cuyo diseño estaba enfocado tanto a aplicaciones comerciales como científicas. La primera computadora IBM (International Business Machines), la IBM 701 fue creada en 1952, esta computadora se desarrolló primordialmente para aplicaciones científicas. Se consideran las antecesoras de las máquinas conocidas como "mainframes". Cada máquina era muy grande, del tamaño de una pieza, sus dispositivos eran demasiado grandes y generaban mucho calor, por lo que requerían de complejos sistemas de enfriamiento. La computadora Whirlwind requería de 150 kW de energía, que era similar a la energía necesaria para iluminar a todo Cambridge Massachusetts, además fue una computadora demasiado cara en comparación con el precio de una supercomputadora moderna. Las Universidades y organizaciones de investigación comenzaron a adquirirlas como fuentes centrales a mediados de los 50's, pero el gran costo y complejidad de la "mainframe" la excluyó de usarse en un laboratorio individual. Asimismo, su uso para la adquisición de datos se dio hasta muchos años después y fue el recurso central para la simulación natural en el cálculo de las ecuaciones de Hodgkin-Huxley.

El estímulo inicial para desarrollar sistemas de computadoras, los cuales pudieran adquirir y procesar datos en tiempo real (un requerimiento clave para la adquisición de datos en el laboratorio) llegó de la militarización. En particular el proyecto Whirlwind en el Laboratorio Lincoln de el Instituto de Tecnología de Massachusetts (MIT) formó el centro principal del sistema de defensa aérea de los Estados Unidos y fue crucial en el desarrollo eventual de la computadora en el laboratorio. Ese MIT científico fue también probablemente el primero en usar la computadora como un vehículo para experimentos de laboratorio, aunque en estudios psicofisiológicos asociados con el proyecto de defensa aérea (1959). En ese tiempo Kenneth Olsen, el fundador de la Digital Equipment Corporation (DEC), fue un alumno de este laboratorio.

Así las computadoras continuaron su desarrollo para superar los años 50's , en particular el remplazamiento de bulbos por transistores en la IBM 7090 en el año de 1959, redujeron grandemente el consumo de energía e incrementaron la rentabilidad. Los transistores asimismo fueron remplazados por circuitos integrados (ICs) como en la serie 360 de IBM de 1964. El desarrollo de circuitos integrados a un dispositivo multi-transistor hecho con simples piezas de silicon, permitió a la computadora incrementar rapidez, complejidad y confiabilidad, mientras al mismo tiempo bajaban el costo de construcción. Este proceso continua incrementándose hasta nuestros días.

Para principios de los 60's un gran número de investigadores, particularmente en los Estados Unidos, exploraban el potencial de las computadoras en el análisis de datos de laboratorio. Para entonces se realizó un esfuerzo para compensar el enorme costo de las "mainframes", permitiendo a algunos usuarios accesos simultáneos para usar la computadora, a este proceso se le conoce como tiempo compartido. La cinta magnética hizo su aparición, la cual debería grabar las señales analógicas durante los experimentos.

Posteriormente cada cinta debería ser tomada y ser reproducida en la "mainframe" para ser procesada.

1.3 Las Minicomputadoras - La LINC y la PDP-11 -

A principios de los 60's el costo de una computadora digital empezó a disminuir, esto se debió al desarrollo de circuitos integrados de media escala, y la minicomputadora llegó a nuestra existencia. Esta clase de computadoras fue significativamente más lenta que algunas "mainframes" pero fueron mucho más baratas. Algunas máquinas fueron demasiado caras para un laboratorio individual, pero su adquisición se justificaba por todo un departamento. Consecuentemente los sistemas fueron desarrollándose, lo cual permitió que las señales experimentales fueran transmitidas a lo largo de líneas telefónicas desde los laboratorios, al departamento de cómputo. La Hartline fue una de las primeras en aplicar la computadora en experimentación fisiológica usando el registro de frecuencias de los disparos nerviosos del ojo del *Limulus* (molusco), para una variedad de estímulos de luz generados por la computadora. Como era usual, fue también usada para computar las ecuaciones de Hodgkin-Huxley.

En el simposium de la Academia de Ciencias de Nueva York bajo el tema 'Computadoras en Medicina y Biología' en 1964, hubo un reporte que puede ser observado como el nacimiento de la computadora de laboratorio, la LINC (Laboratory Instrument Computer). Clark & Molnar (1964) de el Laboratorio Lincoln de el MIT, describieron un pequeño sistema de cómputo equipado con convertidores analógico-digital (A/D) y digital-analógico (D/A) necesarios para digitalizar grabaciones de registros de señales experimentales y producir estímulos. Los resultados deberían ser desplegados gráficamente en la pantalla del osciloscopio vía convertidores D/A. Al mismo tiempo DEC produjo la primera de las minicomputadoras a bajo precio, la cual establecería su dominio en este campo, la PDP-8. Dicha computadora se construyó a un bajo costo y fue relativamente pequeña, fue la primera en usar el llamado Ómnibus de gran importancia en la arquitectura de computadoras. DEC desarrolló una variante de la PDP-8, la LINC-8 basada en el concepto primero de la LINC. Esta computadora llevó el primer sistema estandarizado que llegó a ser de uso general para los electrofisiólogos. Inicialmente estas computadoras carecían de discos magnéticos, los programas se almacenaban en tarjetas perforadas, en cinta perforada y cintas magnéticas. Eran primitivas para los estándares modernos teniendo 4 Kbytes de RAM y 'teletypewriters' (impresoras teleféricas), en lugar de pantalla visual de despliegue. Sin embargo fueron construidas para ser usadas en el laboratorio electrofisiológico y fueron muy usadas en trabajos clínicos, neurofisiología y electrofisiología de la membrana.

Un uso notable de estos sistemas de computadoras fue el análisis de voltaje activado por corriente de Na+ y K+ en el axón de el calamar. Armstrong & Bezanilla (1974) desarrollaron el sistema de criba digital de substracción de corriente para poder extraer registros precisos de corrientes específicas en registros combinados y contaminados con corriente de otras fuentes. Ellos desarrollaron su propia forma digital de registrar, la cual fue ligada a la PDP-8. Otro sistema de uso en ese tiempo fue la computadora LM[2] desarrollada por T. H. Kehl en la Universidad de Washington.

En 1970, DEC introdujo una minicomputadora de 16 bits, la PDP-11, la cual debería orientarse directamente a un aumento en la memoria hasta llegar a los 64 kbytes, y tuviera mucho más poder y elegancia en las instrucciones. Rápidamente fue aceptada como la sucesora de la PDP-8, y usada en el laboratorio hasta mediados de los 80's. Tanto el disco duro como el "floppy" (disco flexible) empezaron a hacerse comunes como dispositivos de almacenamiento.

Una de las razones para el éxito de la PDP-11 fue el rango de excelentes sistemas operativos, en particular el RT11 (Real Time 11) el cual se combinó con el alcance de compiladores para lenguajes de programación, y así por primera vez se produjo una plataforma estándar y poderosa en el desarrollo de software para aplicaciones en el laboratorio.

DEC optó por no proveer versiones especializadas para el laboratorio de la PDP-11 como sucedió con la LINC-8. En su lugar algunas pequeñas compañías especialistas, ambas en los Estados Unidos y la Gran Bretaña empezaron a producir PDP-11's basadas en sistemas equipados con gran ejecución de conversiones análogo-digital y sub-sistemas. En la Gran Bretaña el sistema 502 de el CED (Cambridge Electronic Design) buscó una aceptación general con Indec Inc. Este nuevo y poderoso sistema arribó en un tiempo fortuito para los electrofisiólogos. Las técnicas de análisis empezaron a ser desarrolladas y la computadora jugó una parte esencial en dicho desarrollo, en particular el cálculo de la conductancia de canales iónicos unitarios por poderosos análisis de espectros de fluctuaciones de corriente de iones. En 1976 el desarrollo de el método 'patch clamp' permitió el registro de canales iónicos unitarios. La estadística natural de cada señal requiere de la medida exacta de miles de pulsos de corriente en los canales de iones, por eso se necesita de una poderosa computadora digital.

En los E.U. una versión especifica del lenguaje Basic, el Basic-23 fue desarrollada por Brown & Hobbs para usarse en el laboratorio de electrofisiología. Este lenguaje combinaba la simplicidad del Basic con un extendido set de instrucciones diseñado para capturar, manipular y desplegar señales analógicas. Basic-23 fue usado por un gran número de programadores y fue un vehículo de éxito para la diseminación de aplicaciones de "software" de uso entre laboratorios.

En la Gran Bretaña se hizo más uso del lenguaje Fortran IV y Macro II (lenguaje ensamblador de la PDP-11), desarrollando "software" en un gran número de laboratorios, incluyendo ejemplos de programas para análisis de un canal unitario desarrollados por Colquhoun (1983) en la University College London y "software" de uso electrofisiológico desarrollado por algunos otros grupos entre 1975 y 1989. Para mediados de los 80's la utilidad de estos géneros de "software" empezó a reconocerse, pero fue obvio que su producción fue dificultosa y consumió tiempo, debido a que de este "software" se hicieron intercambios informales entre laboratorios, en los cuales alguien tenía relación o lazos con el desarrollador del "software" o se había interesado por demostraciones del "software" en encuentros científicos.

1.4 La Microcomputadora

Aunque algunos costos continuaron bajando y se realizaron mejoras continuas, la minicomputadora siguió durante su tiempo de vida como una computadora de laboratorio a un gran costo. Los laboratorios obtuvieron solamente parte de algunos de sus usos múltiples. En 1984 una PDP-11/23 costaba mucho más que el set completo para registros electrofisiológicos. Un cambio radical empezó a ocurrir con el desarrollo de el microprocesador en los años 70's, en donde una Unidad Central de Proceso (CPU) completa se reducía a simples circuitos integrados. Los CPU's de las minicomputadoras se construyeron típicamente como tarjetas de circuitos con múltiples circuitos integrados. Inicialmente los microprocesadores fueron simples computadores de 4 bits (Intel 4040) principalmente usados para el control de procesos industriales y construcción de calculadoras, pero la introducción del Intel 8080 en 1974 y de Motorola 6502 como microprocesadores de 8 bits hicieron posible la producción de una pequeña computadora a costos bajísimos y que nunca antes se había usado, - la primera microcomputadora-. Notables en esta generación fueron la Commodore Pet y la microcomputadora Apple. Su funcionamiento no fue tan diferente de la vieja minicomputadora PDP-8, pero su bajo costo fue determinante en su rápida y eficiente introducción en el laboratorio. Por primera vez el concepto de computadora personal, es decir una microcomputadora usada sola y exclusivamente por una persona, se introdujo en el ambiente computacional.

Al producirse en grandes volúmenes por primera vez estas nuevas computadoras, estimularon la innovación en el desarrollo de "software"; en particular, el programa VisiCalc para tabulación y ejecución de cálculos en tablas de números, y el procesador de palabras Wordstar. De esta forma las ecuaciones de Hodgking-Huxley fueron inmediatamente recalculadas en estas máquinas. La Apple II fue provista excepcionalmente para ser usada en trabajos de laboratorio, por su habilidad para aceptar una variedad de tarjetas de interfaces especializadas, incluyendo convertidores A/D y D/A, manejadores de discos, controladores de motor etc..

Paradójicamente la primera generación de microcomputadoras tiene un menor impacto en los laboratorios electrofisiológicos de lo que se esperaba. Esto se debió a la gran disparidad entre el poder de computo de la microcomputadora y la existente minicomputadora PDP-11. La PDP-11 fue una computadora de 16 bits y era en orden de magnitud muy rápida, soportando 10 Mbytes en disco duro y 12 bits en conversiones A/D con rangos de muestreo del orden de 25 khz. Un sustancial cuerpo de aplicaciones de software fueron desarrolladas, las cuales no serían soportadas por máquinas más pequeñas, como las de 8 bits. La primera generación de 8 bits alcanzó satisfactoriamente el gran nivel de lenguajes comparables en cualidades con FORTRAN en la PDP-11. Es también claro que las computadoras de 8 bits no fueron lo suficientemente poderosas para el cálculo de los espectros de cálculos y análisis de un canal unitario, trabajos para los cuales las minicomputadoras eran empleadas en ese tiempo.

Sin embargo, el bajo costo de las máquinas inspiró muchos intentos para utilizar la nueva tecnología, no obstante sus limitaciones. Los intentos se hicieron para tratar de dividir las señales de adquisición y las tareas de procesamiento entre mas de un procesador de 8 bits. Cambridge Electronic Design tomó este acercamiento y produjo la CED 1401, una inteligente unidad de interface de laboratorio con su propio microprocesador (6502) y

memoria interna, además de convertidores A/D, D/A y cronómetros. Las tareas de procesar un arreglo de adquisición deberían ser ejecutadas más rápidas por el CED 1401 que por alguna computadora. Muchos usos se hicieron del CED 1401 en combinación con la Apple II o BBC Micro.

1.5 La Computadora Personal IBM y la familia Apple Macintosh

En los 80's una segunda generación de microcomputadoras empezó a aparecer basada en un microprocesador de 16 bits. Más significativamente, en 1981 IBM, el dominante supremo en la industria de la computación produjo la Computadora Personal IBM (IBM PC). Esta máquina cambiaba radicalmente la naturaleza de la industria de la computación, se basaba en el microprocesador Intel 8086 el cual tenía un código de instrucción en bits tan poderoso como la PDP-11 y una memoria de 1 Mbyte, en oposición a los 64 kbytes de la PDP-11. Tuvo un conjunto de celdas de expansión (slots) como la Apple II, y suministrada con un efectivo sistema operativo llamado MS-DOS (IBM lo llama PC-DOS) desarrollado por la entonces pequeña e independiente compañía de "software" Microsoft. Inusualmente IBM reveló las especificaciones técnicas detalladas para la PC permitiendo que otras compañías produjeran computadoras compatibles a la IBM PC y que fueran capaces de correr el mismo "software" y aceptar las mismas tarjetas de expansión. Estos factores combinados con la aprobación de IBM de el concepto de computadora personal, resultaron en la llegada de la más exitosa computadora alguna vez producida, la PC.

El desarrollo de interfaces de laboratorio con tarjetas de expansión para la PC permitió que esta fuera usada en el laboratorio. Un gran rango de tarjetas estuvieron disponibles, entre las que destacan Labmaster de Scientific Solutions, la DT2801A de Data Translation. El "software" específico diseñado para el análisis de señales electrofisiológicas comenzó a ser producido, teniendo un gran éxito el paquete pClamp. Aún cuando la IBM PC fue claramente una computadora adaptada al uso de laboratorio, no superaba a la PDP-11 en ejecuciones. Sin embargo, la computadora IBM PC AT (Advance Technology) basada en 6 Mhz del microprocesador Intel 80286, e introducida en el año de 1984 fue cuatro veces más rápida que la PC original y totalmente comparable con la PDP-11/23. Cuando mejores cualidades, menor precio y compiladores de lenguajes estuvieron disponibles para la familia de microcomputadoras IBM PC's, estas empezaron rápidamente a remplazar a la PDP-11 de los laboratorios de electrofisiología.

Desde entonces han tenido un constante mejoramiento en ejecución y capacidad de memoria. Una PC con 50 Mhz y microprocesador 80486 es al menos 100 veces más rápida que la original PC. Grandes avances se han hecho en el desarrollo de "software" de aplicación con grandes y sofisticados paquetes, cumpliendo casi siempre con las mismas características.

En este mismo periodo una segunda y significativa familia de computadoras apareció, la Apple Macintosh PC's, basadas en la serie de microprocesadores Motorola 6800. La Macintosh fue radicalmente diferente en concepto de la PC ya que usaba una Graphical User Interface (GUI) donde el usuario interactuaba con la computadora por medio de localización de objetos (íconos) o seleccionando desde menús por medio del mouse. La familia Macintosh provee notablemente este uso, atrayendo mucho a los usuarios

de computadoras quien inicialmente habían sido inhibidos por la complejidad y diversidad de los comandos requeridos para un uso convencional como lo era el sistema operativo MS-DOS.

Sorpresivamente, y no obstante el papel que jugaron la expansión de "slots" en el éxito de la primera serie de computadoras Apple II, la original Macintosh carecía de características de semejanza o compatibilidad. Por esta razón se fue dificultando la suma de convertidores A/D (tarjetas) esenciales para usarse en el laboratorio. Los Dispositivos tendrían que haberse producido para ser usados por el puerto serial de la Macintosh, pero estuvieron limitadas por la baja rapidez de sus canales de comunicación. La serie de computadoras Macintosh II remedio esto, fue creada con un conjunto más convencional de celdas de expansión, llamados los Nu-Bus. Desde entonces ha estado disponible un gran número de tarjetas de interface. Sin embargo, aun cuando la Macintosh se estableció en el área de análisis de imágenes (donde la GUI es probablemente una ventaja) tuvo mucho menor uso para el análisis de señales, en lugares como en el laboratorio de electrofisiología.

1.6 "Software" y Métodos de Análisis

En el presente la familia de computadoras IBM PC y Macintosh constituyen la mayoría de las computadoras usadas en el laboratorio. La gran disponibilidad de tarjetas de expansión compatibles ha dado como resultado un predominio de las PC's en el campo de la adquisición de datos en electrofisiología. Mas programas de análisis electrofisiológico se han desarrollado para las PC's usando el sistema operativo MS-DOS, y el índice de paquetes disponibles para éstas, excede lo que estuvo disponible para la PDP-11. El "software" ha continuado desarrollándose por algunos laboratorios, sin embargo ha habido una gran tendencia hacia su distribución comercial más que hacia el libre cambio. El original pClamp producido por CalTech ha sido aumentado substancialmente y es ahora producido por Axon Instruments. De forma similar mas interfaces de laboratorio son reemplazadas por programas de adquisición de datos. Con el desarrollo para la IBM PC de una versión de Axon Instruments AXOBASIC, muchos de los programas originalmente producidos para la PDP-11 pueden ahora ser ejecutados en una PC.

Las computadoras de laboratorio han alcanzado ahora un estado de madurez, donde ni el costo ni el desarrollo del "hardware" son una limitación significativa para su aplicación en el laboratorio. Comparada con la era PDP-11, el sistema de análisis por computadora es ahora probablemente el componente más caro de todo el laboratorio electrofisiológico; costando tanto como un microscopio. Se pueden hacer lecturas no caras disponiendo de convertidores A/D pudiendo fácilmente digitalizar señales que muestran rangos de 100 Khz o mas. Sin embargo, como es verdad en otras áreas de computación, es claro que el desarrollo de "software" apropiado y confiable es el principal impedimento para progresar en esta área.

Como la computadora viene a ser el medio normal de análisis de datos experimentales, es necesario entonces conocer el procedimiento involucrado, saber cuales son sus ventajas y limitaciones y razones para su uso. En tiempos pasados, cuando el autor del programa de adquisición era únicamente el usuario del mismo, se tenía una visión individualizada de cada proceso y se debía a la consecuencia automática de escribir el programa. Sin embargo el incremento del "software" especializado en laboratorios ha dado

como resultado su compra como un producto comercial u obteniéndolo como un obsequio. Los inconvenientes que ahora se presentan en desarrollar "software", es que la mayoría presentan demasiadas características similares, pareciendo que tan sólo se está "re-escribiendo".

La poca publicación de métodos de adquisición basados en computadoras presenta una gran dificultad para la estandarización de las adquisiciones. Asimismo la presentación de principios generales sería de gran uso, pero mucho del valor de los métodos de análisis por computadora radican en la implementación de sus programas, en donde el código fuente de cada programa es complejo, muy largo, y con características específicas para una computadora particular o laboratorio particular, por lo que no son particularmente convenientes para publicación vía los canales normales como son artículos diarios. Así muchos aspectos de la tecnología de computadoras no satisfacen totalmente las necesidades en la experimentación científica, pero es bueno dar algunos principios generales para el desarrollo de un sistema de adquisición con la tecnología ya establecida en un laboratorio y tratando así que la única dificultad que se pudiera presentar, fuera la de implementar los programas de adquisición, teniendo ya los algoritmos básicos para realizar dicha tarea. A su vez la extraordinaria rapidez de evolución de la computadora, desde la "mainframe" hasta la microcomputadora, ha tendido ha reducir el tiempo de vida de muchos de los mejores textos en el tema, aunque estos sean muy pocos. Esto deberá dar nacimiento a una nueva técnica que se aproveche y no sea obsoleta después de algunos años de publicación. Lo que en el futuro se prevee es el uso de una computadora con un nuevo sistema operativo complejo y multiusuario. Sin embargo en los últimos 10 años el énfasis ha sido en una computadora para experimentar, y con un sistema operativo uni-usuario.

Comparado con la vasta literatura en el uso de las computadoras personales para su aplicación en los negocios, hay relativamente pocos textos discutiendo los temas acerca de la computadora en el laboratorio, particularmente el registro y análisis de datos analógicos.

2. REGISTRO DIGITAL DE SEÑALES ANALÓGICAS

El tema principal de esta tesis es el desarrollo apropiado de un sistema para la conversión de señales analógicas de voltaje derivadas de registros electrofisiológicos en una forma digital, convenientes para almacenarse, analizarse, procesarse y graficarse por medio de la computadora. Debido a esto es conveniente discutir acerca de los principios de conversiones analógicas a digitales y como pueden ser aplicadas a el registro de señales electrofisiológicas, por lo que es inevitable hablar de ciertos aspectos del "hardware" y "software" de la computadora. Mucho de el desarrollo de la adquisición de datos en el laboratorio en los años recientes se debe a las PC's, y existen muchas interfaces de laboratorio disponibles, por lo que ciertos detalles en el diseño de las PC's las hacen eminentemente las más aptas para los trabajos de adquisición de datos.

2.1 Digitalización de Señales Analógicas

Una característica distintiva de la computadora digital es que manipula números enteros (i.e. ...,-2,-1,0,1,2,...) y solamente estos números pueden ser manejados en su

memoria. Las señales analógicas, son producidas por aparatos de registro electrofisiológico, los cuales son distintos a las computadoras porque manejan señales continuas, por lo que una copia completa de una señal analógica no puede ser representada con una computadora digital. Sin embargo una aproximación digitalizada a la señal analógica puede ser hecha por repetidos muestreos del nivel de voltaje de la señal a intervalos fijos de tiempo, y asignando a esta amplitud el valor entero más próximo que la computadora pudo tomar. La señal analógica continua, es así cortada y representada en la computadora como una serie de números enteros.

La calidad de la aproximación digital depende de la fineza tanto del tamaño del intervalo entre muestras como del nivel entero usado para representar la amplitud, por lo que es de suma importancia coordinar perfectamente las velocidades de muestreo (MHz o kHz) con el numero de muestras que se deben adquirir, para lograr así una mejor representación de la señal evitando la perdida de información importante de la señal analógica. *(fig. 1).*

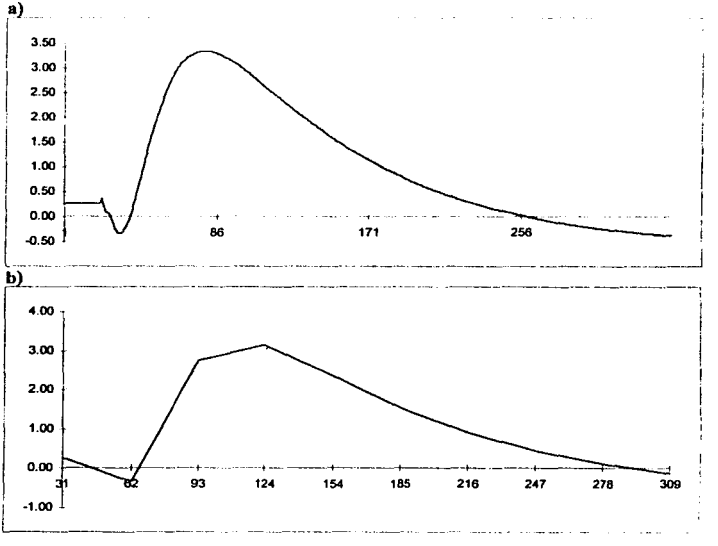


fig. 1. **a)** Registro tomado con 1000 muestras por segundo y **b)** Registro tomado con 30 muestras por segundo. Se puede notar que el registro b), presenta segmentos de rectas debido al mayor tamaño del intervalo de muestreo.

2.2 El Convertidor Analógico-Digital

El repetido muestreo de señales analógicas de voltaje es realizado por un dispositivo conocido como convertidor analógico a digital (ADC, por sus siglas en ingles). Dicho ADC es un voltímetro capaz de medir rápidamente la señal analógica presentada como entrada y asignándole un número proporcional a dicha señal, el cual puede ser leído y almacenado en una memoria de computadora. La resolución del ADC es especificado como el número de bits usados para representar el voltaje analógico. Ocho, 12, y 16 bits de resolución, producen mediciones con números enteros en el rango de 0-255, 0-4095 y 0-65535. La amplitud en el rango de éstos números, determina lo fino o exacto de la medición. Los ADC's son clasificados en términos de su resolución y el máximo índice al cual pueden medir la señal analógica (velocidad de muestreo).

Debe de considerarse la resolución y velocidad máxima de muestreo cuando se esta eligiendo un ADC conveniente para digitalización de señales electrofisiológicas. Una amplia variedad de ADC's están disponibles cumpliendo con amplia resolución, alta velocidad o bajo costo. El ADC debe ser capaz de muestrear a grandes velocidades, suficientes para poder seguir el tiempo en curso de la señal analógica. Algunas señales electrofisiológicas como son el voltaje creado por corrientes de Na^+ tienen particularmente una rapidez de tiempo en curso de 1-2 ms y requiere de una gran velocidad del ADC para digitalización.

Mientras que los ADC's de 8 bits son baratos y pueden tener una gran velocidad de muestreo, un porcentaje de resolución de solo 0.4 % no es suficiente para propósitos de investigación. Un ADC de 16 bits provee una mejor resolución, una gran velocidad y un costo accesible. En suma los problemas de ruido en el típico sistema de mediciones electofisiológicas son remotos al ser suficientemente bajas con el uso de 16 bits de resolución. Por esto mismo con un rango de 10 volts de entrada, los niveles de ruido deberán de tener menos de 300 microvolts haciendo uso de todos los 16 bits. Un ADC común y corriente de 12 bits provee una adecuada resolución para separar la señal del ruido común de los experimentos electrofisológicos, su velocidad de muestreo pude ser tan rápida que es capaz de igualarse a la señal electrofisiológica más veloz, y su costo no es excesivo.

2.3 La Interface de Laboratorio y su Programación

Un ADC con gran rapidez y gran resolución es un dispositivo que presenta gran dificultad para construirse, y se obtienen actualmente como circuitos integrados de precisión fabricados por abastecedores especializados. Además, se requieren circuitos auxiliares de soporte para permitir controlar al dispositivo desde la computadora. El sistema completo es conocido como una *Interface de Laboratorio*.

En suma a el ADC, la interface de laboratorio proporciona :

- Gran rapidez del reloj(es) para una coordinación precisa.
- Comunicaciones de la computadora al ADC.
- Múltiple entrada analógica de canales.
- Conversiones Digitales a Analógicas (DAC, salidas).
- Pulso digital de entrada y salida.

La interface toma la forma de una tarjeta de circuitos (alguna veces conectadas a módulos adicionales externos), la cual se inserta en un "slot" de expansión en la computadora. Muchas de las computadoras usadas en el laboratorio tienen un conjunto de celdas de expansión semejantes, diseñadas para permitir a las computadoras el tener la facilidad de ser extendida debido a la suma de tarjetas de circuitos especializados.

En forma general, la interface de laboratorio realiza la función de modo tal que al correr un programa en la unidad central de proceso (CPU) de la PC, ésta pueda enviar códigos de control y leer datos desde la interface de laboratorio vía las celdas de expansión de la computadora. El reloj de muestreo puede ser programado desde la computadora para enviar un pulso a intervalos de tiempo precisos para el ADC, instruyéndolo para muestrear la señal analógica de entrada.

Cuando la conversión se ha realizado, el número digital es transferido a la computadora y almacenado en algún lugar en su memoria. Si más de un canal analógico va a ser muestreado, entonces la computadora deberá también instruir al multiplexor, para poner un interruptor al canal apropiado a través de la entrada del ADC.

La comunicación entre una computadora y la interface de laboratorio se realiza vía un conjunto de puertos de entrada/salida (I/O) suministrados por la tarjeta de interface. Desde el punto de vista del programador, los puertos de I/O actúan de una manera similar a la memoria de la computadora, excepto que el dato escrito en ella es transferido por la interface a la memoria RAM. De manera similar, los datos pueden ser transferidos desde la interface a la PC por lecturas del puerto I/O. Los puertos I/O tienen únicamente números de direcciones, similar a la memoria direccionable y códigos especiales de instrucción de máquina para lectura y escritura de ellos. Los puertos de I/O en la interface de laboratorio pueden ser clasificados en las siguientes categorías, aún cuando el número exacto de puertos y detalle de sus funciones pueden variar considerablemente entre fabricantes:

- Puertos de Control de Escritura

  Mandan a la interface a realizar alguna acción, es decir sirve para inicializar una conversión A/D, y arrancan o paran el reloj de muestreo.
- Puertos de Estado de Lectura

  Determinan el resultado de un comando previo, es decir sirve para determinar si una conversión A/D ha sido realizada, o si ha ocurrido un error.
- Puertos de Datos

  Se usan para transferir valores de datos entre la computadora y la interface, es decir, leer para obtener el resultado de una conversión A/D.

2.4 Interface de Laboratorio para Transferir Datos a la Computadora

Programar una interface de laboratorio para digitalizar señales analógicas consiste de 3 funciones distintas. En la fase de inicialización, a la interface se le prepara para realizar un número requerido de conversiones A/D, se le dan los índices de muestreo, y los canales de entrada. Una vez empezando a muestrear, la segunda fase maneja la transferencia de datos de las muestras tomadas por el convertidor A/D a la memoria de la computadora. Finalmente después de adquirir el número de muestras que han sido seleccionadas, la interface de laboratorio entra en la fase de terminación en la cual se regresa a la interface a

su estado primero. Aún cuando los detalles difieren entre interfaces, la inicialización y la terminación son usualmente un proceso directo de poner los bits apropiados en los puertos de comandos de la interface. La transferencia de datos requiere especial atención, particularmente cuando se muestrea a grandes velocidades. En general no es menor la interface que tiene la habilidad de almacenar muestras A/D internamente, una muestra tomada por el ADC debe ser transferida y almacenada en la memoria de la computadora antes de que ocurra la siguiente conversión. Esto es un tarea fácil cuando se muestrea a índices de 1 kHz o menos, pero se llega a incrementar la dificultad a grandes índices de muestreo. De hecho, esto usualmente es el índice al cual las muestras A/D pueden ser transferidas desde la interface a la memoria de la computadora, lo que limita el índice máximo de muestras tomadas, cuya velocidad de muestreo deberá ser menor que el tiempo de conversión de el ADC mismo. Existen cuatro métodos comúnmente usados para transferir datos, pero no todas las interfaces de laboratorio tienen "hardware" disponible para soportar todos los métodos, estos son los siguientes:

- Programando
- Manejando interrupciones
- Acceso directo a memoria
- Memoria compartida

• Programando transferencia de datos.

También es conocido como programando I/O, requiere de lo mínimo en términos de sofisticación de "software" para la interface de laboratorio. Este método puede ser usado con algunas interfaces a muy bajo costo, es relativamente simple, involucra requerimientos a la interface para ejecutar una conversión A/D esperando hasta que esta sea realizada, entonces se lee la muestra del puerto de datos de la interface y se almacena en memoria. En términos generales, el programa implementa el siguiente algoritmo:

a) Inicializar la tarjeta de interface
b) Programar el reloj de muestreo
c) Esperar hasta que una muestra A/D este disponible
d) Copiar la muestra desde el puerto de datos del ADC hasta un arreglo de almacenamiento
e) Repetir c) y d) hasta que todas las muestras sean coleccionadas
f) Parar el reloj de muestreo

Un problema fuerte al programar la transferencia de datos, es que el programa debe monitorear constantemente el estado del ADC para verificar si una nueva muestra está disponible. Consecuentemente, cuando registramos esto en un proceso, es difícil poder usar la computadora para otras operaciones en caso de que las muestras estén mezcladas. Esto es importante porque para muchas aplicaciones es esencial poder ejecutar otras funciones mientras se está realizando la adquisición, como son el escribir datos a disco, responder a las teclas presionadas en el teclado, o actualizar el desplegado en pantalla. En suma el método de programar la transferencia de datos es fácil de implementar y entender, y se puede usar con las más simples interfaces de laboratorio. Sin embargo, para alcanzar grandes índices de muestreo requeridos para trabajos electrofisiológicos, el programa de muestreo A/D debe ser meticulosamente escrito en lenguaje ensamblador. Por otra parte no es posible muestrear continuamente a grandes velocidades por largos periodos, ya que no hay tiempo para la computadora de poder transferir muestras a disco una vez que la memoria RAM se ha llenado.

- Transferencia de datos manejando interrupciones.

Para mejorar un programa de transferencia de datos, es necesario incrementar el monitoreo constante al estado del ADC. Se requiere entonces un mecanismo que permita a la interface de laboratorio registrar los servicios de la computadora cuando una conversión se ha realizado, indistintamente de que programa está actualmente corriendo en el tiempo.

En las microcomputadoras modernas, una característica conocida como el interruptor de "hardware" ha sido introducida para resolver este problema. Este es un método por el cual un dispositivo periférico puede señalar al CPU de la computadora que requiere servicio. El programa que está siendo ejecutado es suspendido temporalmente y el control se transfiere al código apropiado para tratar con lo que necesita el dispositivo periférico. Las interrupciones del "hardware" son implementadas en la familia de PC's usando el INTEL 8259A (interruptor programable y controlador), el cual soporta 8 líneas independientes de interrupción (IRQ0,...,IRQ7) en la expansión del bus de PC's. Estas líneas pueden ser conectadas al circuito de un dispositivo periférico que es usado para invocar alguna de las 8 diferentes interrupciones de servicio de rutinas para programas.
Las líneas de interrupción son puestas en uso con IRQ0 siendo usada para soportar el reloj de tiempo del MS-DOS, IRQ1 el teclado, y otras asignadas a manejar puertos de comunicación en serie y en paralelo.

Usando interrupciones para transferir muestras A/D a la memoria de la computadora, el "hardware" de la interface de laboratorio debe ser ligado a la línea IRQ para poner un "jumper" o "switch" localizado en la tarjeta de interface. El siguiente servicio de interrupción por rutina debe ser llamado para que pueda tomar la acción apropiada cuando una conversión A/D se ha realizado. Desafortunadamente hay un precio a pagar por usar interrupciones, cuando una interrupción A/D ocurre, la rutina de servicio de interrupción debe preservar el estado del programa ejecutado actualmente, así que este debe ser realmacenado.

- Transferencia de datos vía Acceso Directo a Memoria (DMA)

Parte del problema con las técnicas anteriores (programando y manejando interrupciones) es que la transferencia de datos es un proceso de 2 fases, primero una transferencia desde el ADC a un registro del CPU y una segunda transferencia desde el registro del CPU a la memoria. El uso de algunos métodos de transferencia que requieren el uso del CPU arrojan una pérdida relativa de tiempo, el cual es ocupado para ejecutar una serie de instrucciones, tomando cada una de 1 a 2 milisegundos. En la memoria RAM de la computadora puede leerse o escribirse un dato en un tiempo de entre 0.1-0.2 milisegundos. El método de acceso directo a memoria (DMA) evita ese problema, por transferir directamente los datos desde un dispositivo periférico a la memoria de la computadora. La transferencia de datos DMA, sin embargo, requiere "hardware" especializado para usarse con la computadora en forma de un dispositivo controlador de DMA capaz de manejar la transferencia de datos en lugar del CPU.

La familia de computadoras PC's son apropiadas para hacer uso de la transferencia de datos DMA, ya que poseen el "hardware" necesario y construido con características estandarizada. Todas las PC's tienen al menos un controlador INTEL 8237A programable de DMA y un microprocesador INTEL 80286 - 80386-80486. Cada 8237A provee 4

canales independientes de transferencia de datos DMA, y los dispositivos periféricos requieren el servicio del controlador DMA. El procedimiento para transferir una muestra A/D desde la interface de laboratorio a la memoria de la computadora usando DMA es el siguiente:

a) El reloj de muestreo de la interface es programado para inicializar conversiones y fijar intervalos automáticamente.

b) Sobre la terminación de una conversión A/D, la interface pide al controlador DMA una línea DREQ1 (línea de registro DMA) para un canal DMA.

c) El controlador DMA anota el requerimiento DMA y el CPU es puesto en un estado de captación, las señales pasan a la interfaces vía la línea DACK1 que está lista para recibir datos.

d) Cuando la interface recibe la señal de la línea DACK1, ésta pone el primer byte de la muestra A/D en la interface del bus. Entonces el controlador DMA maneja el proceso copiando el byte en la memoria RAM. Si un segundo byte va ha ser transferido (como es usual para muestras A/D de 12 bits), la interface de laboratorio pone nuevamente el DACK1 en línea y una segunda secuencia de transferencia (c,d) toma lugar en la siguiente localización de memoria.

e) Entonces el controlador DMA libera al CPU, el cual recupera el control del bus de la interface y continua sus operaciones.

En todos los pasos [(a);(e)] el procedimiento de transferencia toma lugar completamente en el hardware, debido a la ejecución de instrucciones del CPU. Dichos pasos son ejecutados rápidamente y pueden soportar un máximo de transferencias de datos de 500 kbytes s-1 en una típica computadora de la familia IBM-PC. Asimismo, no requieren de instrucciones del CPU o usar algunos de sus registros de datos, por lo que no hay grandes procesos involucrados en preservar el estado del programa antes y después de transferir datos. De la misma manera que la interrupción de líneas, algunos de los canales DMA ya son usados por la PC como parte de esta operación normal, el canal DMA 0 es una parte integral del sistema de memoria RAM en PC's y el canal 2 es usado para transferir datos de y desde el manejador del disco. El canal 1 de DMA, sin embargo, está usualmente libre para usarse por unidades en la interface y también el canal 3 de DMA en los procesadores 80286 y 80386.

Programar para transferir datos DMA requiere de un entendimiento detallado de las funciones de el controlador 8237A de DMA. Las ventajas de usar transferencia de datos vía DMA claramente importan más que las dificultades involucradas en programar el controlador DMA. En comparación con el método de manejo de interrupciones, la transferencia vía DMA es mucho más eficiente.

- Memoria Compartida

El método DMA está limitado por la velocidad de transferir datos a través de la expansión del bus de la computadora, la cual en las entradas estandarizadas de las PC's es de alrededor de 250 kHz. Para alcanzar rangos de este orden se puede instalar una forma especial de memoria RAM, conocida como puerto dual de la RAM (dual-port), la cual como su nombre lo sugiere puede leer o escribir datos desde dos fuentes separadas. Una parte es conectada en forma normal a la computadora, mientras la otra es directamente conectada a la interface de laboratorio. La interface puede también escribir directamente en

esta memoria. Esta técnica es conocida como Memoria Compartida (Shared-Memory), y muestrea a velocidades tan grandes como 1 Mhz, y puede llevarse a cabo con un "bus" estándar de computadora.

2.5 Muestreo Continuo a Disco

El hecho de que el método de transferencia de datos vía DMA deja al CPU de la computadora libre para realizar otras tareas, significa que podemos tomar esto como la fundación del método de registro digital requerido para señales electrofisiológicas. Probablemente el método más importante es el muestreo continuo a disco. Se obtiene de una necesidad de adquirir muy largas secuencias de registro con altos índices de muestreo. Según cada aplicación, este método será necesario para sostener índices de muestreo de alrededor de 20 kHz por algunos minutos, y coleccionar algunos millones de muestras. Sin embargo, registros demasiado largos exceden la capacidad disponible de la típica computadora en su sistema de memoria RAM (de 640 kbytes a 8 Mbytes). El método de muestreo continuo a disco, provee un medio de escritura de muestras A/D de memoria RAM a disco antes de que el límite de la memoria RAM sea excedido, mientras el muestreo A/D está silenciosamente en progreso.

El índice máximo de muestreo continuo a disco se ve limitado por el número de muestras que pueden ser transferidas a disco, el cual depende particularmente de cómo se ha programado el procedimiento que realiza esa tarea, la velocidad con que se ejecuta la instrucción de la computadora, y la rapidez de el manejador de disco (drive) para grabar. La típica PC usada actualmente en el laboratorio, puede realizar fácilmente muestreos a disco de alrededor de 30 kHz, usando un procedimiento estándar del MS-DOS para escribir datos a archivo. Índices de 60-70 kHz pueden ser alcanzados usando grandes rutinas, especiales y eficientes para escribir a disco, y velocidades tales como 250 kHz pueden alcanzarse usando una computadora (25 Mhz en una 80386) y teniendo una gran ejecución del SCSI (Small Computer Systems Interface) manejando el disco duro.

2.6 Detección y Registro de Señales Espontáneas

El muestreo continuo a disco nos permite hacer un registro digital completo de las señales bajo estudio. Mientras ésta es una facilidad extremadamente valiosa, puede también ser muy ineficiente cuando se aplica al registro de señales de pequeña duración, ocurriendo solamente a intervalos de baja frecuencia. Por lo mismo, un típico experimento de señales neuromusculares puede involucrar el registro de series de corriente en miniatura, la cual solamente dura de 2 a 3 milisegundos, y ocurre espontáneamente una ó dos veces por segundo. Esto hace necesario coleccionar un largo número de señales para análisis estadístico. Para evitar que este 99% de tiempo se pierda y también espacio en disco, se obtiene preferentemente sólo una colección de muestras A/D cuando una señal ya es prevista. Esto es un proceso relativamente simple cuando las señales bajo estudio son evocadas por un estímulo externo, así el registro puede ser asociado a un pulso externo

2.7 Algunas Interfaces de Laboratorio

Las interfaces de laboratorio comercialmente disponibles varían tanto en características, como en ejecución y costo. Es por lo tanto de gran valor especificar cuáles son los requerimientos para una interface de laboratorio de propósitos generales eficiente. Idealmente debería tener las siguientes especificaciones:

- 4 ó más canales de entrada (muestreando a 100 kHz)
- Selección automática de canal para registros multi-canal
- 2 ó más canales de salida analógica
- 2 ó más relojes con capacidad de disparo externo
- Transferir datos DMA
- Capacidad de muestreo simultáneo A/D y salida D/A

De las interfaces que se encuentran comercialmente disponibles actualmente para el laboratorio, solamente algunas cuentan con las características especificadas, y las otras son deficientes en al menos una de ellas, por lo que no son satisfactorias para muchas rutinas de aplicación.

Mientras la discusión se enfoca hacia las características y desempeño provistas por la interface de laboratorio (hardware), la disponibilidad de "software" apropiado es probablemente más importante. El desarrollo de "software" para controlar y coleccionar datos desde una interface de laboratorio es difícil y consume tiempo. Mientras mas distribuidores ofrecen algunas bibliotecas y subrutinas para controlar su interface, esta puede no proveer todas las funciones necesarias, ya mencionadas. Por esto, si no se tiene el tiempo o a un experto para realizar un extenso programa es esencial primero obtener un paquete de "software" apropiado, y entonces seleccionar una interface compatible con él.

Al menos un paquete de análisis de datos electrofisiológicos está disponible para cada una de las interfaces más conocida como son:

a) Scientific Solutions Labmaster

Es probablemente la más comúnmente usada como interface de laboratorio. Ha estado disponible al menos por 8 años para PC's y fue una de las primeras en tener un "software" para un uso electrofisiológico.

b) Cambridge Electronic Design (CED) 1401

Tiene una larga historia suministrando para la minicomputadora PDP-11 sistemas para laboratorios electrofisiológicos. Este es un dispositivo más largo y más sofisticado que el Labmaster, con su propio microprocesador programable en la tarjeta. Puede ocuparse no solamente con la familia de las computadoras compatibles IBM-PC's, si no también con las Apple Macintosh, Acorn Archimedes, BBC Micro y algunas DEC VAX.

c) Data Translation Interfaces

Es la más grande, suministrando interfaces de laboratorio. Producen un amplio rango de tarjetas a bajo costo, con soporte para dispositivos con índices de muestreo de pocos kilohertz, hasta dispositivos con 750 kHz, poniéndose en los límites de ejecución que puede soportar una PC.

d) National Instruments Lab-PC

Es otro gran suministrador de tarjetas de interface para PC's. Son conocidos como los primeros suministradores de la tarjeta de interface IEEE-488 para PC's y algunas otras,

siendo al menos la industria estándar en esta área. IEEE-488 es una especificación de la interface la cual permite el uso de dispositivos de medición como son osciloscopios digitales y contadores de frecuencias o multímetros, para comunicarse y ser controladas por una computadora.

e) Otras Interfaces de Laboratorio

Las interfaces mencionadas anteriormente son sólo las mejor conocidas de muchas disponibles, sin embargo hay algunas otras que también son usadas en el laboratorio electrofisiológico como son, LC Electronic's, Computerscope-Phy System, el Intracel-5200, Keithley, Instrulech M2 System etc..

3. CONDICIONANDO LA SEÑAL ANALÓGICA

En el tema anterior, tomamos como supuesto que las señales de voltaje producidas por el sistema de registro electrofisiológico, estaban apropiadamente conectadas a la entrada de la interface de laboratorio. Este no es siempre el caso, y existe la necesidad de adecuar la señal de voltaje para hacer un buen registro, o hacer un registro de todo el evento. El condicionamiento de una señal puede involucrar amplificación del nivel de la señal, remover los niveles de DC, filtración de la señal, y la detección y procesamiento de pulsos de sincronía.

Muchas señales intracelulares ocurren solamente durante una fracción de segundo y requieren alguna forma de estimulación para iniciarlas, un típico ejemplo es el nervio estimulado en potenciales de placa terminal. Es necesario entonces sincronizar un registro digital con el estímulo del evento, usando la sincronización de salida del estimulador al disparo de muestreo A/D.

3.1 Amplificación de la Señal

La amplificación de la señal se obtiene necesariamente porque el convertidor A/D en la interface de laboratorio tiene un voltaje de sensibilidad relativamente bajo comparado con el de dispositivos más comunes como es el osciloscopio. Algunos ADC's tienen solamente rangos fijos de entrada (+-5V para el ced 1401) ó un rango programable limitado (+-10V,+-5V, +-2.5V,+-1.25V de Data Translation DT2801A). Algunos osciloscopios de bajo costo usualmente tienen un rango de 10V ó también una sensibilidad de entrada de 5mV/div (+-100 mV) a 5V/div (+-50V). Esta carencia de sensibilidad puede crear problemas cuando trabajamos con instrumentos para electrofisiología, los cuales tienen restricciones en niveles de salida. Los Amplificadores de microelectrodos en primera instancia pueden amplificar la señal de voltaje de la medición celular hasta por un factor de diez. Mientras la sensibilidad de voltaje del osciloscopio usualmente puede ser ajustada a la conexión del nivel de la señal que se provee, esto no se puede hacer siempre para la interface de laboratorio. El problema se hace particularmente agudo cuando estudiamos señales pequeñas tal como los potenciales miniaturas de placa terminal (MEPP), con amplitudes de solamente 1-2 milivolts. Desde un ADC de 12 bits con un rango de entrada

de +-1.25 V se tiene una sensibilidad de 0.01 mV/bit, el digitalizador MEPP amplificará solamente un poco la señal.

Para hacer efectivo el uso de 12 bits de resolución del ADC, las señales pequeñas deberán ser amplificadas para alcanzar ordenes de magnitud tan largas como la sensibilidad del ADC. En el caso del MEPP, para alcanzar un 0.1 % de resolución, la señal debe ser amplificada 1000 veces, atando 25% del rango de entrada de voltaje del ADC. Para hacer versátil y permitir conexiones para una gran variedad de tipos de diferentes señales, el amplificador deberá tener una ganancia variable o factor de amplificación entre 1 y 1000. La ganancia se puede variar continuamente usando un potenciómetro de precisión ó interrupciones entre una serie de niveles constantes como en un osciloscopio.

3.2 Removiendo un Nivel DC

Esto se hace necesario solamente, para remover algún nivel constante de DC que pudiera existir en la señal analógica antes de que la amplificación tome lugar. Esto puede hacerse usando un amplificador diferencial. Un amplificador diferencial, como su nombre lo sugiere, amplifica la diferencia entre 2 señales ((+) y (-)). Los niveles DC son substraídos de la señal alimentando un nivel de voltaje constante del potenciómetro en la entrada (-), la cual es substraída de la señal desde el microelectrodo amplificado alimentando en la entrada (+), pero antes siendo multiplicada por el amplificador. Este método es también conocido como entrada "DC Offset".

3.3 Filtrando Señales

Acorde a la teoría de Fourier una señal analógica puede ser descrita como la suma de una serie de componentes de ondas sinoidales de varias amplitudes, extendidas sobre un rango de frecuencias. Las propiedades de la señal que corresponden a componentes de baja frecuencia, cambian lentamente, y aquellas que contienen componentes de alta frecuencia, cambian rápidamente. Filtrando los procedimientos selectivamente, removemos componentes de frecuencias particulares y por medio de eso modificamos el tiempo en curso de la señal analógica. Con filtros "pasa-bajas" que dejan pasar bajas frecuencias se remueven componentes de señales de alta frecuencia, sobre todo un filtro definido como un cortador de frecuencias suaviza la señal. A la inversa con filtros "pasa-altas" que permite el paso de altas frecuencias filtra y remueve el DC y las bajas frecuencias, eliminando lo transitivo y cambiando rápidamente partes de la señal. La combinación de los dos es el filtrado ancho de banda. El filtrado más generalmente aplicado como un proceso de condicionamiento de una señal es el de una variedad del filtrado a "pasa-bajas", y esto se hace por dos razones:

- Anti-aliasing .- filtro para eliminar artificios en la señal inducidos por el proceso de muestreo digital.
- Smoothing .- suavizador de la señal removiendo componentes de altas frecuencias de ruido de fondo para no proveer de una gran proporción de señales de ruido.

3.4 Señales Fantasmas (Anti-aliasing)

Mientras una señal analógica es una cantidad continua, el registro digital de la misma señal es restringida a una serie de muestras tomadas a intervalos fijos. Si el intervalo de muestreo no es pequeño, comparado con el tiempo en curso de la señal, la versión digital no será una representación fiel de la señal original. Este problema es particularmente agudo en el caso de señales periódicas de alta frecuencia, produciéndose un registro digital engañoso.

Cuando muestreamos a índices que son relativamente altos para el tiempo en curso de una señal, no se presenta este problema. Pero es útil saber qué tan alta debe ser la frecuencia de muestreo, para evitar que se generen señales deformadas o inexistentes (señales fantasmas). Para obtener esta información se emplea el teorema de Nyquist, que establece el índice mínimo de tiempo de muestreo dado por:

$$fnyq = 2fmax$$

donde fmax es el componente más alto de las frecuencias presentes en la señal en estudio. Por otra parte, dado que las frecuencias que están por encima del límite Nyquist en un sistema de registro digital pueden distorsionar el resto de la señal, es necesario filtrarlas antes de ser digitalizada la señal, a este proceso se le conoce como filtrado "anti-alias". Esto hace a la digitalización del registro más presentable cuando se despliega en la pantalla de gráficas de las PC's.

3.5 Suavizado de señales y ruido (Smoothing)

Las señales electrofisiológicas son generalmente registradas en la presencia de ruido de fondo, inherente a los procedimientos de registro. Tal ruido (definido aquí como señales no deseadas que oscurecen la señal de interés) puede ser dividido en 2 tipos principales:

- Interferencia de fuentes de señal externa, como son ondas electromagnéticas desde 50/60 Hz AC de principales líneas de poder, o el prendido o apagado de dispositivos electrónicos.
- Ruido aleatorio desde fuentes tales como el ruido Johnson en microelectrodos de alta resistencia o plataformas de entrada del amplificador.

Las señales de interferencia pueden ser removidas completamente por el uso de campos eléctricos de los aparatos de registro y una apropiada atención a la tierra eléctrica de las conexiones de los aparatos.

Por otro lado el ruido aleatorio no puede ser removido fácilmente debido a que es inherente a los aparatos de registro. Una de las fuentes de ruidos aleatorios más comúnmente encontrado es el ruido Johnson, el cual es una propiedad de todos los conductores eléctricos. Se manifiesta como fluctuaciones aleatorias en mediciones de voltaje debido a la terminal que induce movimientos de los transportadores de carga, electrones en un metal conductor o iones en una solución salina. El ruido Johnson es una forma de ruido blanco, y como tal, estas fluctuaciones aleatorias son extendidas uniformemente sobre un rango infinito de frecuencias. La magnitud de este ruido es

proporcional a la onda de frecuencias de los aparatos de registro y también a la resistencia del conductor, dada por la ecuación:

$$Vj=\sqrt{4kTRfc}$$

donde k es la constante de Boltzmann ($1.38x10^{-23}$ J/K), T es la temperatura absoluta (K), R es la resistencia (Ω), y fc es el límite de frecuencias más altas (Hz) de la banda de registro, puesta por el corte de frecuencias (cutt-off) de la señal de paso más baja condicionada al filtro. El ruido Johnson produce el factor que limita la mínima señal registrable usando microelectrodos intracelulares. Esto es un problema particular cuando tratamos con pequeñas células, las cuales requieren puntos muy finos, y por lo tanto gran resistencia en los microelectrodos.

La ecuación también se usa cuando registramos señales de pequeña amplitud usando un filtro de pasa-bajas (low-pass) para remover las grandes frecuencias del ruido aleatorio tanto como sea posible, sin afectar las señales de interés. Esto puede ser particularmente importante cuando registramos señales de corriente desde algunos sistemas de captación de voltaje, donde ahí deben ser substancialmente atrapadas las grandes frecuencias de ruido de fondo. Sin embargo si las componentes de frecuencias de la señal se extiende más allá del corte del filtro, entonces algunos de estos componentes son removidos también, resultando en alguna distorsión de la señal en curso. Una selección óptima del filtro fc es importante también para minimizar el ruido de fondo sin inducir señales de distorsión significativas

3.6 Detección del Evento y Disparo (Sincronía)

Como se ha discutido anteriormente, el inicio del registro digital debe obtenerse sincronizándolo con un evento externo, como la estimulación de un nervio o un comando de cambio de voltaje. Algunas interfaces de laboratorio proveen un medio para tal sincronización, usualmente por ligar el comienzo de el reloj de muestreo A/D a un pulso de entrada llamado disparo externo (external trigger). Esta entrada es usualmente diseñada para responder a un pulso estándar TTL (Transistor-Transistor Logic). TTL es la más común en un gran número de interfaces estandarizadas, permitiendo la interconexión de circuitos lógicos digitales. Las señales TTL son definidas nominalmente como 2 niveles, Low=0V (<1.4V) y High=5V (>3.5V). Generalmente una transición TTL, que consiste en cambiar de un nivel específico a otro (High a Low o Low a High) es la señal de disparo. La transición High a Low es la que se usa con mayor frecuencia, ya que provee mejor inmunidad al ruido accidental que podría disparar el sistema.

Para asegurar que la fase inicial de un evento sea registrada, el pulso de sincronía debe presentarse antes del pulso principal de estimulación. Muchos estimuladores pueden proveer también un pulso de sincronía, pero este no es siempre compatible eléctricamente con la entrada de la interface de laboratorio TTL. Ocurre un problema cuando un pulso de sincronía ha sido grabado en una cinta FM, la cual obtiene un rango de voltaje de salida de no mas de +/- 1V, demasiado pequeño para disparar directamente el circuito TTC.

Se requiere entonces un circuito con un disparador variable, el cual puede encenderse en un ancho rango de voltaje de transición que la propia interface del laboratorio dispara TTL. Así pues los circuitos pueden ser construidos usando un dispositivo provisto de un circuito integrado, que se conoce como comparador. Este dispositivo es como un

amplificador operacional, acepta 2 señales de entradas de voltaje (V+, V-) pero provee un pulso TTL digital de salida. Cuando V+<V- la salida del comparador pone 0V (TTL Low) e inversamente, si V-<V+ la salida es 5V (TTL High).

3.7 Fuentes de Interferencia Eléctrica

Los registros electrofisiológicos son también conocidos por ser propensos a interferencias de fuentes externas. Esto no es sorpresa dado que las amplitudes de las señales de interés son en orden de magnitud más pequeñas que el voltaje principal del AC en el laboratorio. Una variedad de interferencias son posibles, capactiva, inductiva, y conductiva. La captación por acoplamiento de interferencia se transmite vía la pequeña pero significativa capacitancia mutua que existe entre algunos de los conductores. En el laboratorio la fuente principal es la falta de protección de las conexiones de cables de energía AC, (instalación eléctrica).

El acoplamiento inductivo y conductivo ocurre primariamente como una consecuencia de múltiples conexiones a la tierra eléctrica. Las señales eléctricas deben darse con referencia a algún potencial estándar, usualmente el potencial de la tierra debajo de la construcción. También dos conexiones de la señal deben ser hechas a el tejido, una conexión a la señal medida y una a la tierra. Igualmente aquí debe ser una señal y una conexión a tierra ya que siempre una señal eléctrica es transmitida entre 2 regiones de los aparatos. En el sistema de registro electrofisiológico normal, todos los dispositivos cuentan con sus propias conexiones a tierra. La interferencia conductiva aumenta porque las conexiones entre la tierra de un aparato y la tierra muestra puede no ser perfecta, y también esto hace posible pequeños cambios de voltajes producidos por la corriente suministrada.

Las computadoras en el laboratorio pueden contribuir con interferencia adicional, dada la gran rapidez de pulsos digitales asociados con sus operaciones, los cuales crean largos acumulamientos de ruido de frecuencias de radio. Algunos problemas pueden ocurrir si este ruido es acoplado en un registro sensible (a la cabeza) o también los circuitos a tierra o electromagnéticamente.

Los problemas de interferencias deben ser eliminados o al menos minimizados por la distribución apropiada de las conexiones a tierra con el sistema de registro.

En relación a la computadora, los problemas por tierras son de mucho menor interés que los de su uso. La computadora personal moderna es un dispositivo pequeño, con poderosos circuitos como el resto de los aparatos.

CAPÍTULO II

ASYST
(A SCIENTIFIC SYSTEM)

1. DESARROLLO DE PROGRAMAS DE CÓMPUTO (SOFTWARE) PARA APLICACIONES CIENTÍFICAS

Existen cuatro rutas para obtener programas de cómputo para análisis de datos. Una de ellas es adquirir un paquete científico comercialmente disponible, la segunda es comisionar a una compañía de "software" para que lo escriba acorde a las especificaciones que se le den, la tercera es obtener el "software" de otros laboratorios con el mismo campo de trabajo, y la cuarta y última es desarrollarlo en casa. Cada una de estas maneras de obtener "software" tiene sus propios méritos y sus propios problemas, pero se debe de tener en cuenta que va a costar dinero y tiempo, por lo que su elección es de suma importancia.

ASYST es el "software" del cual nos ocuparemos en este capítulo, por lo que a continuación daremos algunas características y definiciones en torno al tema que nos permitan indagar en lo que realmente es este sistema científico (ASYST).

1.1 Compiladores

Un *compilador* convierte un programa escrito en un lenguaje de programación a un lenguaje de máquina o a un lenguaje ensamblador. En contraste al ensamblador el cual genera una instrucción de máquina por cada instrucción fuente del lenguaje ensamblador, el compilador usualmente genera algunas instrucciones de máquina por cada instrucción fuente del lenguaje de programación. La compilación se considera generalmente más complicada que un ensamblador, ya que la estructura de un lenguaje de programación tiende ha ser más compleja que la estructura de un lenguaje ensamblador[1]. Así pues un lenguaje de programación es necesariamente dependiente de un compilador. A continuación mencionamos los pasos involucrados por lo general, en una compilación:

1. El compilador lee el programa fuente, instrucción por instrucción y ejecuta los siguientes procesos por cada instrucción:
   a) Análisis léxico para identificar palabras clave, nombres, constantes, puntuación de caracteres y otras.

[1] Harry Katzan Jr. "Introduction to Computers and Data Processing"
D. Van Nostrand Company, USA 1979

b) Análisis sintáctico para identificar el tipo de instrucción y determinar que esa estructura sea admisible.
c) Identificación de lugares constituidos por instrucciones en listas y tablas para facilitar la generación de código máquina y para permitir el análisis global del programa.

2. Se lleva a cabo un análisis en el flujo del programa, para revisar los errores de instrucciones internas y para proveer información sobre cómo debe asignarse un registro de máquina.
3. Se optimiza el programa y se generan las instrucciones de máquina.
4. Se producen un programa objeto y un listado del programa.

Ejemplos de compiladores conocidos y muy utilizados en el ambiente científico son sin duda Fortran, Pascal y "C". En ellos el código fuente se escribe en un editor de texto, y posteriormente el compilador lo convierte en un archivo binario (código objeto), es cuando se obtiene el archivo en código objeto cuando se asocia a otros archivos en código objeto mediante un ligador (linkage editor), para finalmente producir el archivo en código ejecutable. Estos lenguajes de programación deben usarse preferentemente para la realización de proyectos largos, ya que son unos de los lenguajes mejor estructurados.

1.2 Intérpretes

Un tipo de "software" que permite modificaciones al programa durante su ejecución es el *Intérprete*. El *Intérprete* es un "software" que ejecuta un programa fuente sin producir un programa objeto[2]. Un intérprete procesa un programa fuente escrito en un lenguaje de alto nivel, justo como un compilador. La diferencia principal es que los intérpretes ejecutan una versión del programa fuente directamente al instante de trasladarlo al código de máquina. Un intérprete usualmente ejecuta funciones de análisis léxico y sintáctico como se ha descrito para un compilador, y entonces se traslada un lenguaje fuente en una forma interna. El proceso de traslación de un programa fuente en alguna forma interna, es simple y más rápido que compilar en un código de máquina. Sin embargo, la ejecución de un programa trasladado por un intérprete es mucho más lenta que la ejecución del código máquina producido por un compilador[3]. La ventaja real de un intérprete sobre un compilador, sin embargo, son las facilidades de depuración que éste último ofrece. Un intérprete opera como sigue:

1. El intérprete lee el programa fuente, instrucción por instrucción y ejecuta el procedimiento correspondiente para cada instrucción:
   a) La instrucción es buscada, identificada, analizada, e interpretada para determinar la operación que deberá ser ejecutada.

---

[2] Harry Katzan Jr. "Introduction to Computers and Data Processing"
D. Van Nostrand Company, USA 1979
[3] Leland L. Beck "System Software, An Introduction to Systems Programming"
Adisson-Wesley-Publishing Company, USA 1990

b) Las operaciones requeridas son ejecutadas por el intérprete y los resultados intermedios son retenidos.

2. La siguiente instrucción que es interpretada depende del resultado de la instrucción ejecutada anteriormente (como en el caso de una instrucción Go To).

Existen algunos intérpretes diferentes que varían en diseño interno, la clave consiste en que no producen un programa objeto y que todas las instrucciones no son necesariamente procesadas por el intérprete. La técnica interpretativa es usada frecuentemente con un uso fácil del lenguaje, o en un ambiente operacional donde los programas son ligados para ser recorridos muchas veces. Sin duda el intérprete más conocido es el Basic, en el cual las líneas de código fuente son trasladadas a una instrucción de máquina una por una (interpretadas), y siguiendo el flujo del programa cuando éste es ejecutado, lo que lo hace muy lento comparado con algunos compiladores. Sin embargo los programas escritos en Basic pueden ser compilados separadamente usando un compilador de Basic para producir un programa que corra tan rápido como uno en Fortran, Pascal o "C". Generalmente se usa al intérprete de Basic para probar los programas, ya que su depuración es mucho más sencilla, y una vez que corren sin errores se procede a compilarlos. Basic es ideal para proyectos a pequeña escala, que no rebasen las 1000 líneas de código, lo que ocurre fácilmente con programadores sin experiencia debido a las limitaciones que ofrece el propio lenguaje y por la ausencia de ciertas estructuras, especialmente la carencia de subrutinas independientes, ocasionando que se escriban programas largos e inmanejables.

1.3 "Software" para aplicaciones electrofisiológicas

Sin duda alguna los paquetes de programación más conocidos y usados para los fines de registro y análisis de señales electrofisiológicas son el *CED, PCLAMP* y *ASYST* a continuación hablaremos brevemente de los dos primeros, ya que a Asyst le ofrecemos la siguiente sección completa.

CED "software" para aplicaciones electrofisiológicas

Cambridge Electronic Design provee algunos programas de aplicación para tipos específicos de análisis, los cuales al escribirlos, crean un gran número de paquetes con un sistema modular de "software" para análisis electrofisiológico. El "software" es escrito en Pascal y hace uso del paquete para gráficas "METAWINDOWS"[4].

PCLAMP

PCLAMP es una colección de programas para adquisición y análisis de datos desarrollados para su utilización con el método experimental "patch-clamp", originalmente desarrollado en el Tecnológico de California (Cal. Tech.), pero ahora realizado por Axon Instruments Inc. (PCLAMP, Axon Instruments Inc.). El "software" ha sido constantemente actualizado llegando ahora a la versión 4. Ha sido escrito usando el compilador QuickBasic

[4] Fraser P.J. "Microcomputers in Phisiology, A Practical Aproach"
Departament of Zoology, University of Aberdeen, Irl Press, Oxford-Washington DC 1988

2.0 de Microsoft, más algunas rutinas en lenguaje ensamblador. Se requiere la tarjeta Tecmar Labmaster como la interface de laboratorio, suministrando un CAD con índices de muestreo de entre 33 a 70 kHz dependiendo de la computadora.[5] La falta del modo DMA para la adquisición de datos (ver capítulo anterior), es la limitación distintiva de la tarjeta Tecmar. El hecho de que el código fuente suministrado esté completamente escrito en un lenguaje simple y accesible como Basic, permite al usuario actualizar y modificar estos programas para sus propios propósitos.

2. *ASYST*, ¿QUÉ ES Y CÓMO FUNCIONA?

Asyst ofrece un sistema de programación versátil en cuanto a adquisición y análisis de datos se refiere. Asyst (creado por la Macmillan Software Company) es un "software" con diversas aplicaciones en el área científica, que permite manipular funciones matemáticas trascendentes tales como las funciones trigonométricas e hiperbólicas, la función exponencial, y algunas funciones más sofisticadas como la beta, la gamma y las funciones Bessel. Además permite realizar transformadas de Fourier y análisis estadísticos, así como diferenciar e integrar algunas funciones sencillas, o manejar números complejos. Sin duda alguna estas aplicaciones son importantes, pero en la que nos enfocaremos nosotros y es tal vez la más importante de este "software", es en su capacidad de ***adquisición, análisis, y presentación de datos***, recurriendo a algunas funciones matemáticas ya mencionadas, y permitiendo el manejo de gráficas con gran facilidad, lo que es uno de sus mayores atractivos.

Asyst es un Intérprete que permite tanto ejecutar un simple comando, o realizar programas complejos dependiendo de nuestras propias necesidades. Asyst al igual que el Basic nos permite programar en un lenguaje sencillo para interactuar con la máquina, ya que como lo mencionamos en la sección 1.3 damos una instrucción, se ejecuta y espera la siguiente dando el resultado inmediatamente, pero a la vez podemos agrupar un conjunto de instrucciones que realicen una tarea específica, y ejecutarlas. Para su programación Asyst nos ofrece la opción de crear palabras (words) que más adelante describiremos a detalle, y además nos permite realizar un Sistema de Ejecución (Run-time System) que es similar a un ejecutable y del cual hablaremos a detalle en su sección correspondiente.

Asyst maneja comandos capaces de soportar un gran número de interfaces de laboratorio como son, Data Translation, Cambridge Electronic Design, Tecmar, Metrabyte Data Acquisition y otras, puede desplegar archivos de datos digitalizados en diversos formatos, y provee operaciones con arreglos de datos, suavizar señales etc. Sin embargo Asyst por ser en esencia un intérprete, es substancialmente más lento que un programa escrito en algún lenguaje de alto nivel y compilado a disco para crear un programa ejecutable. En muchos casos esto no tiene demasiada importancia ya que son mucho

[5] Fraser P.J. "Microcomputers in Phisiology, A practical Aproach"
Departament of Zoology, University of Aberdeen, Irl Press, Oxford-Washington DC 1988

mayores las ventajas que ofrece el Asyst en su programación a cambio de una pequeña diferencia en la velocidad de ejecución.

2.1 Requerimientos y Configuración

Hay varias formas de hacer uso del *Asyst* dependiendo de las necesidades que se tengan. Por un lado podemos hacer uso de los comandos que maneja, excepto aquellos que se utilizan para la adquisición de datos, ya que para ello necesitamos de una tarjeta de adquisición. A continuación enunciaremos solamente lo necesario para hacer uso del Asyst en su forma más austera, y en el capítulo siguiente hablaremos de los requerimientos que se tienen para efectuar la adquisición de datos.

Configuración mínima de Asyst:

- Una computadora con sistema operativo MS-DOS (compatible), de preferencia con un procesador 386,486 ó péntium.
- Un cooprocesador matemático 80X87 (SX) instalado.
- Un mínimo de 640 KBytes en memoria RAM.
- Tener un disco duro instalado.
- Tener los siete discos de los cuales consta el "software" Asyst:
  - El disco del sistema que contiene 5 archivos.
  - El disco de ayuda que contiene un archivo.
  - El disco de configuración de "Overlays" conteniendo 17 archivos.
  - El disco que contiene los sistemas "Overlays" con 44 archivos disponibles.
  - El disco de "External DAS Driver" (manejador de tarjetas externas) con 6 archivos.
  - Dos discos con ejemplos de aplicaciones que contienen una serie de programas de demostración y utilidades.
- Un conector de protección contra copias (la licencia), el cual permite el uso de Asyst. Este conector es un dispositivo pequeño el cual se conecta a algún puerto paralelo de la impresora. El conector de protección es transparente a la impresora, por lo que es posible hacer uso de ella con el dispositivo conectado.

Se procede a instalar Asyst colocando primeramente el bloque de protección en el puerto paralelo correspondiente, y a continuación copiar los siete discos que contienen el "software" en un directorio creado previamente en el disco duro y llamado Asyst.

Asyst es un sistema interactivo y además un sistema que permite desarrollar programación. Una vez instalado en disco duro, basta con escribir la palabra Asyst y dar <enter> y aparecerá una pantalla de presentación pidiendo una clave para continuar, se oprime cualquier tecla, y a continuación aparecerá un OK que es el cursor (<prompt>), que nos indica que está listo. Antes de poder hacer uso de los comandos, es necesario saber que podemos crear diversas aplicaciones de Asyst con diferentes configuraciones, y atendiendo a uno o varios problemas específicos cada una. Una aplicación en Asyst no es otra cosa que una versión de Asyst con ciertos subsistemas cargados (overlays), que permiten hacer uso de comandos dirigidos a una tarea en particular, como son el manejo del editor, el sistema

de ayuda, el manejo de archivos de datos, el manejo de la impresora, la adquisición de datos, y otros.

Una vez elegidos los "overlays" que se necesitan, se procede a grabar esta versión de Asyst y a darle un nombre, por lo que existen dos teclas que son muy útiles e importantes en su manejo como: <F1> y <F2>. La tecla <F1> despliega el sistema de ayuda de Asyst, que se encuentra dividido en diversos temas para que el usuario pueda encontrar fácilmente el apoyo que necesita, la tecla <F2> nos da acceso al menú de configuración de Asyst en el cual podemos seleccionar "overlays", memoria, tarjetas etc., y al final salvar esta configuración y darle un nombre para usos posteriores. Una vez que se ha configurado Asyst y se ha salvado esa aplicación, desde el <prompt> OK podemos hacer uso de los comandos que dicha aplicación nos permite.

Sistemas de Bibliotecas (Overlays).

Asyst ofrece un amplio rango de posibilidades, las cuales pueden usarse simultáneamente en algunas aplicaciones. La arquitectura de sistemas de bibliotecas que maneja Asyst, nos permite cargar en memoria solamente las capacidades que se necesitan para una aplicación particular, y descargarlas cuando ya no nos sean necesarias. Las bibliotecas (overlays) que se usan frecuentemente, pueden ser configuradas como una parte permanente de la aplicación en uso. En suma, emplear la arquitectura de sistemas de bibloteca (overlays) permite un uso extremadamente eficiente de la memoria de la computadora.

Asyst soporta tres diferentes tipos de "overlays": los del sistema, los de aplicación, y los de configuración. A continuación se explicará brevemente cada uno de ellos.

Los "overlays" del sistema son archivos conteniendo funciones descritas como comandos de Asyst. Todos los comandos de estadística por ejemplo, son agrupados en un "overlay" llamado SPFN.SOV. Los "overlays" del sistema son aproximadamente 35 y tienen la extensión .SOV.

Un "overlay" del sistema puede ser configurado como una parte permanente del sistema, o pueden ser cargados de manera temporal mientras son requeridos. Un subsistema temporal es llamado un "overlay" transitorio. Para poder cargar un "overlay" transitorio es necesario dar desde el <prompt> OK, el comando **Load.Overlay** y a continuación el nombre del "overlay" correspondiente. Una vez que se ha terminado de trabajar en Asyst y se desea salir se da el comando **BYE** y <enter> , así el "overlay" es borrado de memoria y para la siguiente ocasión que accesemos esa aplicación no existirá. Para hacer uso de los "overlays" del sistema como una parte permanente de nuestra aplicación, se deberá de hacer uso del menú de configuración de Asyst, salvando como un archivo la versión que contiene dicha aplicación.

Existen sistemas de bibliotecas (overlays) que solamente están disponibles para el usuario en cuestión, y cuyos comandos solamente pueden ser usados por él, esto es, cuando nosotros desarrollamos nuestros propios programas para complementar las necesidades de

algunas aplicaciones. Así el usuario puede crear sus propios "overlays" que se les llama de aplicación y usan la extensión .AOV. Los "overlays" de aplicación pueden ser cargados de manera transitoria; y pueden ser accesados por el menú de configuración del Asyst. Al igual que los anteriores pueden también formar parte permanente de nuestra aplicación.

El tercer tipo de "overlays" soportados por el Asyst, son los llamados de configuración o internos, y no son usados o modificados por el usuario. Estos "overlays" son usados por el menú de configuración y sirven para optimizar el sistema.

Configuración de Asyst.

Muchos aspectos operacionales de Asyst pueden ser fácilmente configurados para satisfacer los requerimientos propios del usuario, usando el menú de Configuración del sistema. Estos aspectos operacionales de Asyst pueden ser usados temporalmente mientras dura la sesión actual o ser salvados permanentemente en una nueva versión de Asyst.

El menú de configuración de Asyst se accesa con la tecla <F2>. Existen diversas opciones de configuración como son:

a) Configuración de Sistemas de Bibliotecas (Overlays):
Especifica cuales "overlays" se han dejado como parte permanente de Asyst, además de que pueden darse de alta los que se necesiten.

b) Configuración de Memoria:
Especifica el tamaño de las diversas áreas de memoria, como son el uso del diccionario, la tabla de símbolos, el teclado, etc.

c) Configuración DAS :
Especifica la configuración de la tarjeta utilizada para la adquisición de datos, su cantidad y tipo. Esta opción requiere que se haya cargado previamente el manejador DAS en el menú de configuración de "overlays".

d) Configuración GPIB:
Especifica la configuración del "bus" de la interfaz GPIB (general purpose interface board, IEEE-488). Este menú requiere que haber cargado un manejador GPIB del menú de configuración de "overlays".

e) Configuración del "Hardware":
Se usa para especificarle a Asyst con que tipo de microprocesador está trabajando, su velocidad, tipo de monitor y el tipo de impresora(s) en el sistema.

f) Configuración de Colores de Texto:
Se usa para especificar los colores con que se trabajara en las diferentes ventanas usadas por el Asyst. Estas son: en despliegue normal (normal display), despliegue

de la pila (stack display), la de ayuda (help), la del editor de arreglos (array edit), y la ventana del editor de textos (text edit).

g) Configuración de Gráficas:

Se usa para especificar el estilo de gráficas a ser usado por Asyst. Se precisa el modo gráfico, el estilo de los ejes, el área de texto, el estilo de las etiquetas para los ejes, los colores del texto, el tamaño del área de graficación, el estilo de la gráfica (línea, puntos etc.), y los colores usados en el área de graficación (fondos). Cabe mencionar que durante la utilización de una aplicación los colores pueden modificarse con algunas instrucciones que veremos más adelante, pero al iniciarla de nuevo se arrancara con los especificados en este menú.

h) Analizador de Archivos:

Se usa para analizar, controlar y evaluar diversos archivos.

i) Estado del Sistema:

Se usa para especificar el estado del sistema para Análisis, Despliegue e Interpretación Numérica, el Uso de la Campana, Despliegue de Errores, Tamaño del Sistema y "buffers" para el teclado.

j) Modo de Despliegue:

Se usa para establecer el modo de despliegue que debe iniciar al arrancar la aplicación en uso, se puede seleccionar entre despliegue normal (normal display), modo gráfico (Graphics Display), y despliegue de la pila (Stack Display).

k) Salvar/Salir (Save/Exit):

Permite que las opciones de configuración sean salvadas en una versión actualizada de Asyst. Una vez que se ha salvado la nueva versión o aplicación de Asyst, el control se dejara en el <prompt> de DOS.

Es preciso mencionar que antes de hacer uso de la opción de configuración DAS se debe(n) instalar la(s) tarjeta(s) de adquisición de datos en la máquina.

Cuando se salva una versión actualizada de Asyst, se crean dos archivos: "Nombre".COM y "Nombre".OVL, donde "Nombre" es el nombre que asignamos en el menú de salvar cambios. Para entrar a la nueva versión o aplicación de Asyst, se da el "Nombre" desde el <prompt> de DOS. Si se ha creado un nuevo directorio para la nueva aplicación, o esta se ha copiado a un disco, es preciso tener además de los dos archivos "Nombre".COM y "Nombre".OVL el archivo ASYST.MSG, de lo contrario al ejecutar esta nueva aplicación se generará un mensaje de error.

2.2 Manejo de Asyst

Como ya lo mencionamos anteriormente podemos hacer uso de Asyst de dos maneras: de una manera interactiva, o por medio de programas de aplicación. Existe también la posibilidad de hacer pequeños programas que puedan ser utilizados cuando se está trabajando de forma interactiva. A continuación explicaremos en qué consisten y la forma en que trabajan cada una de ellas, así como el manejo de la pila (stack), los formatos de despliegue numérico que se pueden usar y los tipos de números que soporta.

Operación Interactiva

Podemos utilizar Asyst en forma interactiva, es decir nos permite que se ejecuten al instante los comandos que se le den desde el <prompt> OK, regresando el resultado inmediatamente. Dichos comandos pueden ser operaciones matemáticas con números, con variables, con arreglos de datos, o la ejecución de algunas otras funciones como son la graficación, la creación de archivos de datos, o la adquisición de datos por medio de una tarjeta de adquisición. El modo interactivo provee la facultad de accesar estas poderosas funciones y comandos directamente desde el teclado de la computadora. Es preciso mencionar que todas aquellas instrucciones que se utilizan de manera interactiva también pueden utilizarse para formar parte de módulos de programación que veremos en la siguiente sección, exceptuando solamente aquellas que se indiquen.

Programando en Asyst.

Para entender cual es la forma de programar en Asyst debemos primeramente definir algunos conceptos manejados como son: palabras (words), el diccionario, definición de dos puntos (colon definitions), saber como cargar archivos de texto que contengan instrucciones básicas de Asyst, y palabras que formen parte de programas en código fuente.

Palabras (Words): Se le llama *palabra* a todo aquello que es definido por el usuario y que no forma parte del diccionario de Asyst. Las palabras pueden ser nombres de variables, nombres de arreglos, áreas de graficación, nombres de plantillas para adquisición de datos, y nombres de rutinas de programación que contengan un conjunto de instrucciones. Es preciso mencionar que cuando una palabra representa una rutina de programación, esta puede hacer uso de palabras que han sido definidas anteriormente a ella, es decir puede usar variables, plantillas, arreglos, y otras rutinas de programación que hayan sido definidas por el usuario, y se encuentren en el diccionario de Asyst.

El Diccionario: Todas las palabras disponibles para Asyst son listadas en un diccionario. Para poder ver el contenido del diccionario, basta con escribir **OK ?WORDS** y dar <enter> , a continuación se desplegara una lista de todas las palabras que Asyst tiene disponible para su uso en ese momento. Como la lista generalmente es muy larga, y no puede verse en una sola pantalla, es preciso pararla oprimiendo cualquier tecla y oprimir otra cuando se desee continuar. Puede darse el caso que una palabra sea renombrada, o que por equivocación se repitan dos con el mismo nombre, en cuyo caso solamente se tomará en cuenta la última versión de la palabra.

Definición de dos puntos (Colon definitions): Una de las características más poderosas de Asyst es el hecho de que se puede extender el diccionario para incluir palabras y conceptos que no son parte del sistema normal de Asyst, pero que el usuario requiera.

La manera de formar palabras (words) es muy simple, y se puede llamar una nueva palabra a cualquier cosa definida por el usuario y que no sobrepase de 31 caracteres. Estos caracteres pueden ser cualquiera que se puedan escribir del teclado excepto un espacio.

Para definir una nueva palabra en Asyst, se le da un nombre único, se especifican las instrucciones (en caso de que sea una definición de dos puntos) que va a ejecutar dicha palabra, y finalmente se le indica que ha terminado de definirse esa palabra.

La instrucción que se utiliza para definir nuevas palabras es ":" (por lo que se le llama definición de dos puntos), a continuación se da el nombre de la nueva palabra, inmediatamente después las instrucciones que involucra, y finalmente se indica que se ha terminado ";" . Así para definir una nueva palabra que lleve por nombre precisamente **"Nueva.Palabra"** y que ejecute un conjunto de n instrucciones se deberá escribir como sigue:

OK **: Nueva.Palabra**
OK *instrucción 1*
OK *instrucción 2*
OK .
OK .
OK .
OK *instrucción n*
OK ;

Si a continuación se escribe la instrucción ?Words, se verá que la palabra que se encuentra en el tope del diccionario es *Nueva.Palabra*, y puede ser usada en el momento en el que se le requiera. Podemos definir de la misma manera otra nueva palabra que a su vez aparecerá en el tope del diccionario, y que puede hacer uso de la *Nueva.Palabra* que se definió con anterioridad a ella.

Mientras las definiciones de dos puntos pueden hacerse de manera interactiva, como en la definición de *Nueva.Palabra*, por lo regular sólo se utiliza este método cuando se está experimentando en Asyst. Los programas reales son demasiado largos para hacerse de manera interactiva, por lo que en respuesta a esta dificultad Asyst puede cargar programas de archivos de texto ASCII, los cuales han sido creados usando el editor de texto de Asyst o algún otro editor de texto estándar.

Archivos de Texto y Programas en Código Fuente:
Como ya se mencionó anteriormente Asyst puede cargar archivos de texto que contengan programas en código fuente. Los archivos de texto son definidos como archivos compatibles en DOS conteniendo solamente caracteres ASCII. De esta manera se pueden generar archivos en código fuente para un programa de aplicación el cual podremos modificar tan solo con editarlo. En este archivo se pueden hacer comentarios, definir variables, arreglos, plantillas para adquisición de datos, áreas de graficación y palabras

utilizadas como rutinas de programación. Es de esta manera que se puede estructurar un programa de aplicación o realizar un pequeño sistema que involucre varias palabras utilizadas como rutinas de programación. Una de las características de esta manera de programar es que para poder enlazar varias palabras o programas, es preciso definir en el archivo de texto primeramente las palabras que van ha ser utilizadas por otras y a continuación las que se utilizan, esto da la particularidad de que por lo general la última palabra definida en el archivo de texto, es la que se encarga de enlazar y ligar a todas las anteriores, y casi siempre es con la que se arranca el sistema de aplicación o el programa principal. A continuación diremos como se realiza generalmente un sistema o programa dependiendo de su tamaño y su(s) aplicación(es) según sea el caso. el programa se llamará **ProgEjem.Dmo**. Recordemos que nos encontramos en un editor de texto:

```
\ Archivo   ProgEjem.Dmo
\ La anterior y esta son líneas de comentarios las cuales se empiezan con " \ "
\

\ **Definición de variables, arreglos, plantillas etc. que han de utilizarse (Ambiente)**
variable1, variable2, ..., variableN
arreglo1, arreglo2,..., arregloN
plantilla1, plantilla2 ,..., plantillaN

\
\ **Definición de Palabras**
: Palabra1
{instrucciones}
;
: Palabra2
{instrucciones}
;
.
.
.
: PalabraN
{instrucciones}
;

\
\ Fin del programa
```

Esta es la manera en que generalmente se escriben los programas de aplicación, en donde debe mencionarse que la palabra que lleva el nombre *PalabraN*, puede hacer uso de las N-1 palabras definidas anteriormente a ella, es decir, dentro del conjunto de sus instrucciones puede mandar ejecutar a alguna de las anteriores N-1, y así sucesivamente, pero la *Palabra1* por ejemplo, no puede hacer uso de ninguna de las siguientes palabras.

Una vez teniendo este archivo de texto con el programa *ProgEjem.Dmo* se procede a cargarlo para poder hacer uso del, o de las palabras que en el se definen, por lo que es necesario usar la instrucción load cuya forma general es:

OK LOAD "nombre-del-archivo"

Donde el nombre-del-archivo es el que se le da al crear el archivo de texto. En nuestro ejemplo bastaría con dar la instrucción:

OK Load ProgEjem.Dmo

Así al ver nuevamente el diccionario estarán las nuevas palabras definidas y bastaría con dar el nombre de alguna de ellas para que el conjunto de instrucciones que en ella se encuentran sean ejecutadas.

Manejo del Editor de Asyst

En la configuración estándar de Asyst, el editor no está normalmente disponible, pero puede ser cargado como un "overlay" transitorio para que pueda ser utilizado. Esto se hace posible dando la siguiente instrucción:

OK Load.Overlay EDITOR.SOV

El editor de textos ha sido diseñado para permitir al usuario editar programas fuentes mientras se encuentra trabajando en el ambiente Asyst. Esto es muy conveniente cuando se requiere depurar o modificar nuevos programas, los cuales generan errores al cargarse. Sin embargo el editor se ve restringido al hacer uso de archivos que sean más pequeños que 64Kbytes, para poder editar a estos archivos se puede hacer uso de otro editor de texto de la preferencia del usuario. Para editar un archivo fuente ya existente, es necesario dar la siguiente instrucción:

OK EDIT <enter>

A continuación pedirá el archivo a ser editado, este puede ya haber sido creado o si es nuevo nos deja una pantalla para poder escribir, también permite hacer uso de las funciones <F9> para poder cambiar a un directorio en particular, y <F10> para salir. Una vez dado el nombre de nuestro archivo, es leído por Asyst, y si excede los 64K de tamaño arrojará un error, de igual manera si alguna de sus líneas excede 80 caracteres de longitud, y no permitirá que se tenga acceso al archivo.

Ya habiendo pasado las restricciones anteriores, y dando el nombre de algún archivo válido, se presentara un nuevo menú de opciones y una pantalla para que el usuario pueda hacer uso de ella. Es en esta pantalla en donde el usuario puede escribir sus programas, textos, u otros archivos que el desee, haciendo uso de todos los caracteres que el teclado

contiene, y además de aquellos que permite el código ASCII. Existen algunas teclas de las cuales podemos hacer uso y permiten realizar funciones específicas, como son:
<Ins> , nos permite insertar caracteres.
<Home>, permite regresar el cursor a la línea 0 columna 0 de la pantalla actual.
<end>, leva el cursor a la línea 19 columna 0.
<Pg-Up>, lleva una pantalla arriba al cursor.
<Pg-Dn>, lleva al cursor una pantalla abajo.

En la parte del menú el cual mencionamos anteriormente, se encuentra la tecla a usar y especifica de forma general el uso de la misma, estas son:
<F1> Word, al presionarla y hacer uso de las flechas para poder movernos, los movimientos serán de palabra en palabra.
<F2> Line, similar a la anterior pero los movimientos serán de línea en línea.
<F3> Page, los movimientos serán de página en página.
<F4> File, combinada con <home> y <End> permite regresar el cursor al principio o al final del archivo.

Las cuatro funciones anteriores se desactivan tan sólo volviendo a oprimirlas, y regresando al estado normal (manejo por caracteres).
<F5> Direction, esta tecla tiene significado tan sólo cuando se va a borrar texto, así pues, si la presionamos, y a continuación presionamos la tecla <Del> todo el texto que le sigue sobre la misma línea es borrado.
<F6> Find&Repl, se usa para buscar rápidamente puntos específicos en el archivo, esta búsqueda se hará solamente de la posición en la que se encuentre en ese momento el cursor hasta el final del archivo (nunca hacia atrás), se puede hacer uso de esta tecla, para buscar, remplazar, repetir la acción y condicionarla.
<F7> Block Mark, se usa para marcar una área de texto específica, se oprime donde va ha ser el inicio de esta área, se mueve el cursor hasta donde va ha ser el final de esta área y ahí se vuelve a oprimir, quedando marcada una área de texto para realizar alguna función posterior con ella.
<F8> Block Merge, el uso de esta tecla siempre sigue a la de <F7>, ya que ésta nos permitirá elegir la operación que se desea realizar con el bloque de texto marcado, estas opciones son: Copiar, Mover, Borrar, Escribir en un archivo, Desmarcar, Condicionar, y la opción se ejecutará tan sólo oprimiendo del teclado la primera letra de la opción escogida.
<F9> Save, permite salvar el archivo de texto con los cambios que se le hayan realizado después de la última vez que fue salvado.
<F10> Exit, al oprimir esta función, no se saldrá inmediatamente del editor de Asyst, ya que se desplegará otro pequeño menú de opciones en donde nos permite escoger entre Cargar (Load), Continuar (Continue), Condicionar (Else) y Salir (Exit). Para escoger una de estas opciones basta con oprimir la primera letra de su nombre. Si se escoge Load el archivo se salva y se carga, similar a una instrucción "load", y sale del editor, si hay algún error al cargarlo, al volver a editarlo se preguntará si se desea posicionar en el lugar donde ocurrió el ultimo error, si se da "y" el cursor se posicionará ahí de lo contrario se posicionará al inicio del archivo. Si se escoge Continue, nos regresara al archivo en edición. Si se escoge Exit, saldremos del editor y nos volverá el cursor al ambiente de Asyst.

Manejo de la Pila

Todos los números, arreglos, y cadenas en Asyst son consultados por medio de una área especial de memoria llamada el número de pila. Esta pila es técnicamente llamada una LIFO stack (LIFO de "Last In, First Out"). Esto es que el último elemento que se pone en la pila, es el primer elemento que puede ser removido. Para poder tener acceso al segundo elemento de la pila, se deberá primero remover o hacer uso del primero (en este caso es el último que entró a la pila).

Entender el funcionamiento de una pila es muy fácil, ya que podemos tomarla como una sucesión de elementos que están entrando y saliendo de un recipiente, al cual sólo puede entrar o salir un elemento a la vez. Cuando se toma un elemento de la pila, este elemento es el que se encuentra en el tope del recipiente, pues los demás se encuentran por debajo de él. Para poder tomar a uno de los elementos inferiores hay que remover primero todos los que se encuentran por encima de él. Dicha operación se realiza sin perder los elementos que son removidos, pero se requiere de otro recipiente para colocarlos; de forma similar se trabaja con varias pilas. Por esta razón se hace necesario algún formato para definir las pilas que se deseen ocupar, para lo cual se utiliza la siguiente instrucción:

Tamaño **STACK** *Nombre-Pila*

Tamaño es un número entero que indica el tamaño en bytes de la pila que estamos definiendo, STACK es la instrucción que indica su creación y *Nombre-Pila* es el nombre que se le asigna a la nueva pila.

De esta forma podemos definir las pilas que necesitemos sin sobrepasar el área de memoria que se ha asignado a su manejo, en donde cada número real requiere de 6 bytes, cada complejo 10 bytes, y cada arreglo requiere aproximadamente de 20 bytes. Para hacer uso de alguna pila ya definida basta con dar su nombre y <enter> a continuación, así queda activada para su uso y la anterior desactivada. Por omisión se hace uso de la pila llamada DEF.STACK (de 512 bytes de tamaño) hasta que no se haga referencia a alguna otra pila que ya se haya definido anteriormente.

Poner un número en la pila de Asyst es algo muy simple. Solamente se debe dar un número en la línea de comandos y en seguida dar <enter>. Asyst debe reconocer esto como un número y automáticamente ponerlo en el tope de la pila. El número debe permanecer en la pila de Asyst hasta que el usuario active removerlo de ahí. Al igual que los números, también se pueden apilar arreglos, los cuales deben ser definidos anteriormente a su colocación en la pila (ver sección 2.3) y pueden operarse de la misma manera que un número al ser removido o puesto en ella. Para el manejo de cadenas de caracteres se utiliza una pila diferente que para los números y arreglos, la pila es llamada pila de "strings", y es en ella se apilan todas las cadenas de caracteres que han de usarse. Hacer uso de esta pila es similar a las de números y arreglos, las cadenas pueden definirse con anterioridad a su entrada en la pila, o asignarlas sin nombre es decir sólo como una cadena de caracteres, escribiendo desde la línea de comandos un string como sigue :

"Cadena de caracteres" y a continuación <enter>, y la cadena se encuentra en el tope de la pila de "strings".

De manera alternativa se puede dar más de un elemento en una misma línea dejando entre cada elemento un espacio de separación. Cada elemento es puesto en la pila, en secuencia tal que el último número de la línea deberá finalizar en el tope de la pila.

La pantalla de ambiente "stack.display" despliega el contenido de la pila usando la parte superior de la pantalla normal, solamente se despliegan los cuatro valores actuales que se encuentran del tope de la pila hacia abajo. Es preciso mencionar que toda operación que se realice con elementos de la pila, los remueve automáticamente, y el resultado es puesto en el tope, por lo que se hace necesario el uso de nombres de variables para que les sean asignados los valores que se van modificando o que sean de interés para su utilización posterior. En la siguiente sección se detallará cuales son los comandos que se utilizan para el manejo de la pila, y sus diferentes formatos.

Tipos de Números y Formatos de despliegue Numérico

Asyst soporta números enteros, reales, y complejos. Cada uno de esos tipos de números, pueden ser usados en simple o doble precisión. El tipo de número es especificado cuando un número es introducido para operaciones directas y es especificado en la definición de escalares o arreglos. Asyst permite mezclar tipos de números en una operación dada, sin tener que especificar a la computadora como realizar la operación con tipos de números mezclados. El coprocesador numérico se usa para todas las operaciones numéricas produciendo gran precisión, y permite el manejo de un gran rango de números. El número de bytes de memoria usados para almacenar, así como el rango y precisión de varios tipos de números están dados como sigue:

Enteros de Simple Precisión	
Memoria necesaria	2 bytes
Precisión	5 dígitos
Rango	-32,768 a +32,767
Enteros de Doble Precisión	
Memoria necesaria	4 bytes
Precisión	10 dígitos
Rango	-2,147,483,648 a +2,147,483,647
Números Reales de Simple Precisión	
Memoria necesaria	4 bytes
Precisión	7 dígitos
Rango	$8.43 * 10^{-37}$ a $3.37 * 10^{38}$
Números Reales de Doble Precisión	
Memoria necesaria	8 bytes
Precisión	16 dígitos
Rango	$4.19 * 10^{-307}$ a $1.67 * 10^{308}$

La especificación del tipo de número depende de cuanta memoria sea necesaria para almacenar el conjunto de datos con los cuales se trabajará, la precisión que demanda el cálculo, y la rapidez requerida para el cálculo. Asyst permite almacenar los datos en un arreglo que contenga los tipos de números antes mencionados. El arreglo de datos más grande que Asyst puede almacenar es de 65,536 bytes. Si se usan enteros de simple precisión, se pueden almacenar hasta 32,768 datos individuales. Con enteros de doble precisión o números reales de simple precisión, el máximo tamaño del conjunto de datos es de 16,384 elementos. Con números reales de doble precisión, el tamaño máximo de un arreglo de números es de 8,192 elementos.

Los números complejos requieren del doble de memoria, ya que ambas partes, la parte real y la imaginaria deben ser almacenadas. Si se necesita almacenar grandes arreglos de datos, es recomendable usar enteros de simple precisión, con lo cual se perderá precisión si se realizan un gran número de cálculos con redondeo, pero se ganará mayor rapidez en la ejecución de los cálculos. La mejor precisión se obtiene usando números reales de doble precisión, pero este tipo de números usa demasiada memoria. En muchos casos, los números reales de simple precisión ofrecen la mejor opción entre eficiencia de almacenamiento y precisión de cómputo.

Es preciso notar que al realizar operaciones con números, el resultado que se obtiene adquiere el tipo de número de los números que tomaron parte en la operación, recurriendo así al redondeo que en muchas ocasiones puede generar errores que llegan a ser de consideración en el resultado final. Es por esto muy importante que el usuario tenga claro lo que necesita de las operaciones que se van a realizar, para así optar por una buena selección en el tipo de número que va a utilizar.

De la misma forma en que Asyst permite seleccionar el tipo de números con los que se va a trabajar, también permite seleccionar el formato de despliegue de salida, que puede ser un formato fijo (el cual se da por omisión), o en un formato científico. También permite que se le indique el número total de dígitos que deberán aparecer, y el número de dígitos a la derecha del punto decimal. Para poder usar un formato de despliegue se usan dos palabras que pueden darse desde la línea de comandos OK. Las palabras son:

Fix.Format [n1 n2 -]

Formato en Modo Fijo. El tamaño total es de n1, con un espacio reservado para el punto decimal, y un espacio reservado para un signo menos. Si ocurre un "overflow" (más dígitos de los que se permiten), se debe imprimir un *. El argumento n2, de entrada indica el número de dígitos desplegados después del punto decimal. Hay un total de n1-1 dígitos desplegados para números negativos.

Sci.Format [n1 n2 -]

Formato en Modo Científico. Despliega la mantisa del número con un total de n1-6 dígitos (incluyendo el punto decimal) para números positivos, y n1-7 dígitos para números negativos. De esos n1-6 ó n1-7 dígitos totales, n2 de ellos deberán ser desplegadas a la

derecha del punto decimal. La medida total es de n1. Se usan cinco espacios para imprimir el exponente y un espacio adicional para el punto decimal.

2.3 Comandos Básicos

Asyst tiene diversas aplicaciones, como ya lo mencionamos anteriormente, cada una de ellas contiene instrucciones y sintaxis propias. Existen instrucciones que son básicas en todo lenguaje de programación, son las que se utilizan lo mismo en programas muy simples o en programas sumamente complejos. En Asyst estas instrucciones se basan principalmente en el manejo de la pila, los modos de despliegue, la definición de variables y las operaciones básicas con ellas, manejo de cadenas de caracteres, arreglos de datos, y gráficas. Es por eso que a continuación daremos la sintaxis de cada instrucción, los parámetros que usa y la descripción de la misma, además de su localización en el sistema de "overlays", ya que si el "overlay" correspondiente no ha sido cargado, no se puede hacer uso de dicha instrucción .

Comandos para el Manejo de la Pila

La pila es usada para todas las operaciones matemáticas, por lo que es necesario manejar el orden o posición de los números en ella. Existen instrucciones que permiten reacomodar los elementos de la pila, duplicarlos y eliminarlos, en donde un elemento puede consistir de un simple número(escalar) o un arreglo de datos numéricos. Estos elementos pueden ser o incluir cualquier tipo de números y pueden tener cualquier formato. Es preciso mencionar antes de describir cada una de las instrucciones utilizadas para el manejo de la pila, que para almacenar un elemento en ella basta con escribir el nombre de dicho elemento desde el <prompt> de Asyst, y a continuación dar <enter>. Por ejemplo, si se desea almacenar en la pila un elemento (el cual pudo haber sido definido como una variable numérica de cualquier tipo, como un arreglo, o un número), se escribe:

OK ***elemento*** <enter>

A continuación describiremos cada una de las instrucciones que permiten el manejo de la pila, dando por hecho que ya se ha elegido la pila con la que se va a trabajar (como se vio en la sección anterior) o si se ha omitido se entenderá que se está trabajando con la pila estándar del sistema (DEF.STACK).

DEF.STACK
Localización: Sistema Base
Descripción: Nos da el manejo de la pila estándar del sistema cuyo tamaño es de 512 bytes.

?STACK
Localización: Sistema Base
Descripción: Despliega el nombre y tamaño de la pila que se encuentra en uso.

?STACKS
Localización: Sistema Base
Descripción: Despliega los nombres y tamaños de todas las pilas definidas. El nombre de la pila que se encuentra actualmente en uso aparecerá resaltado en forma luminosa.

STACK.CLEAR
Localización: Sistema Base
Descripción: Borra todos los elementos que se encuentran almacenados en la pila.

STACK.RESET
Localización: Sistema Base
Descripción: Es similar a stack.clear excepto que este también reinicializa los arreglos sin nombre y borra todos los "tokens" creados.

#Entradas **XFER>** *Nombre-Pila*
Localización: Sistema Base
Descripción: Transfiere el número de entradas de la pila en uso, a la pila especificada con Nombre-Pila. Los valores transferidos no incluyen el valor #Entradas, además de que estos valores se pierden de la pila en uso.

?S
Localización : Sistema Base
Descripción: Despliega el contenido de la pila sin eliminar ninguno de sus elementos.

DUP
Localización: Sistema Base
Descripción: Duplica el número que se encuentra en el tope de la pila (cima).

DROP
Localización: Sistema Base
Descripción: Elimina el número que se encuentra en el tope de la pila.

SWAP
Localización: Sistema Base
Descripción: Intercambia los dos números que se encuentran en el tope de la pila, es decir el primero y segundo número que pueden ser tomados de la pila, [#2 #1 - #1 #2].

OVER
Localización: Sistema Base
Descripción: Duplica el segundo número de la pila, y lo coloca en el tope de la misma, [#2 #1 - #2 #1 #2].

ROT
Localización: Sistema Base
Descripción: Desplaza el dato que ocupa la tercera posición de la pila, hacia la primera posición, [#3 #2 #1 - #2 #1 #3].

UNROT
Localización: Sistema Base
Descripción: Reordena los tres primeros números que se encuentran en la pila, desplazando dos veces el que ocupa la tercera posición hacia la primera, [#3 #2 #1 - #1 #3 #2].

PICK
Localización: Sistema Base
Descripción: Copia el n-ésimo número de la pila a el tope de la misma, [#n...#1 - #n...#1 #n].

ROLL
Localización: Sistema Base
Descripción: Mueve el n-ésimo número de la pila a el tope de la misma,
[#n #n-1...#1 - #n-1... #1 #n].

UNROLL
Localización: Sistema Base
Descripción: Mueve el número que se encuentra en el tope de la pila a la n-ésima posición de la misma, [#n...#2 #1 - #1 #n...#2].

***DUP**
Localización: Sistema Base
Descripción: Duplica las n entradas que se encuentran en la pila, [#n...#1 - #n...#1 #n...#1].

***DROP**
Localización: Sistema Base
Descripción: Elimina los primeros n elementos que se encuentran en la pila,
[#n+1 #n...#1 - #n+1].

DEPTH
Localización: Sistema Base
Descripción: Regresa el número de elementos que se encuentran actualmente en la pila, colocando este número en el tope de la misma.

Modos de Despliegue
Asyst ofrece una técnica conveniente para visualizar las operaciones que se están realizando, para esto permite el manejo de tres formas de despliegue en video que son: Despliegue Normal (Normal.Display), Despliegue de la Pila (Stack.Display) y Despliegue de Gráficas (Graphics.Display). A continuación daremos la sintaxis de las instrucciones

usadas para controlar el formato de despliegue en video, así como una breve explicación de cada una de ellas, y del formato que describe.

NORMAL.DISPLAY
Localización: Sistema Base
Descripción : Pone en uso el modo normal de despliegue de texto. Este modo permite que el texto sea impreso sobre la pantalla de video, la cual consta de 25 líneas por 80 columnas. Este es el modo de despliegue con el que trabaja el sistema por omisión.

STACK.DISPLAY
Localización: Sistema Base
Descripción: Es un modo especial de despliegue de texto, que usa la porción superior de la pantalla de video para proveer un despliegue corrido de los cuatro elementos que se encuentran en el tope de la pila. El modo normal de despliegue aparece y puede usarse en la parte inferior de la pantalla, que son aproximadamente las dos terceras partes restantes.

Los cuatro elementos que se encuentran en el tope de la pila, son actualizados y desplegados después de que se ejecuten las instrucciones o la instrucción que se dió en la última línea de comandos (<prompt> OK). Este modo de despliegue sirve para dar seguimiento a las palabras que operan con elementos de la pila y verificar que se están ejecutando correctamente.

GRAPHICS.DISPLAY
Localización: Sistema Base
Descripción: Es usado como interruptor del adaptador gráfico para permitir el uso del modo gráfico. También inicializa el "software" para el manejo de gráficas utilizado por Asyst, y coloca por omisión una área de graficación y una área de texto. Las dos áreas más comunes para graficación son las que abarcan dos terceras partes de la pantalla ya sea en el lado derecho de esta y colocando el área de texto a la izquierda, o colocando el área de graficación en la parte superior de la pantalla y el área de texto en la parte inferior. Las áreas de graficación y de texto pueden ser modificadas por el usuario de acuerdo a sus necesidades.

Definiendo Variables (Escalares)
Además de poder almacenar temporalmente números o arreglos en la pila, es importante poder hacer referencia a ellos con nombres que los identifiquen y que nos permitan almacenarlos y accesarlos muchas veces, sin perder o modificar su contenido. En suma los nombres (ya sea de variables, arreglos, o cadenas) permiten hacer referencia a esos valores para recuperar los datos que contienen. Asyst se refiere a estos nombres de variables como "*escalares*".

Antes que pueda usarse algún escalar, primero se define el tipo de dato y el nombre con el que será reconocido por Asyst. Cualquier escalar que se defina, es reconocido por

Asyst con un único nombre, el que el usuario le ha dado. La única restricción para nombrar escalares es que el nombre sea menor que 32 caracteres de longitud y no contenga espacios. El formato general para la definición de un escalar puede darse de la siguiente manera:

Tipo_Número **SCALAR** *Nombre_Escalar*
Localización: Sistema Base
Descripción: Define un escalar con el nombre ("*Nombre_Escalar*") y el tipo de número ("*Tipo_Número*") especificado. Se puede hacer referencia a este nuevo escalar para subsecuentes operaciones en la pila, con tan sólo escribir "*Nombre_Escalar*" y dar <enter>, además lo inicializa con cero.

Pueden haber varios tipos de escalares, se encuentran definidos en el formato anterior por *Tipo_Número*, y se definen remplazando esta palabra por una de las siguientes instrucciones:

INTEGER
Descripción: Pone el tipo de número actual para definición de escalares como entero de simple precisión. El efecto de la instrucción INTEGER durará hasta que algún otro tipo de número sea especificado.

REAL
Descripción: Pone el tipo de número actual para definición de escalares, como real de simple precisión. Durará hasta que algún otro tipo de número sea especificado.

COMPLEX
Descripción: Coloca el tipo de número actual como un complejo de simple precisión. Su duración es condicionada como las anteriores.

DP. INTEGER
Descripción: Coloca el tipo de número como enteros de doble precisión.

DP.REAL
Descripción: Coloca el tipo de número como reales de doble precisión.

DP.COMPLEX
Descripción: Coloca el tipo de número como complejos de doble precisión.

?SCALARS
Localización: Sistema Base
Descripción: Despliega en pantalla una lista de las variables escalares que han sido definidas para el sistema. El despliegue incluye los tipos de números de los escalares

Asyst no tiene límites explícitos para el número de escalares que puedan ser definidos, pero es recomendable que el propio usuario haga un uso óptimo de este recurso para realizar programas más eficientes. Es recomendable al darle un nombre a un escalar, que este sea representativo de su contenido y uso.

Definiendo Cadenas de Caracteres ("Strings")

Al igual que con los escalares, se pueden definir cadenas de caracteres de acuerdo a las necesidades del usuario, para ser usadas como etiquetas de despliegue. Dichas cadenas de caracteres son secuencias de uno o más caracteres en código ASCII., que se manejan de manera análoga a los escalares, tomando todos ellos valores en la pila. Sin embargo, esta pila no es la misma que la usada para los escalares y arreglos de números, ésta se nombra pila de símbolos. Las cadenas pueden ingresarse directamente desde el teclado, y automáticamente se colocan en el tope de la pila de símbolos siguiendo el mismo procedimiento que con la pila de escalares, también pueden definirse variables de cadenas de caracteres para utilizarlas en procedimientos posteriores. Las instrucciones usadas para la definición y manejo de cadenas de caracteres son las siguientes:

." texto"
Localización: Sistema Base
Descripción: Imprime el texto que se encuentra entre las comillas (") en el dispositivo de salida que se encuentre activado en ese momento. La iniciación ." debe ser seguida por un espacio en blanco, el cual no se incluye en la impresión; sin embargo no se necesita un espacio en blanco antes de las comillas finales.

" texto"
Localización: Sistema Base
Descripción: Genera una cadena que contiene el texto que se encuentra entre las comillas ("), al igual que la anterior, la primera comilla debe ser seguida de un espacio en blanco. El "string" es puesto en el tope de la pila de símbolos.

"TYPE
Localización: Sistema Base
Descripción: Muestra la cadena más reciente que se encuentra en el tope de la pila de símbolos. El "string" pudo haber sido generado por alguna de las diversas instrucciones.

Máxima_Longitud **STRING** *Nombre*
Localización: Sistema Base
Descripción: Crea un "string" variable cuyo tamaño máximo está dado por el número Máxima_Longitud y se le reconocerá con la palabra Nombre.

?STRINGS
Localización: Sistema Base
Descripción: Nos muestra una lista de todas las variables "strings" que se encuentran actualmente definidas y que son reconocidos por Asyst.

DIM [*n1* . *n2*] STRING.ARRAY *Nombre*
Localización: Sistema Base
Descripción: Es usado para definir un arreglo uni-dimensional de "strings" que será reconocido con Nombre. El primer parámetro que es n1 es el número de "strings" en el arreglo, mientras el segundo parámetro n2, es el número máximo de caracteres por string.

"[n1]
Localización Sistema Base
Descripción: Es usada para especificar un elemento en el arreglo de "strings". El elemento especificado n1 es el elemento que será accesado, y será puesto en el tope de la pila de símbolos.

"TIME
Localización: Sistema Base
Descripción: Produce un "string" que contenga el tiempo actual, y lo coloca en el tope de la pila de símbolos.

.TIME
Localización: Sistema Base
Descripción: Imprime el tiempo actual.

"DATE
Localización: Sistema Base
Descripción: Produce un "string" Que contenga la fecha actual, y lo coloca en el tope de la pila de símbolos.

.DATE
Localización: Sistema Base
Descripción: Imprime la fecha actual

"INPUT
Localización: Sistema Base
Descripción: Requiere un "string" desde un programa en Asyst. Típicamente es precedido por un mensaje que se ha desplegado y a continuación se escribe el "string" desde el teclado.

"LEN
Localización: Sistema Base
Descripción: Regresa la longitud de la cadena que se encuentra en el tope de la pila de símbolos.

"DUP
Localización: Sistema Base
Descripción: Duplica el "string" que se encuentra en el tope de la pila de símbolos.

"SWAP
Localización: Sistema Base
Descripción: Intercambia las posiciones de los dos "strings" que se encuentran el tope de la pila de símbolos.

"DROP
Localización: Sistema Base
Descripción: Elimina el "string" que se encuentra en el tope de la pila.

"CAT
Localización: Sistema Base
Descripción: Une las das cadenas que se encuentran en el tope de la pila de símbolos y produce uno nuevo, dejándolo en el tope de la pila. La parte izquierda de la tercer cadena está compuesto por el primero que entró a la pila y la parte derecha por el segundo, es decir el que esta en el tope de la pila.

ASCII *Caracter*
Localización: Sistema Base
Descripción: Regresa el código ASCII del caracter que le sigue.

"COMPRESS
Localización: Sistema Base
Descripción: Es usado para remover todos los caracteres cuyo código ASCII se encuentra actualmente en la pila, de la cadena que se encuentra en el tope de la pila de símbolos.

?SS
Localización: Sistema Base
Descripción: Se usa para mostrar el contenido de la pila de símbolos.

SS.CLEAR
Localización: Sistema Base
Descripción: Borra todo lo que se encuentra actualmente en la pila de símbolos.

Arreglos de Datos
La información más comúnmente utilizada en el área científica y de ingeniería se da en forma de conjunto de datos o arreglos que contienen datos similares o relacionados entre sí, estos datos pueden ser simples números o escalares, o cadenas de caracteres. En muchas cosas los arreglos de datos son similares a los escalares, ya que utilizan los mismos operadores aritméticos, y algunas funciones especiales que se analizarán más adelante.

De manera sencilla, un arreglo puede ser definido como una colección de números del mismo tipo. Los arreglos tienen una organización específica de datos. Esto es, puede ser un simple arreglo unidimensional de datos, y alternativamente, un arreglo puede ser

organizado como una matriz bidimensional, y así sucesivamente hasta 16 dimensiones que son el límite que marca Asyst.

Los arreglos como los escalares, tienen un tipo de dato asociado a ellos. Los tipos de datos de un arreglo definen el formato de almacenamiento rango y exactitud de los datos almacenados como elementos de un arreglo. Se pueden crear arreglos de diferente tamaño y tipo, alternativamente los arreglos pueden aparecer como arreglos temporales sin nombre que existen solamente en la pila, y además se puede crear un tipo especial de variable llamada "token", el cual puede ser un escalar o un arreglo de tamaño dinámico.

Antes de ser utilizado un arreglo debe ser primero definido con un único nombre, cuya única restricción es que sea menor que 32 caracteres de longitud y sin contener espacios. A continuación se describen las principales instrucciones utilizadas para la creación y manejo de arreglos.

Tipo **DIM** [dim1, dim2, ...] **ARRAY** *Nombre*
Localización: Sistema Base
Descripción: Especifica el tipo de arreglo, el número de dimensiones y sus tamaños y el nombre con el que lo identificaremos. El tipo de variable puede ser definido como: enteros, reales complejos de simple o doble precisión. Los números dim1, dim2, ... , especifican el tamaño por cada dimensión, es decir dim1 especifica el tamaño de la primera dimensión, dim2 especifica el tamaño de la segunda dimensión y así sucesivamente. Para poder identificar el arreglo definido se especifica un Nombre que sigue a la instrucción ARRAY.

CREATE.COPY *Nombre*
Localización: Sistema Base
Descripción: Es un método alternativo para crear un arreglo nuevo de datos. Usa un arreglo que se encuentra en el tope de la pila para declarar la organización, el tamaño, el tipo y precisión del nuevo arreglo, cuyo nombre se especifica con *Nombre. Create.Copy*; también almacena el contenido del arreglo de datos que se encuentra en el tope de la pila, en el nuevo arreglo creado.

Tipo **DIM** [dim1, dim2, ...] **UNNAMED.ARRAY**
Localización: Sistema Base
Descripción: Crea un arreglo de datos en forma temporal, el arreglo no tiene nombre y se coloca en el tope de la pila. *Tipo y DIM* son similares a la primera instrucción dada en esta sección.

[]COPY
Localización: Sistema Base
Descripción: Copia el número o arreglo en el tope de la pila, en un número o un arreglo sin nombre. El resultado debe tener el mismo tipo de datos y dimensiones en el caso de un arreglo que el original.

Tamaño Tipo **RAMP**
Localización: Sistema Base
Descripción: Crea un arreglo sin nombre uni-dimensional, que contiene un número de elementos especificado por *Tamaño*, y cuyo tipo de elementos se da por *Tipo*. El arreglo se inicia con valores que van: 1 como primer elemento, 2 en el siguiente, y así sucesivamente.

?ARRAYS
Localización: Sistema Base
Descripción: Lista todos los arreglos definidos actualmente con su tipo y organizaciones.

Nombre **[]RAMP**
Localización: Sistema Base
Descripción: Coloca en el arreglo de datos especificado por *Nombre*, enteros consecutivos comenzando con 1,2,..., hasta haber colocado el número correspondiente a todas las coordenadas de dicho arreglo.

TOKEN *Nombre*
Localización: Sistema Base
Descripción: Esta instrucción crea una variable que se identifica como *Nombre*. Puede tomar el valor de un escalar o un arreglo de datos, su mayor importancia radica en que permite definir arreglos dinámicos. Con esta instrucción solamente hace falta especificar que vamos a utilizar una área de memoria para *Nombre*, posteriormente puede crearse un arreglo con la instrucción RAMP (también puede ser un escalar) y en seguida asignarle *Nombre* a dicho arreglo para su identificación.

BECOMES> *Nombre*
Localización: Sistema Base
Descripción: Dicha instrucción permite asignar el "token" *Nombre* a un escalar o arreglo, dependiendo de lo que se encuentre en dicho momento en el tope de la pila. En caso de que sea un arreglo, el "token" *Nombre* tendrá la misma organización que la del arreglo que se encontraba en el tope de la pila. Es preciso mencionar que en un programa, el mismo "token" puede cambiar de tamaño y organización dependiendo de lo que se le asigne durante la ejecución del programa, es decir puede tomar varias formas y tamaños.

?TOKENS
Localización: Sistema Base
Descripción: Despliega una lista de todos los "tokens" definidos en el sistema.

CLEAR.TOKENS
Localización: Sistema Base
Descripción: Inicializa todos los "tokens" a enteros de simple precisión.

Nombre [n1, n2, ...]
Localización: Sistema Base
Descripción: Nos permite accesar un elemento específico en un arreglo de datos, ya sea para asignarle un nuevo valor o para realizar alguna operación con él.

Nombre **SUB[** origen, rango, incremento ; ...]
Localización: Sistema Base
Descripción: Crea un subarreglo del arreglo que se encuentra en el tope de la pila, y que en este caso se le identifica con *Nombre*. Solamente modifica el índice de los elementos del arreglo, los elementos del subarreglo no se separan físicamente del arreglo original. El número de dimensiones del subarreglo es el mismo que el del arreglo original, solamente el tamaño de las dimensiones cambia. El origen es el índice en donde el subarreglo comienza en una dimensión dada. El rango es el tamaño del subarreglo deseado, por omisión el rango es el tamaño máximo disponible cuando éste no se da explícitamente. El incremento es el tamaño del paso o el número de elementos en que se incrementará para considerar un nuevo elemento como parte del subarreglo, por ejemplo cada tres elementos, cuatro etc.. Cuando no se especifica, el incremento es de 1.

Nombre **XSECT[** índice1, índice2, ...]
Localización: Sistema Base
Descripción: Extrae un subarreglo con una dimensión más pequeña que el original identificado por *Nombre* o del que se encuentre en el tope de la pila. La lista de índices especifica los valores de las coordenadas que se desean. Si se da el símbolo ! en el lugar de alguna de las dimensiones se entenderá que se requiere de todos los elementos de esa dimensión.

Nombre **[] MIN**
Localización: Sistema Base
Descripción: Toma el número más pequeño de todos los elementos del arreglo y lo coloca en el tope de la pila. El arreglo puede identificarse con *Nombre* o se tomará el que se encuentre en ese momento en el tope de la pila.

Nombre **[] MAX**
Localización: Sistema Base
Descripción: Toma el número más grande de todos los elementos del arreglo y lo coloca en el tope de la pila. El arreglo de trabajo se identifica con *Nombre* o tomará el que se encuentre en el tope de la pila.

Nombre **[]SIZE**
Localización: Sistema Base
Descripción: Regresa el número total de elementos del arreglo identificado con *Nombre* y lo pone en el tope de la pila.

Operadores y Funciones Matemáticas

Asyst contiene operadores que nos permiten realizar operaciones aritméticas básicas y manejar cadenas de caracteres, así como algunas funciones matemáticas más complejas. Todos estos operadores y funciones trabajan con diferentes tipos de número, con tipos de números mezclados, con números complejos y algunos con escalares y arreglos.

Para poder usar estos operadores y funciones, se requiere de dar argumentos para realizar una operación, los cuales se colocan en la pila y a continuación la operación que se va a realizar. El resultado de la operación (en caso de que exista) se coloca en el tope de la pila para ser usado posteriormente, y los argumentos que utilizados desaparecen del tope de la pila. En el caso de que se estén utilizando arreglos de datos como argumentos para una operación, éstas se realizan elemento a elemento, cuando se tienen dos arreglos como argumentos, y si la operación permite realizar una operación con un escalar y un arreglo, se realizará del escalar con cada uno de los elementos del arreglo. A continuación daremos el formato y uso de estos operadores aritméticos y de algunas de las funciones matemáticas más importantes.

n1 n2 :=
Localización: Sistema Base
Descripción: Asigna el valor que se encuentra en el tope de la pila (n1) a la variable n2, es decir (n2 = n1).

s1 s2 ":="
Localización: Sistema Base
Descripción: Asigna el "string" s1 a la variable s2, la cual debió ser definida previamente como cadena de caracteres ("string").

n1 n2 +
Localización: Sistema Base
Descripción: Suma los dos valores que se encuentran el tope de la pila, (n1 + n2).

n1 n2 -
Localización: Sistema Base
Descripción: Resta los dos valores que se encuentran en el tope de la pila, (n1 - n2)

n1 n2 *
Localización: Sistema Base
Descripción: Multiplica los dos valores que se encuentran en el tope de la pila, (n1 * n2).

n1 n2 /
Localización: Sistema Base
Descripción: Divide los valores que se encuentran en el tope de la pila, (n1 / n2).

n1 n2 **
Localización: Sistema Base
Descripción: Eleva el número base n1 al exponente n2, ($n1^{n2}$).

n1 n2 **MIN**
Localización: Sistema Base
Descripción: Coloca en el tope de la pila el número menor de entre n1 y n2.

n1 n2 **MAX**
Localización: Sistema Base
Descripción: Coloca en el tope de la pila el número mayor de entre n1 y n2.

n1 **ABS**
Localización: Sistema Base
Descripción: Regresa el valor absoluto del número que se encuentra en el tope de la pila (n1).

n1 **NEG**
Localización: Sistema Base
Descripción: Regresa el negativo del número que se encuentra en el tope de la pila (n1).

n1 **INV**
Localización: Sistema Base
Descripción: Regresa el inverso multiplicativo del número que se encuentra en el tope de la pila (n1).

n1 **SGN**
Localización: Sistema Base
Descripción: Regresa el signo del número que se encuentra en el tope de la pila (n1).

n1 **SQRT**
Localización: Sistema Base
Descripción: Regresa la raíz cuadrada del número que se encuentra en el tope de la pila (n1).

n1 **EXP**
Localización: Sistema Base
Descripción: Regresa el valor de e^{n1} .

n1 **LN**
Localización: Sistema Base
Descripción: Regresa el logaritmo de base *e* del número n1.

n1 **LOG**
Localización: Sistema Base
Descripción: Regresa el logaritmo de base 10 del número n1.

n1 **FUNCIÓN.TRIGOHIP**
Localización: Sistema Base
Descripción: Regresa el valor de la función trigonométrica o hiperbólica que se especifica del número que se encuentra en el tope de la pila (función.trigohip n1), en este caso se reemplazará la palabra Función.Trigohip con cualquiera de las siguientes: SIN, COS, TAN, SEC, CSC, COT, ASIN, ACOS, ATAN, ASEC, ACSC, ACOT, SINH, COSH, TANH, SECH, CSCH, COTH, ASINH, ACOSH, ATANH, ASECH, ACSCH, ACOTH.

Interface de DOS y Sistema de Entrada/Salida (Manejo de la Pantalla y Teclado)
Asyst ofrece la opción de poder ejecutar comandos que operan como si se estuviera trabajando en ambiente MS-DOS. Estos comandos también pueden incluirse en programas realizados en Asyst y de manera interactiva. Para que los comandos puedan operan directamente con archivos o con directorios, existe una opción que invierte la sintaxis de los comandos, esto con el objetivo que durante la ejecución de un programa se puedan pedir directorios o archivos como variables y a continuación poder ejecutar una instrucción con ellos, la instrucción que se encarga de realizar dicha operación es DEFER>, la cual describiremos más adelante. Existe también la opción de darle salida a información por medio de la consola de video, de una impresora o de un archivo de texto, por lo cual el manejo de la pantalla es de suma importancia. También ofrece la opción de comunicarse con Asyst por medio del teclado en forma interactiva y cuando se está ejecutando un programa. A continuación daremos una lista de las principales instrucciones que se utilizan para usar estos comandos, y en las cuales se dará la sintaxis diferida (DEFER>) cuando la instrucción permita el uso de la forma alternativa de ejecución.

Nombre **DEFER>** *Instrucción*
Localización Sistema Base
Descripción: Cambia la acción de las instrucciones, difiriendo la ejecución de una instrucción hasta que pueda leerse un "string" de la pila de símbolos. Es entonces cuando ejecuta la instrucción con el parámetro o nombre que se encuentra en el tope de la pila de símbolos.

DIR *Directorio*
Directorio DEFER> DIR
Localización: Sistema Base
Descripción: Despliega una lista de todos los archivos que se encuentran en el directorio especificado.

DELETE *Nombre*
Nombre DEFER> DELETE
Localización: Sistema Base
Descripción: Borra el archivo que se le identifica por Nombre.

PRINT *Nombre*
Nombre DEFER> PRINT
Localización: Sistema Base
Descripción: Imprime el archivo especificado con Nombre en la impresora LPT1.

TYPE *Nombre*
Nombre DEFER> TYPE
Localización: Sistema Base
Descripción: Despliega el contenido del archivo *Nombre* en el dispositivo de salida especificado.

COPY *Nombre-Fuente* **TO** *Destino*
Nombre-Fuente DEFER> COPY Destino DEFER> TO
Localización: Sistema Base
Descripción: Copia el archivo llamado *Nombre-Fuente* a uno que llevará por nombre *Destino*.

RENAME *Nombre-Fuente* **TO** *Destino*
Nombre-Fuente DEFER> RENAME Destino DEFER> TO
Localización: Sistema Base
Descripción: Renombra el archivo llamado *Nombre-Fuente* con el nombre *Destino*.

"INSTRUCCIÓN" *Nombre-Directorio*
Nombre-Directorio DEFER> "INSTRUCCIÓN"
Localización: Sistema Base
Descripción: Esta "INSTRUCCIÓN" debe remplazarse por cualquiera de las siguientes: **MKDIR** para crear un nuevo directorio llamado *Nombre-Directorio*, **CHDIR** para cambiar del actual directorio al directorio *Nombre-Directorio*, y **RMDIR** para borrar un directorio llamado *Nombre-Directorio* el cual debe de estar vacío, es decir no contener archivos.

.DIR
Localización: Sistema Base
Descripción: Imprime en pantalla el nombre del directorio de trabajo actual.

"DIR
Localización: Sistema Base
Descripción: Regresa el nombre del directorio actual, como un "string" y lo coloca en el tope de la pila de símbolos.

BYE
Localización: Sistema Base
Descripción: Nos permite salir del ambiente Asyst y regresar al sistema operativo MS-DOS.

CR
Localización: Sistema Base
Descripción: Salta una línea de salida del actual dispositivo de salida.

BELL
Localización: Sistema Base
Descripción: Genera un pequeño sonido que es emitido por la PC.

n1 **SPACES**
Localización: Sistema Base
Descripción: Pone el número de espacios n1 dado en el tope de la pila en el dispositivo de salida activo.

Num_caracter_Código_ASCII **EMIT**
Localización: Sistema Base
Descripción: Pone el código ASCII especificado en el dispositivo de salida activo.

n1 n2 n3 n4 **WINDOW** *Nombre*
Localización: Sistema Base
Descripción: Define una nueva ventana de despliegue y se le identificará con *Nombre*. La ventana se definirá en normal.display o graphics.display dependiendo del modo que se encuentre activo en el momento de su definición. El número n1 especifica la línea del tope superior de la ventana, n2 la columna a la izquierda, n3 la línea que marca el tope inferior de la ventana, y n4 la columna que limita a la ventana por la derecha. Para activar una ventana basta con dar su nombre.

vb hb tl bl br tr **BORDER.CHARS**
Localización: Sistema Base
Descripción: Se utiliza para definir los seis caracteres que se usarán para dibujar los bordes que sirven de marco a la ventana que se encuentra activa. El número vb es el caracter en código ASCII que se usará para dibujar el borde vertical, el hb el borde horizontal, tl la esquina superior izquierda, bl la inferior izquierda, br la inferior derecha tr la superior derecha.

{BORDER}
Localización: Sistema Base
Descripción: Dibuja un marco alrededor de la ventana activa utilizando los caracteres definidos en la instrucción Border.Chars.

{DEF}
Localización: Sistema Base
Descripción: Pone en activo la ventana que por omisión es usada por Asyst en modo *normal.display*. Es definida con 25 líneas (0-24) por 80 columnas (0-79).

Columna Linea **GOTO.XY**
Localización: Sistema Base
Descripción: Se usa para colocar el cursor en las coordenadas especificadas por *Línea* y *Columna* en la ventana que se encuentra en uso en ese momento.

?COL
Localización: Sistema Base
Descripción: Regresa al tope de la pila la columna en la cual se encuentra el cursor actualmente, este número de columna es con respecto a la ventana que se utiliza en el despliegue de video.

?REL.COL
Localización: Sistema Base
Descripción: Regresa la columna en la cual se encuentra el cursor actualmente en relación a la columna izquierda que limita a la ventana en uso.

?ROW
Localización: Sistema Base
Descripción: Regresa la línea en la que se encuentra el cursor actualmente, esta línea es con respecto a la ventana que se utiliza en el despliegue de video.

?REL.ROW
Localización: Sistema Base
Descripción: Regresa la línea en la que se encuentra el cursor actualmente en relación a la línea que tiene por tope la ventana que se encuentra en uso.

INVERSE.ON
Localización: Sistema Base
Descripción: Pone en video invertido (invierte los colores de fondo y de escritura) para todo el texto de salida subsecuente . Solamente se usa en modo normal.

INVERSE.OFF
Localización: Sistema Base
Descripción: Regresa todo el texto de salida subsecuente, a el modo no invertido. Es el modo que se da por omisión.

SCREEN.CLEAR
Localización: Sistema Base
Descripción: Borra todo el texto que se encuentra en la ventana en uso.

#Color **FOREGROUND**
Localización: Sistema Base
Descripción: Se usa para cambiar de color los letreros o letras que se definen a continuación de dicha instrucción, sin cambiar el color de fondo de la pantalla. El #Color es el color

original que se desea cambiar; cada color tiene asignado un número entero que lo identifica y los cuales daremos más adelante.

#*Color* BACKGROUND
Localización: Sistema Base
Descripción: Al igual que la anterior se usa para cambiar el fondo de la pantalla, sin modificar el color del cursor y de los letreros que se escriban.

ECHO.ON
Localización: Sistema Base
Descripción: Permite emitir un eco al cargar un archivo de texto, en caso de que ocurra un error. Así permite verificar y seguir los errores en caso de que existan.

ECHO.OFF
Localización: Sistema Base
Descripción: Desactiva la instrucción anterior.

#INPUT
Localización: Sistema Base
Descripción: Espera un valor numérico del teclado y lo coloca en el tope de la pila, si hay éxito en la operación pone un condicional verdadero en el tope de la pila de símbolos. Si se ingresa un número no válido (algún otro caracter no numérico), pone un condicional falso en la pila de símbolos y no coloca nada en el tope de la pila de números.

****
Localización: Sistema Base
Descripción: Delimita el inicio de un comentario. Todo el texto que sigue a "\" y hasta el fin de la línea es ignorado.

KEY
Localización: Sistema Base
Descripción: Devuelve el código ASCII de alguna tecla dada. Si no se ha dado esa tecla en una operación desde el teclado, la instrucción espera hasta que una tecla sea activada desde el teclado.

PCKEY
Localización: Sistema Base
Descripción: Devuelve tanto el código ASCII de una tecla dada como un condicional falso, o una función de una tecla y un condicional verdadero.

?KEY
Localización: Sistema Base
Descripción: Revisa que una clave esté disponible en el "buffer" de la PC, y regresa un condicional verdadero si ésta existe.

?NORMAL.DISPLAY
Localización: Sistema Base
Descripción: Regresa un condicional falso o verdadero (True/False) a la pila de símbolos dependiendo del sistema de despliegue en el que se encuentre. Un valor verdadero (True) indica que el sistema está en modo normal.

?GRAPHICS.DISPLAY
Localización: Sistema Base
Descripción: Al igual que la anterior regresa un condicional falso o verdadero dependiendo de el modo de despliegue en que se encuentre. Un valor verdadero indica que el sistema se encuentra en modo gráfico.

Como mencionamos anteriormente los colores para el manejo de la pantalla, tanto en modo gráfico como en normal, se pueden mandar llamar tanto por su nombre como por un número entero que se les asigna para manejarlos más fácilmente. A continuación daremos una lista de los nombres y números enteros asignados a esos colores, en los casos en que solamente aparece el nombre en español es porque se activan sólo por el número que se les asignó.

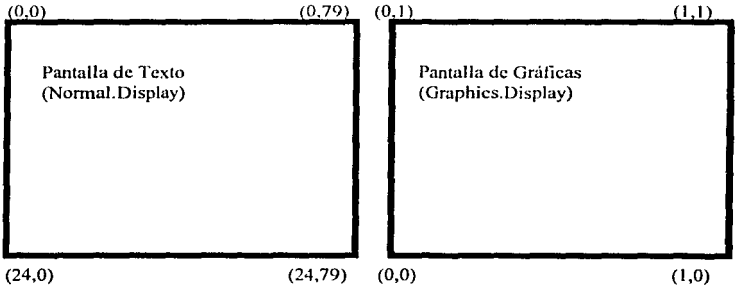
#Color	Color	#Color	Color
0	Black (Negro)	8	Grey (gris)
1	Blue (Azul)	9	Azul intenso
2	Green (Verde)	10	Verde intenso
3	Cyan (Violeta)	11	Violeta intenso
4	Red (Rojo)	12	Rojo intenso
5	Magenta (Magenta)	13	Magenta intenso
6	Brown (Café)	14	Yellow (Amarillo)
7	White (Blanco)	15	Blanco intenso

Gráficas
ASYST permite realizar gráficas tanto en una pantalla de video, como en una impresora. Maneja diferentes sistemas de coordenadas que permiten definir posiciones y tamaños de las mismas. Para el manejo de los tamaños de gráficas, maneja "*vuports*" que son áreas rectangulares de la pantalla de video o una región de papel sobre la cual se puede graficar.

Para realizar una gráfica se necesita por lo menos de un conjunto de datos que nos permitan representarlos en un sistema de coordenadas X, Y. Estos datos son presentados en forma de arreglos de datos, que pueden graficarse como un sistema (X,Y), en donde X representa el número que ocupa el elemento ?? en el arreglo y Y el valor del elemento ?? del arreglo, existe también la posibilidad de graficar dos arreglos que se encuentren en el tope de la pila en cuyo caso la coordenada X será el elemento tomado del primer arreglo

que entre a la pila y Y será tomada del segundo arreglo que entró a la pila, es decir, el que se encuentra en el tope de la misma.

Dentro del manejo de pantalla en modo normal se pueden especificar 25 líneas por 80 columnas, en donde para su identificación, la coordenada (0,0) es la que se encuentra en la parte superior izquierda de la pantalla, la (0,79) se encuentra en la parte superior derecha, la (24,0) en la parte inferior izquierda, y la (24,79) en la parte inferior derecha. Cuando nos encontramos en modo gráfico el cual se activa escribiendo el comando *graphics.display*, el manejo de la pantalla es diferente al anterior, ya que en este caso la pantalla de graficación se concibe como un elemento que mide 1 de largo por 1 de ancho, así pues, la esquina inferior izquierda se identifica con las coordenadas (0,0), la esquina superior izquierda con (0,1), la inferior derecha con (1,0), y la superior derecha con (1,1). Si deseamos hacer referencia al punto medio de la pantalla de graficación, éste será identificado por (.5,.5).



Se pueden manejar entonces, dos tipos de coordenadas en el modo gráfico, las coordenadas normalizadas (*normal coords*) y las coordenadas universales (*world coords*). Dichas coordenadas especifican una posición en un "vuport", y además la definición del mismo se hace mediante las coordenadas normales. Una vez definidos los "vuports" pueden activarse como área de trabajo con sólo escribir su nombre, y entonces pasarán a formar un todo en el modo gráfico, es decir, para identificar sus coordenadas se toma un área de graficación que va del (0,0) al (1,1) como se explicó anteriormente. En resumen las coordenadas normales, especifican un rango lineal de 0 a 1 y especifican una posición como una fracción de el ancho y largo definido para el "vuport". Las coordenadas universales dependen de los datos graficados, ya que son determinadas por el valor máximo y mínimo de cada eje de graficación. Así pues para identificar una coordenada de la gráfica, basta con tomar la intersección de el valor que aparece en el eje X de la gráfica con el valor de el eje Y correspondiente a la misma gráfica.

Existe la posibilidad de usar las gráficas en forma interactiva, es decir podemos tener acceso a las áreas de graficación, podemos pedir las coordenadas en las que se ha colocado el cursor, seleccionar pequeñas áreas de la gráfica y hacer acercamientos de las mismas (*zoom*). Para tener una mejor presentación de una gráfica cuando se va a mandar a impresión, también se tiene la opción de poder poner títulos sobre las gráficas después de que estas aparecen en la pantalla de video, esto para darnos la opción de hacer anotaciones sobre elementos de la gráfica cuando sea necesario.

A continuación daremos las principales instrucciones utilizadas para poder graficar en un sistema de coordenadas (X,Y), pero se debe tener presente que se pueden realizar gráficas en tres dimensiones y de pastel. Todas las instrucciones deberán de ejecutarse cuando el ambiente se encuentra en modo gráfico.

Arreglo **Y.AUTO.PLOT**
Localización: Sistema Base
Descripción: Automáticamente gráfica el arreglo que se encuentra en el tope de la pila tomando todos sus valores sobre el eje Y, contra el arreglo de índices que tomará los valores sobre el eje X. Así la primera coordenada será el número que un elemento tiene en el arreglo y la segunda el elemento mismo.

ArregloX ArregloY **XY.AUTO.PLOT**
Localización: Sistema Base
Descripción: Automáticamente graficará dos arreglos de datos que se encuentran en el tope de la pila. Los elementos del primer arreglo que entra a la pila (ArregloX) toman sus valores a lo largo del eje de abcisas (eje X) mientras los elementos del segundo que entra a la pila (ArregloY) toma sus valores sobre las ordenadas (eje Y).

Arreglo **Y.DATA.PLOT**
Localización: Sistema Base
Descripción: Se utiliza para graficar un arreglo que se encuentra en el tope de la pila. Se usa después de haber ejecutado un Y.AUTO.PLOT, es similar pero pinta la gráfica sobre la que se había realizado con anterioridad.

ArregloX ArregloY **XY.DATA.PLOT**
Localización: Sistema Base
Descripción: Es similar a XY.AUTO.PLOT pero pinta la gráfica sobre la que se había realizado anteriormente a ella.

NORMAL.COORDS
Localización: Sistema Base
Descripción: Pone el manejo de las gráficas en coordenadas normales, es decir en el sistema que maneja rangos de 0 a 1 y que se explicó anteriormente.

WORLD.COORDS
Localización: Sistema Base
Descripción: Pone el manejo de las gráficas en coordenadas universales, es decir se utilizan dependiendo de los datos que se han graficado.

HORIZONTAL
Localización: Sistema Base
Descripción: Nos permite realizar modificaciones a las abcisas (eje X).

VERTICAL
Localización: Sistema Base
Descripción: Nos permite realizar modificaciones a las ordenadas (eje Y).

AXIS.ON
Localización: Sistema Base
Descripción: Pone en activo la graficación de los ejes tanto X o Y, para las siguientes gráficas.

AXIS.OFF
Localización: Sistema base
Descripción: Desactiva la graficación de los ejes X o Y para siguientes graficaciones.

GRID.ON
Localización: Sistema Base
Descripción: Pone en activo la graficación de la gradilla tanto del eje X o el Y para las siguientes gráficas.

GRID.OFF
Localización: Sistema Base
Descripción: Desactiva la graficación de la gradilla para los ejes X o Y para las siguientes gráficas.

Xdiv Ydiv **AXIS.DIVISIONS**
Localización: Sistema Base
Descripción: Especifica el número de marcas utilizadas para dividir tanto la abcisa (Xdiv), como la ordenada (Ydiv).

#Color **AXIS.COLOR**
Localización: Sistema Base
Descripción: Se utiliza para escoger el color de los ejes y la gradilla que van a pintarse en las gráficas siguientes a la instrucción.

#*Color* LABEL.COLOR
Localización: Sistema Base
Descripción: Se utiliza para escoger el color de las etiquetas que van a pintarse en las gráficas siguientes a esta instrucción.

GRAPHICS.READOUT
Localización: Sistema Base
Descripción: Activa el uso de las flechas para mover el cursor a través de la gráfica que se encuentra en el "vuport" en uso. Permite activar las siguientes teclas de control para tener acceso al área de graficación.

<*Flechas*> Mueve el cursor a la derecha, izquierda, arriba o abajo, según sea la flecha que se ha activado. El incremento en el movimiento del cursor está dado por su previa selección.

<*Re-Pág*> Cambia el tamaño del incremento con el cual se van ha mover cualquiera de las flechas. Los incrementos varían de 1 a 9 y son proporcionales al número que los identifica.

<*Inicio*> Esta tecla tiene dos funciones. Si la instrucción readout>position se ha ejecutado, entonces las coordenadas actuales (X,Y) del cursor serán desplegadas en la posición que se indique, y serán desplegadas cada vez que la tecla <Inicio> sea oprimida. Si la instrucción readout>array ha sido ejecutada, entonces cada vez que la tecla sea oprimida, las coordenadas X y Y actuales del cursor deberán ser salvadas en un arreglo sin nombre.

<*Supr*> Esta tecla desactiva el uso de todas las teclas utilizadas en gráficas interactivas, es decir termina el uso de la instrucción GRAPHICS.READOUT .

ARRAY.READOUT
Localización: Sistema Base
Descripción: Permite el uso de gráficas interactivas para el arreglo de datos más recientemente graficado. Dos líneas verticales serán desplegadas en la gráfica que van de la línea de graficación al eje horizontal. Cada una o ambas líneas pueden ser movidas siguiendo los datos del arreglo graficado. En este caso solamente se pueden utilizar las flechas izquierda o derecha para moverlas. Las líneas utilizadas en este modo interactivo son las siguientes.

<*Fin*> Activa solamente la línea que se encuentra a la izquierda, y desactiva la que se encuentra a la derecha. Se puede mover con las flechas.
<*Av-Pág*> Activa solamente la línea que se encuentra a la derecha, y desactiva la que se encuentra a la izquierda. Se mueve con las flechas.
<*Flecha-Abajo*> Activa ambas líneas, tanto la derecha como la izquierda. Para mover las líneas será con las flechas, y las dos se moverán simultáneamente cuando se active una de las flechas.

<Flecha-Izquierda ó Derecha> Mueve ambas o una de las líneas según sea el caso a la derecha o a la izquierda, dependiendo de la flecha que se ha activado, y el incremento que se ha seleccionado.
<Re-Pág> Al igual que en la instrucción anterior cambia el incremento del cursor.
<Inicio> Tiene la misma función que en la instrucción anterior, donde muestra las coordenadas (X,Y) en las que se encuentra actualmente el cursor o las guarda en un arreglo según sea el caso.
<Insert> Permite graficar solamente el subarreglo que se encuentra entre las líneas derecha e izquierda. Vuelve a graficar esta región de la gráfica y se puede hacer uso de las teclas que aquí se definen, por lo que se le da el nombre de "Zoom".
<Supr> Esta tecla desactiva todas las teclas utilizadas en gráficas interactivas, es decir termina el uso de la instrucción ARRAY.READOUT.

NX NY **READOUT.POSITION**
Localización: Sistema Base
Descripción: Despliega las actuales coordenadas donde se encuentra el cursor, ya sea en modo de *graphics.readout* o *array.readout*. Esta información se despliegan en la posición NX, NY del "vuport" en uso, generalmente se dan en coordenadas normales (normal.coords).

Nom/#Color **COLOR**
Localización: Sistema Base
Descripción: Nos permite escoger el color con el cual se va a pintar la línea de graficación, se le puede llamar por su nombre o por su número asignado.

NX NY **POSITION**
Localización: Sistema Base
Descripción: Pone el cursor de la gráfica en la posición NX, NY. Estos argumentos pueden darse en coordenadas normales o en coordenadas universales.

"String" **LABEL**
Localización: Sistema Base
Descripción: Pinta la cadena de caracteres dado ("string") en la gráfica actual, y en la posición en la que se encuentre actualmente el cursor de la gráfica.

Ángulo **CHAR.DIR**
Localización: Sistema Base
Descripción: Especifica la dirección que tomarán los caracteres individualmente en la gráfica, esto permite darle presentación a las gráficas rotando el ángulo de los caracteres de una cadena. El ángulo es un número entero que va de 0 a 360 grados, por omisión su valor es 0.

Ángulo **LABEL.DIR**
Localización: Sistema Base
Descripción: Especifica la dirección que tomará una cadena de caracteres en su conjunto en una gráfica. El ángulo será de 0 a 360 grados.

VUPORT *Nombre*
Localización: Sistema Base
Descripción: Crea una nueva región de graficación ("vuport") llamada *Nombre*, al cual se le pueden dar o definir características propias y para hacer referencia a él sólo se escribe su *Nombre*.

VUPORT.CLEAR
Localización: Sistema Base
Descripción: Limpia el área de graficación o "vuport" que se encuentra activo en ese momento.

NXm NYm **VUPORT.ORIG**
Localización: Sistema Base
Descripción: Especifica el origen de un "vuport" usando coordenadas normales NXm y NYm, y tomando como base el área de graficación.

NXm NYm **VUPORT.SIZE**
Localización: Sistema Base
Descripción: Especifica el tamaño de un "vuport" usando coordenadas normales, NXm es el tamaño que tendrá en el eje X, mientras NYm en el eje Y. Todas estas son en relación al número 1 que es la cota de estas coordenadas tanto hacia arriba como hacia la derecha.

OUTLINE
Localización: Sistema Base
Descripción: Dibuja un marco alrededor de el "vuport " en uso.

?VUPORT(S)
Localización: Sistema Base
Descripción: Despliega el o los nombre(s) del "vuport(s)" en uso con sus medidas asociadas.

?CURSOR
Localización: Sistema Base
Descripción: Regresa el valor en coordenadas universales de la posición del cursor en la gráfica actual. Primero ingresa a la pila la coordenada en X y le sigue la Y.

#Color **VUPORT.COLOR**
Localización: Sistema Base
Descripción: Se utiliza para poner el color de fondo que tendrá el "vuport" que se encuentra actualmente en uso.

#Color **CURSOR.COLOR**
Localización: Sistema Base
Descripción: Se utiliza para especificar el color que tendrá el cursor en las gráficas.

Num_Imp Nombre **BD.PRINTER**
Localización: Interface de "Baby Driver", GENBD.EXE
Descripción: Selecciona el dispositivo de salida y el tipo de impresora. El dispositivo de salida dado como *Nombre* es una manera de identificar el dispositivo. PRN, LPT1, LPT2, LPT3, y AUX son reconocidos como dispositivos de salida. El Num_Imp deberá ser un entero que identificará el tipo de impresora. El archivo babydriv.lst contiene una lista de impresoras soportadas por "Baby Driver".

BD.SCREEN
Localización: Interface de "Baby Driver"
Descripción: Imprime la pantalla de graficación completa. Se debe estar en modo gráfico, Se devuelve un valor falso/verdadero ante algún error de estado, falso si fue exitosa la impresión o verdadero si no.

BD.VUPORT
Localización: Interface de "Baby Driver"
Descripción: Imprime lo que se encuentre en el "vuport" que está activo. Al igual que la anterior, se deberá estar en modo gráfico.

Orientación **BD.PAGE**
Localización: Interface de "Baby Driver"
Descripción: Selecciona la orientación de la gráfica para todas las subsecuentes salidas a impresión. Las opciones de orientación son "portrait" o "landscape". Por omisión se imprime en "portrait".

Hres Vres **BD.RES**
Localización: Interface de "Baby Driver"
Descripción: Ajusta la resolución vertical y horizontal en puntos por pulgada.

Hesc Vesc **BD.SCALE**
Localización: Interface de "Baby Driver"
Descripción: Ajusta las escalas vertical y horizontal para subsecuentes impresiones. Toma como referencia para sus argumentos (Hesc, Vesc) el tamaño de la imagen impresa en la pantalla de video. Así pues, si se desea una impresión con las magnitudes que se encuentran en video, tan sólo se dará Hesc=1000 y Vesc=1000, si se desea del doble 2000 etc.. No necesariamente deberá ser la misma escala vertical u horizontal, pudiendo variar de acuerdo al gusto o necesidad.

PORTRAIT
Localización: Interface de "Baby Driver"
Descripción: Selecciona la orientación de la gráfica en forma horizontal, es decir en una hoja tamaño carta al eje horizontal de la gráfica le corresponderá el lado menor.

LANDSCAPE
Localización: Interface de "Baby Driver"
Descripción: Selecciona la orientación de la gráfica en forma vertical, es decir al eje horizontal de la gráfica le corresponderá el lado más grande de la hoja.

2.4 Archivos de Datos

El manejo de archivos es muy importante para el manejo de la información arrojada por algún proceso o algún experimento científico. Sin embargo, es difícil que el "software" científico ofrezca las mismas facilidades para el manejo de archivos que las ofrecidas por el "software" enfocado a procedimientos administrativos. ASYST no es la excepción y presenta pequeñas dificultades en el manejo de archivos de datos, pues su diseño está enfocado primordialmente para el manejo de información numérica y limita el manejo de información utilizada en procedimientos administrativos. Para el manejo de archivos ASYST permite especificarle las características de los mismo, su organización y su número de elementos, a través de un formato especial que se conoce como plantilla de archivo (*file.template*). A los elementos del archivo se les llamará subarchivos y tendrán un número asignado que los diferenciará de los demás. Estos subarchivos son arreglos de datos que pueden ser grabados en un archivo como elementos individuales, al igual que cualquier otro escalar. Por otra parte ASYST permite transferir información de sus archivos a un archivo en Lotus 1-2-3 o Excell (archivos con extensión .wks y .wk1) y viceversa.

A continuación describiremos la principales instrucciones utilizadas para el manejo de archivos, para lo cual debe de estar cargado el "overlay" DATAFILE.SOV ya sea de forma permanente o transitoria.

FILE.TEMPLATE
Localización: Archivo "overlay" Datafile.Sov
Descripción: Inicializa la descripción de cómo se van a organizar los archivos. Es decir inicia el formato que describirá la estructura que tendrá el archivo. Todos los archivos que son creados después de esta instrucción tendrán la organización que ahí se describe.

#*Com* COMMENTS
Localización: Archivo "overlay" Datafile.Sov
Descripción: Especifica el número de comentarios (#Com) que va a contener el archivo. Un comentario será una línea de 64 caracteres. Si se omite el uso de esta instrucción, se entenderá que no se desean realizar comentarios.

Desc/#Sub **SUBFILE**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Especifica la organización (*Desc*) de un subarchivo individual en un archivo de datos, es decir, describe la organización de un elemento del archivo cuando se usa en conjunto con la instrucción *File.Template*. Cuando se usa fuera de una descripción de archivo, se usará para especificar el número de subarchivo que va a utilizarse.

#Sub **TIMES**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Especifica el número de subarchivos con organización idéntica que contendrá el archivo. Esta instrucción deberá escribirse inmediatamente después de *SUBFILE*.

END
Localización: Archivo "overlay" Datafile.Sov
Descripción: Pone fin a las especificaciones de un archivo iniciadas con *FILE.TEMPLATE*.

FILE.CREATE *Nom_Arch*
Nom_Arch **DEFER> FILE.CREATE**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Crea un archivo de datos con la organización especificada anteriormente a la ejecución de esta instrucción por *file.template*, e inicializa el contenido de los datos con cero. Requiere de un nombre para el archivo (*Nom_Arch*) que será una cadena de caracteres que puede darse inmediatamente después de la instrucción o estar en el tope de la pila de símbolos para el modo DEFER>.

FILE.OPEN *Nom_Arch*
Nom_Arch **DEFER> FILE.OPEN**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Abre un archivo especificado como *Nom_Arch* y permite accesar tanto a los comentarios como a los subarchivos que contiene. El nombre del archivo deberá darse inmediatamente después de ejecutar la instrucción o estar en el tope de la pila de símbolos en caso de utilizarse el modo DEFER>.

FILE.CLOSE
Localización: Archivo "overlay" Datafile.Sov
Descripción: Cierra el archivo que se encontraba en ese momento en uso o abierto.

?FILE.OPEN
Localización: Archivo "overlay" Datafile.Sov
Descripción: Regresa un valor falso o verdadero dependiendo si un archivo de datos está abierto o cerrado. Si está abierto regresa un valor verdadero, si no un valor falso.

FILE.SIZES *Nom_Arch*
Nom_Arch **DEFER> FILE.SIZES**
Localización: Sistema Base
Descripción: Regresa el número de archivos con el nombre *Nom_Arch* y el tamaño de los mismos. Se puede preguntar por un grupo de archivos o por uno solo utilizando los mismos parámetros que en MS-DOS.

#Com **COMMENT>**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Lee el comentario número *#Com* del archivo que se encuentra abierto en ese momento, y lo coloca en el tope de la pila de símbolos para su uso.

#Com **>COMMENT**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Lee la cadena de caracteres del tope de la pila de símbolos, y lo escribe o graba en el archivo que se encuentra abierto como el comentario número *#Com*.

FILE> UNNAMED.ARRAY
Localización: Archivo "overlay" Datafile.Sov
Descripción: Coloca un arreglo sin nombre en la pila, igual en tamaño, tipo de datos y contenido que el que se encuentra en el subarchivo especificado.

FILE.CONTENTS
Localización: Archivo "overlay" Datafile.Sov
Descripción: Regresa el número de comentarios y subarchivos que contiene el archivo de datos que se encuentra abierto. Estos datos se colocan en el tope de la pila.

?FILE.TEMPLATE
Localización: Archivo "overlay" Datafile.Sov
Descripción: Despliega el número de comentarios, el tipo de datos y el tamaño de cada subarchivo, y el número del subarchivo que se encuentra actualmente activado.

?FILE
Localización: Archivo "overlay" Datafile.Sov
Descripción: Despliega los comentarios, el formato del archivo, el número de subarchivos y sus características.

Arreglo **APPEND.ARRAY>FILE**
Localización: Archivo "overlay" Datafile.Sov
Descripción: Agrega o escribe un subarchivo o arreglo de datos en el archivo que se encuentra actualmente en uso. Este arreglo será tomado del tope de la pila y será el último elemento del archivo.

123FILE.CREATE *Nom_Arch*
Nom_Arch **DEFER> 123FILE.CREATE**
Localización: Archivo "overlay" 123IO.Sov
Descripción: Crea un nuevo archivo 1-2-3 que llevará por nombre *Nom_Arch*.

123FILE.OPEN *Nom_Arch*
Nom_Arch **DEFER> 123FILE.OPEN**
Localización: Archivo "overlay" 123IO.Sov
Descripción: Abre un archivo 1-2-3 creado previamente, cuyo nombre es *Nom_Arch*.

123FILE.CLOSE
Localización: Archivo "overlay" 123IO.Sov
Descripción: Cierra un archivo 1-2-3 que fue abierto previamente.

Lín Col **123WRITE.DOWN**
Localización: Archivo "overlay" 123IO.Sov
Descripción: Pone en modo de escritura un archivo 123 que ha sido abierto previamente, además de indicarle que el formato de grabación será por columnas, es decir hacia abajo. Se le indica en que línea y en que columna (Lin,Col) se debe de iniciar la transferencia.

Lín Col **123WRITE.ACROSS**
Localización: Archivo "overlay" 123IO.Sov
Descripción: Igual que la anterior pero el formato de grabación será por líneas.

Arreglo **ARRAY>123FILE**
Localización: Archivo "overlay" 123IO.Sov
Descripción: Escribe un arreglo (o número) en el archivo 123 que se encuentra actualmente abierto y en la dirección y coordenadas de inicio (Lín,Col), que se le especificaron anteriormente con las instrucciones correspondientes. El arreglo de datos lo tomará del tope de la pila.

2.5 Principales Estructuras de Programación

Es muy importante en todo lenguaje de programación, poder tener como herramientas instrucciones que nos permitan decidir cuándo una o varias instrucciones deberán ejecutarse dependiendo de una o varias comparaciones o estados del programa, así como el repetir un conjunto de instrucciones un número determinado de veces. A continuación se hablará y se describirán cada uno de estos procesos que nos permiten agrupar a un conjunto de instrucciones y cuya ejecución estará condicionada. Es preciso señalar que cada una de las sentencias de condicionamiento y ciclos (loops), solamente pueden usarse en una definición de dos puntos (colon definitions) de la cual ya hablamos anteriormente.

Operadores de Comparación

Las operaciones de comparación son utilizadas para tomar decisiones lógicas respecto a que rutas de ejecución del programa deben tomarse. Estas operaciones producen valores falsos o verdaderos en la pila de símbolos, los cuales pueden ser utilizados en conjunto con sentencias de condicionamiento y ciclos (loops). Para realizar dichas comparaciones se necesitan de operadores de comparación que nos permitan relacionar dos o más elementos. Es importante señalar que estos operadores de comparación también son válidos para comparar dos arreglos de datos o un escalar y un arreglo, tan sólo escribiendo los operadores entre corchetes **[operador]** .

n1 n2 =
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero a la pila de símbolos cuando *n1* es igual a *n2*, de lo contrario pondrá un valor lógico falso.

n1 n2 <
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero a la pila de símbolos cuando *n1* es menor que *n2*, de lo contrario pondrá un valor lógico falso.

n1 n2 >
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero a la pila de símbolos cuando *n1* es mayor que *n2*, de lo contrario pondrá un valor lógico falso.

n1 n2 <>
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero a la pila de símbolos cuando *n1* no es igual a *n2*, de lo contrario regresará un valor lógico falso.

n1 n2 <=
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero cuando *n1* es menor o igual que *n2*, de lo contrario regresa un valor lógico falso.

n1 n2 >=
Localización: Sistema Base
Descripción: Regresa un valor lógico verdadero a la pila cuando *n1* es mayor o igual que *n2*, de lo contrario regresará un valor lógico falso.

Comp1 Comp2 **AND**
Localización: Sistema Base
Descripción: Coloca un valor lógico verdadero en la pila de símbolos cuando la primera comparación (*Comp1*) y la segunda comparación (*Comp2*) son verdaderas, de lo contrario colocará un valor lógico falso. Lo que en realidad hace es que si encuentra dos valores lógicos verdaderos en la pila de símbolos los remplazará por uno solo, cualquier otra combinación (verdadero-falso, falso-verdadero, falso-falso) regresará un valor lógico falso.

Comp1 Comp2 **OR**
Localización: Sistema Base
Descripción: Coloca un valor lógico verdadero en la pila de símbolos cuando la primera o la segunda comparación (*Comp1, Comp2*) son verdaderas, de lo contrario regresará un valor falso. Si encuentra un valor verdadero en cualquiera de los dos o inclusive en los dos elementos que se encuentran en la pila de símbolos los remplazará por uno verdadero, si encuentra dos falsos los remplazará por uno falso.

Comp1 Comp2 **XOR**
Localización: Coloca un valor lógico verdadero en la pila de símbolos cuando una y solamente una de las comparaciones fue verdadera. En cualquier otra situación (falso-falso o verdadero-verdadero) se coloca un valor falso. Es un *OR* excluyente.

Comp1 **NOT**
Localización: Sistema Base
Descripción: Coloca un valor lógico verdadero en la pila de símbolos cuando la comparación (*comp1*) no es verdadera, de lo contrario colocará un valor falso. Coloca el complemento lógico del que se encuentre en el tope de la pila.

TRUE
Localización: Sistema Base
Descripción: Coloca un valor lógico verdadero en el tope de la pila de símbolos.

FALSE
Localización: Sistema Base
Descripción: Coloca un valor lógico falso en el tope de la pila de símbolos.

Sentencias de Condicionamiento
Cuando se llama a una definición de dos puntos, permite que se ejecuten una secuencia de instrucciones. Si embargo, en muchas ocasiones se deben escoger los caminos a seguir dependiendo de valores que la misma ejecución de las instrucciones van arrojando, por lo que se hacen necesarias las sentencias de condicionamiento que nos permitan realizar bifurcaciones en un programa o definición de dos puntos (subrutina). Para utilizar las sentencias de condicionamiento es necesario primero realizar alguna comparación o alguna otra operación que permita colocar en la pila alguno de los valores lógicos falso o verdadero.

(T/F)
IF

Palabras o Instrucciones (subrutinas) a ejecutar

THEN

Localización: Sistema Base
Descripción: Permite ejecutar un conjunto de instrucciones, palabras u otras definiciones de dos puntos en caso de que el valor lógico que se haya encontrado en el tope de la pila de símbolos sea verdadero. Estas instrucciones, palabras o definición de dos puntos son las que se encuentran entre IF y THEN. Una vez concluida su ejecución, prosiguen las instrucciones que se encuentran después de la palabra "then" o en caso de que el valor lógico encontrado haya sido falso, no se ejecutan las instrucciones que se encuentran entre IF y THEN y se continúa con las siguientes.

(T/F)
IF

Instrucciones o Palabras a ejecutar si se encontró un valor lógico verdadero

ELSE

Instrucciones o Palabras a ejecutar si se encontró un valor lógico falso

THEN

Localización: Sistema Base
Descripción: Permite ejecutar un conjunto de instrucciones en caso de que el valor lógico que se encontró en la pila de símbolos haya sido verdadero, estas instrucciones se encontrarán entre las palabras *IF* y *ELSE*, en caso de que se haya encontrado un valor lógico falso entonces se ejecutarán las instrucciones que se encuentran entre *ELSE* y *THEN*. Cuando alguno de estos dos conjuntos de instrucciones se ha ejecutado, se sigue con las instrucciones subsecuentes a la palabra *THEN*.

Val_Prueba
CASE
Valor1 **OF**
Instrucciones o Palabras a ejecutar en caso de que el valor a prueba (Val_Prueba) sea igual al Valor1 especificado.
ENDOF
Valor2 **OF**
Instrucciones o Palabras a ejecutar en caso de que el valor a prueba (Val_Prueba) sea igual al Valor2 especificado.
ENDOF
. . .
ValorN **OF**
Instrucciones o palabras a ejecutar en caso de que el valor a prueba (Val_Prueba) sea igual al ValorN especificado.
ENDOF
Instrucciones o Palabras a ejecutar en caso de que ninguna de las opciones anteriores haya sido valida. Si no se especifica ninguna instrucción, se entiende que se desea que no se realice ninguna operación:
ENDCASE
Localización: Sistema Base
Descripción: Permite dividir en casos un grupo de comparaciones, es decir, en lugar de realizar varios *IF*'s en un solo *CASE*, se pueden realizar estas comparaciones. Se toma el valor que se encuentra en el tope de la pila (Val_Prueba) y se va comparando con los valores especificados (*Valor1,...,ValorN*) y en cuanto uno de ellos sea igual a aquél entonces se ejecuta el conjunto de instrucciones que se encuentran entre *OF* y *ENDOF*, y ya no realizan mas comparaciones, sino que se ejecutan las instrucciones subsecuentes a la palabra *ENDCASE*. En caso de que ningún ValorX haya sido igual a *Val_Prueba* se ejecutan las instrucciones que se encuentran después del último *OF-ENDOF* si existen.

Se recomienda utilizar la instrucción *CASE* en vez de *IF-THEN* cuando se realizan comparaciones con la misma variable y se encuentran consecutivamente en un programa. *CASE* va comparando las variables, y cuando encuentra las equivalentes, ejecuta las instrucciones entre *OF* y *ENDOF* y sale de la estructura *CASE*. Por el contrario con *IF-THEN* se realizan todas las comparaciones aunque la válida ya se haya encontrado y se hayan ejecutado las instrucciones asociadas.

Ciclos de Instrucciones (Loops)
En muchas ocasiones, una o varias instrucciones de un programa necesitan repetirse un número determinado de veces, por lo que los lenguajes de programación deben suministrar instrucciones que permitan realizar ciclos de repetición de un conjunto de órdenes, que correspondan a una tarea específica. Los ciclos deberán estar condicionados a un número determinado de iteraciones o a una condición que puede tomar diversos estados durante el programa. Esta herramienta es muy importante en materias donde se realizan cálculos

numéricos extensos. A continuación describiremos los diferentes ciclos ((loops) que podemos utilizar en ASYST.

Máx 1 + Mín
DO

Instrucciones y Palabras que van a ejecutarse.

LOOP
Localización: Sistema Base
Descripción: Permite repetir un conjunto de instrucciones que se encuentran entre *DO* y *LOOP*, donde el límite del ciclo (loop) estará dado por el número de veces que deseamos que se ejecute el ciclo + 1 (*Máx 1 +*), ya que primero revisa que el valor no exceda el máximo deseado y luego ejecuta el ciclo. Al llegar al máximo verificará que no sea mayor que *máximo + 1* y también lo ejecuta, la siguiente iteración no la realizará. El ciclo empezará desde el número mínimo (*Mín*). Así el ciclo se ejecutará un número "Máx" de veces. Los límites con los cuales trabaja esta instrucción son tomados del tope de la pila, tomando primero el mínimo y luego el máximo.

Máx 1 + Mín
DO

Instrucciones y Palabras que van a ejecutarse.

incremento
+LOOP
Localización: Sistema Base
Descripción: Realiza exactamente la misma operación que la anterior, con la variante que nos permite especificar el incremento en cada iteración, y finalizará cuando después del último incremento se rebase el *máximo + 1*. El incremento puede ser negativo, en cuyo caso el valor máximo será un número menor a el mínimo especificado, para que cuando se le sume el incremento respectivo al mínimo establecido, éste decremente hasta llegar o rebasar el máximo (que en este caso en realidad es el mínimo).

BEGIN

Instrucciones y Palabras que van a ejecutarse

(T/F)
UNTIL
Localización: Sistema Base
Descripción: Permite repetir la ejecución de las instrucciones localizadas entre *BEGIN* y *UNTIL*, hasta que alguna condición establecida deje en el tope de la pila de símbolos un

condicional lógico verdadero. Esta estructura permite la ejecución del conjunto de instrucciones por lo menos una ocasión.

BEGIN

Instrucciones y Palabras ejecutadas por lo menos una vez y que generan un condicional lógico

(T/F)
WHILE

Instrucciones y Palabras ejecutadas hasta que la condición es falsa

REPEAT
Localización: Sistema Base
Descripción: Con esta estructura se ejecutarán primeramente las instrucciones entre *BEGIN* y *WHILE* sin ningún condicionamiento, pero durante su ejecución se generará un condicional lógico que será colocado en el tope de la pila de símbolos. En caso de que sea verdadero se ejecutarán las instrucciones que se encuentran entre *WHILE* y *REPEAT*, al termino de lo cual deberá regresar a ejecutar el grupo de instrucciones que se encuentran entre *BEGIN* y *WHILE*, para repetir nuevamente el procedimiento mientras sea verdadera la condición. En caso de ser falsa se saltará el grupo de instrucciones entre *WHILE* y *REPEAT* y seguirá ejecutando las instrucciones subsecuentes a *REPEAT*.

BEGIN

Instrucciones y Palabras ejecutadas.

AGAIN
Localización: Sistema Base
Descripción: Este es un ciclo infinito, ya que ejecuta todas las instrucciones que se encuentran entre *BEGIN* y *AGAIN* sin posibilidad de que sea interrumpido. Solamente termina si:
1. Ocurre un error
2. Una de las palabras entre *BEGIN* y *AGAIN* ejecutan las instrucciones *Abort, Exit,* o *Bye*.
3. Se activan las teclas *control-break.*

I
Localización: Sistema Base
Descripción: Es el contador de las iteraciones del ciclo (LOOP), el cual se incrementa automáticamente y al invocarlo regresa el número de iteración actual al tope de la pila. Solamente se utiliza con *DO-LOOP*

2.6 Adquisición de Datos

Sin duda alguna uno de los mayores atractivos que ofrece ASYST, es el manejo de la Adquisición de Datos. La Adquisición de Datos es un proceso mediante el cual una computadora captura información (voltaje) de fuentes externas (equipo de medición) utilizando un "hardware" especial. Dentro de este "hardware" especial se encuentra una tarjeta de adquisición, la cual recibe el voltaje proveniente de alguna fuente externa y lo digitaliza para ser manejado por la computadora. Para que la computadora pueda recibir esta información o voltaje, se necesita contar con un conjunto de aparatos especializados ("set") que permitan manipular (amplificar, filtrar, sincronizar, etc.) la información original, y hacerla compatible con la tarjeta de adquisición. Una vez captada la información, puede ser almacenada en la memoria de la computadora y el programador o usuario harán uso de ella cuando lo decidan.

Es preciso mencionar que también se pueden realizar conversiones digital/analógicas en las cuales la computadora es la fuente de información, y provee a la tarjeta de adquisición de información digital que convierte en voltaje (analógico) para ser suministrada a una fuente externa. ASYST permite también controlar estos procedimientos.

Para realizar Adquisiciones de Datos en ASYST, se necesita como ya lo mencionamos de una o varias tarjetas de adquisición. Las tarjetas de adquisición que pueden utilizarse en conjunto con ASYST son las siguientes:

- Series DT2800 y DT2820 de Data Translation Board.
- Series 500 y Series 570 de Keithley Data Acquisition Board
- Las tarjetas DASH16, DASH40, y DAS40 de Metrabyte Data Acquisition
- La RTI-800/815 y RTI-802 de Analog Devices Data Acquisition Boards
- La CYB.DACQ de Cyborg Data Acquisition Boards
- La IBM-DACA de IBM
- La Lab.Master, Lab.Tender y Tecmar.Dadio de Scientific Solutions Data Acquisition Boards
- La 8255.Port y IO.Port de General Purpose Data Acquisition Boards
- La WFS.Blank, Dataq.Waveform de Scoller Board

Para que ASYST utilice cualquiera de estas tarjetas de adquisición, debe instalarse previamente en la computadora en alguna de sus ranuras de expansión ("slots"). Hablamos de ellas en el Capitulo I (ASYST puede trabajar con más de una tarjeta). Además, debe contarse con un manejador del Sistema de Adquisición de Datos (external DAS Driver) que nos permita especificar con cuál tarjeta de adquisición estamos trabajando.

Un manejador externo del sistema de adquisición de datos consta de dos elementos de "software" que permiten al usuario o al programador usar una tarjeta en particular. Una parte del programa es el manejador, una colección de rutinas que controlan a la tarjeta. La otra parte se encarga de los soportes del manejador y los enlaces (hooks) con ASYST. Los

manejadores DAS se escriben en alguno de los lenguajes comerciales, se compilan y se ligan en archivos tipo ".exe" de DOS. Este código ejecutable lo usa ASYST para controlar la tarjeta de adquisición de datos. El manejador DAS se encuentra completamente desligado de ASYST y debe de ser suministrado por el proveedor.

El manejador DAS es un programa residente en MS-DOS y contiene rutinas específicas que ejecutan funciones acordes con el manejador definido para ASYST. Si se desea utilizar un manejador externo, deben cargarse las bibliotecas maestras para adquisición de datos y la biblioteca del manejador externo DAS (ACQX.SOV), con el propósito de especificar a ASYST la(s) tarjeta(s) de adquisición con la(s) que se trabaja. Es importante que el manejador externo sea cargado (ejecutado) desde DOS antes de entrar a alguna aplicación de ASYST, o de correr un sistema ejecutable. El manejador externo DAS especificado, permite operar en tres modos distintos tanto para conversiones A/D, D/A e I/O digital. Estos tres modos son:

- Sincrónico
- DMA
- Por Interrupciones

ASYST utiliza el modo sincrónico y DMA para conversiones A/D y D/A. Para Entrada/Salida digital (I/O), solamente utiliza el modo sincrónico. Se utiliza el manejador externo en modo sincrónico, para implementar tareas múltiples que utilizan el reloj de la tarjeta maestra de la computadora como fuente de interrupción. El modo de interrupciones usa un interruptor que se encuentra en la tarjeta de adquisición de datos. En modo DMA, como lo especificamos en el Capítulo I, se transfieren los datos adquiridos directamente a la memoria de la computadora, sin que estos tengan que pasar por el procesador de la misma.

Para la adquisición de Datos mediante ASYST, se necesita cargar la biblioteca *ACQUIS.SOV*, la cual contiene todos los comandos A/D, D/A e I/O digital, con excepción de las instrucciones específicas de los manejadores de adquisición de datos que en particular habrán de utilizarse. También debe de cargarse la biblioteca *ACQX.SOV*, que soporta las especificaciones del manejador externo tipo DAS.

ASYST permite realizar la adquisición de datos por medio de una estructura que se identifica como plantilla de adquisición (template). En esta plantilla de adquisición se especifican los canales que han de utilizarse para el muestreo (0, 15), la ganancia, qué reloj de muestreo se empleará, si la señal de inicio para la captura será externa o interna, el lugar en donde se almacenarán los datos adquiridos, la frecuencia de muestreo (tiempo de retardo entre un dato y el siguiente), y la forma en que se usará la memoria (cíclica o no-cíclica). En suma, la plantilla de adquisición especifica toda la información acerca de cómo desea el usuario que se realice el proceso de captura de los datos.

El usuario o programador puede definir una o varias plantillas de adquisición de acuerdo a sus necesidades. Para hacer uso de alguna de ellas sólo debe escribir su nombre y activarla. Si se trabaja con varios convertidores A/D, se debe especificar cuál tarjeta va a realizar el proceso de conversión.

A continuación se describirán las instrucciones más importantes que permiten la adquisición de datos. Es importante señalar que para realizar conversiones es suficiente con cambiar el texto A/D por D/A en las instrucciones que lo utilicen. Asimismo, si se desea trabajar con acceso directo a memoria (DMA), se debe agregar al inicio de cada instrucción el texto DMA, un punto y a continuación la instrucción.

Nom_Tarjeta
Localización: Archivo "overlay" Acqx.Sov
Descripción: Se encarga de especificar el manejador externo DAS que se utiliza, y por consiguiente la tarjeta de adquisición correspondiente. La palabra *Nom_Tarjeta* es remplazada por alguno de los nombres de tarjetas que puede controlar ASYST.

CanIni CanFin **A/D.TEMPLATE** *Nom_Plantilla*
Localización: Archivo "overlay" Acquis.Sov
Descripción: Crea una plantilla de adquisición que se identificará con *Nom_Plantilla*, utilizará los canales desde *CanIni* hasta el canal *CanFin* (en números enteros), cuando solamente se va a utilizar un canal el número se repite. La plantilla se define para el manejador externo tipo DAS que se encuentre activo en ese momento.

A/D.INIT
Localización: Archivo "overlay" Acquis.Sov
Descripción: Inicializa la plantilla de adquisición que se encuentra en uso actualmente.

A/D.IN
Localización: Archivo "overlay" Acquis.Sov
Descripción: Muestrea en los canales especificados por la plantilla activa, ingresando un valor digital por cada canal al tope de la pila. Esta instrucción solamente toma una muestra por canal y no mas.

nl **TEMPLATE.REPEAT**
Localización: Archivo "overlay" Aquis.Sov
Descripción: Asocia el factor de repetición *nl* a la plantilla de adquisición en uso. Este número va ligado al tamaño del arreglo ligado a la plantilla.

nl **COVERSION.DELAY**
Localización: Archivo "overlay" Acquis.Sov
Descripción: Define el intervalo de muestreo. Está dado por el número *nl* que especifica los milisegundos de retardo. En realidad es la velocidad de muestreo.

?CONVERSION.DELAY
Localización: Archivo "overlay" Acquis.Sov
Descripción: Regresa al tope de la pila el retardo entre muestras para la plantilla de adquisición y la tarjeta en uso.

DAS.INIT
Localización: Archivo "overlay" Acquis.Sov
Descripción: Inicializa todas las plantillas y tarjetas de adquisición en el sistema.

n1 **A/D.GAIN**
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica un factor de amplificación para las señales de voltaje capturadas según el formato de la plantilla de adquisición en uso. La ganancia está dada por el número *n1*, son posible los valores.....

EXT.CLOCK
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica el uso de un reloj externo para asignar el índice de muestreo en las conversiones realizadas con la plantilla activa.

EXT.TRIG
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica el empleo de un disparo externo para comenzar las conversiones realizadas con la plantilla en uso. El disparo externo es un pulso de voltaje que indica a la tarjeta que puede empezar a muestrear.

INT.CLOCK
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica que las conversiones realizadas con la plantilla en uso, deberán usar el reloj de la tarjeta para establecer el índice de muestreo.

INT.TRIG
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica que la tarjeta no espera un disparo externo para comenzar las conversiones con la plantilla en uso, se harán en forma inmediata.

Arreglo **A/D.SCALE**
Localización: Archivo "overlay" Acquis.Sov
Descripción: Convierte los valores contenidos en Arreglo a unidades físicas tales como voltios. Los valores contenidos en Arreglo son producto de conversiones A/D. El resultado al cambiar la escala, lo coloca en el tope de la pila en forma de un arreglo sin nombre. También puede realizar la conversión a voltios de un solo número entero tomado del tope de la pila y el resultado lo regresará a la pila.

?DAS.TEMPLATES
Localización: Archivo "overlay" Acquis.Sov
Descripción: Despliega toda la información sobre el estado de todas las plantillas de adquisición definidas. La que se encuentra en uso se despliega de manera luminosa.

?DAS.TEMPLATE
Localización: Archivo "overlay" Acquis.Sov
Descripción: Despliega la información sobre el estado de la plantilla en uso.

Nom_Arreglo **TEMPLATE.BUFFER**
Localización: Archivo "overlay" Acquis.Sov
Descripción: Asocia el arreglo que lleva por nombre *Nom_Arreglo* a la plantilla de adquisición que se encuentra en uso. El arreglo debe ser de tipo entero.

A/D.IN>ARRAY
Localización: Archivo "overlay" Acquis.Sov
Descripción: Muestrea en los canales especificado por la plantilla activa una o más veces (tantas como sean dadas por el factor *TEMPLATE.REPEAT*), y regresa todos los valores a elementos consecutivos del arreglo designado en *TEMPLATE.BUFFER*.

CYCLIC
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica que alguna operación en la plantilla activa, procederá como un ciclo continuo, es decir, el índice regresa a su valor inicial después de que ha alcanzado su valor máximo.

NO.CYCLIC
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica que alguna operación en la plantilla activa, se detendrá cuando alcance su valor máximo.

n1 **MSEC.DELAY**
Localización: Archivo "overlay" Acquis.Sov
Descripción: Especifica el tiempo que debe tardar la tarjeta entre una muestra y otra. Este tiempo es especificado en milisegundos por *n1* y usa el reloj de la PC.

REL.TIME
Localización: Archivo "overlay" Acquis.Sov
Descripción: Regresa el valor que tiene actualmente el tiempo, y lo coloca en el tope de la pila. Este número será un número real de doble precisión en unidades de milisegundos.

2.7 Creando un Sistema Ejecutable o de Corrido Autónomo (Run-Time System)

Un sistema de corrido autónomo en ASYST, es un conjunto de programas de ambiente que ejecuta las aplicaciones creadas ("software"), pero no puede usarse para desarrollar nuevos programas o definir palabras. Un sistema de corrido autónomo es solamente una aplicación y no puede ser modificado por el usuario final. Se trata de un sistema que utiliza todos los elementos de la versión configurada con la que fue creado (bibliotecas, variables, arreglos, archivos, templados y definiciones de dos puntos), los

cuales se encuentran relacionados entre si para resolver una tarea especifica. Este "paquete" relaciona a los elementos o subprogramas por medio de una definición de dos puntos, que hace las veces de programa principal, y a su vez puede llamar a otras "palabras" que son definiciones de dos puntos (subrutinas), constituyendo así un sistema que resuelve un problema especifico.

Los sistemas de corrido autónomo son de gran utilidad cuando la persona que será el usuario final no tiene conocimientos ni de ASYST ni de programación en general, entonces un programador construye las rutinas necesarias para resolver la tarea que se plantea, creando un sistema autónomo que el usuario maneje de forma fácil y eficiente. Otra gran ventaja de los sistemas autónomos se presenta cuando se realizan trabajos similares en diferentes lugares o laboratorios, entonces se desarrolla sólo un sistema de corrido y se copia a todos los lugares donde se necesite, pagando las licencias correspondientes.

Al crear un sistema de corrido en ASYST, no deben usarse algunas instrucciones en los programas o definiciones de dos puntos que son utilizadas por el sistema de corrido. Esto es debido a que la mayoría de ellas hacen uso del directorio de ASYST tanto para consultar como para borrar las palabras que en el se encuentran, por lo que ocasionarían un conflicto en el sistema de corrido ya que este liga a todas las definiciones de dos puntos y palabras que se encuentran en el diccionario en el momento de su creación, y al borrarlas estaríamos borrando una parte del propio programa. Las instrucciones son:

"Exec	Start.Symbol.Table.Compaction
In.Overlay.File>	Keep
Find.Word>	End.Symbol.Table.Compaction
Array.Name>"	?Words
Forget	?Words/A
Forget/Clr	?Word>
Forget.All	?Word>/A

Un sistema de corrido autónomo debe ser diseñado con ciertos elementos y estructura para operaciones propias.

ASYST también ofrece la opción de que el usuario realice sus propias pantallas de presentación al ejecutar un sistema de corrido autónomo. Las instrucciones que utilizan para la realización de estas pantallas son:

Cr	Inten.On
."	Inten.Off
Inverse.On	{Border}
Inverse.Off	Algún nombre de ventana predefinida

El formato general para las pantallas de presentación en los sistemas de corrido autónomo está dado por las siguientes instrucciones:

BANNER: *Nombre*
Palabras_ Pre-Definidas
;BANNER

Para poder crear un sistema de corrido autónomo, hay que tener en el diccionario de ASYST todas las palabras que serán de utilidad, incluyendo una definición de dos puntos que constituirá el programa principal encargado de ligar todas las palabras empleadas. En seguida se debe de cargar la biblioteca *RUNTIME.AOV*, y se debe verificar que el archivo Asyst.MSG se encuentre en el directorio de trabajo. Una vez cumplido lo anterior, se dan las siguientes instrucciones:

INSTALL *Palabra_Maestra* **IN TURNKEY**
MAKE.TURNKEY *Nom_Sist-Corrido*

Con estas instrucciones se crea un sistema de corrido autónomo que llevará por nombre *Nom_Sist_Corrido* y arrojará dos archivos, uno con extensión .COM y otro con extensión .OVL. Lo que sucede es que *Palabra_Maestra*, que es en realidad el programa principal, se liga al nombre del sistema de corrido por medio de la instrucción *INSTALL*, a continuación se crea y nombra el sistema de corrido.

CAPITULO III

SISTEMA PARA ANÁLISIS Y ADQUISICIÓN DE DATOS ELECTROFISIOLÓGICOS

1. CONSTRUCCIÓN DE SISTEMAS (INGENIERÍA DE SOFTWARE)

Uno de los grandes problemas a los que se enfrentan las personas que desarrollan paquetes de programación destinados a un uso específico ("software"), es realizar programas y sistemas que sean comprensibles, que puedan modificarse fácilmente, y que además sean confiables. Lograr todo ello representa grandes ventajas, pues los programas que componen un sistema desarrollado así, pueden ser entendidos con facilidad por otros programadores y ser modificados cuando sea necesario; además estos mismos programas pudieran ser aprovechados en otros sistemas. De ahí la necesidad de contar con algún método que permita obtener todas estas características, y no requiera tomar en cuenta el área para la cual se desarrollan los programas y sistemas, el "hardware", ni al destinatario final. A continuación hablaremos de los principios que se siguieron en el sistema de análisis y adquisición de datos electrofisiológicos.

1.1 Objetivo y Características en la Construcción de un Sistema

En el desarrollo de un sistema deben tenerse en cuenta diversos factores, por un lado, el óptimo aprovechamiento de todos los recursos con que se cuenta, y por otro, sus limitaciones. Esto es, el sistema debe cumplir con su objetivo en el contexto real en el que se desarrolla, empleando al personal y equipo del que se dispone.

De esta manera podemos decir que la construcción de un sistema de cómputo debe cumplir con el siguiente objetivo:

- Satisfacer la función o funciones para las cuales fue creado, haciendo uso óptimo de los recursos tanto materiales como humanos con que se cuenta, y teniendo las características de ser comprensible, modificable y confiable[1].

Es muy común que un sistema de cómputo cumpla sólo con satisfacer las funciones para las cuales fue creado, que los programas cumplan con los objetivos para los cuales fueron creados; pero que el sistema no sea comprensible o modificable. Para que un sistema

[1] Pressman Roger S., "Ingenieria de Software, Un enfoque Práctico" Mc Graw Hill, Segunda Edición, España 1988

alcance todas las cualidades mencionadas, los subprogramas que lo componen deben de cumplir con las siguientes características:

- Deben ser fáciles de utilizar y no contener errores lógicos. Los programas no deben generar conflictos.
- Deben ser programas sencillos. No hay que complicar la lógica utilizada cuando no lo amerite.
- Deben ser programas legibles. Los nombres de variables, constantes, arreglos, funciones y procedimientos, deben hacer referencia al trabajo o utilización que se les está dando, para identificarlas fácilmente en cualquier parte del programa.
- Deben ser programas fáciles de modificar. Los programas deben estar estructurados de tal manera, que al modificarlos no haga falta realizar cambios en todo el programa, sino en lugares o bloques específicos.

Entre los problemas que pueden presentarse durante el desarrollo de un sistema, se encuentra el equipo de cómputo con el que se va a trabajar. Se debe tener en cuenta siempre, que los equipos tienen limitaciones: en su capacidad de memoria, velocidad de proceso, etc.

Otros aspectos a considerar, tanto en la creación de un sistema como durante su uso, son los siguientes:

a) El tiempo necesario para desarrollarlo y su costo.
b) Mantenimiento y costos de operación
c) Posibilidad de hacer cambios, claridad y confiabilidad
d) Grado de complejidad
e) Realización de los objetivos que se plantearon para su creación

Hasta el momento hemos hablado de que un sistema debe ser comprensible, modificable y confiable, pero no hemos precisado el significado de estos términos, a continuación daremos el significado de cada uno de ellos.

COMPRENSIBLE

Para decir que un sistema es comprensible, fácil de entender, hace falta que cumpla las siguientes condiciones[2]:

- La documentación del sistema debe ser completa y bien estructurada.
- Los programas que lo componen, legibles.
- Su lógica debe ser sencilla y clara, salvo donde se amerite lo contrario.
- Los archivos que maneje deben contener información clara y ordenada.

[2] Softtek, Educación y Tecnología, "Técnicas de Ingeniería de Software aplicadas a Programación" México 1996

Que un sistema sea comprensible es de mucha utilidad, pues propicia las condiciones para futuros cambios, expansiones o adaptaciones a otro equipo de cómputo. Esto es de suma importancia para poder seguir los constantes adelantos que ocurren tanto en el equipo de cómputo ("hardware"), como en los compiladores de lenguajes de programación y demás programas de aplicación ("software").

MODIFICABLE
Las necesidades de los usuarios son dinámicas, y los sistemas debieran permitir ajustes a esos cambios. Para que un sistema sea modificable se requiere que:

- Los programas desarrollados para el sistema y los que manejan información sean flexibles, es decir, acepten códigos nuevos de programación, así como la eliminación de parte de ellos, sin perder su eficiencia.
- Los archivos que contienen la información puedan sufrir cambios, sin alterar la información que contenían originalmente .

Sin duda alguna, un recurso que facilita las posibilidades de modificar un sistema de programación, es darle una estructura modular, pero de ello hablaremos más adelante.

CONFIABLE
Seguramente esta característica es la que más le importa al usuario, ya que se refiere a la veracidad de los resultados arrojados. Para que un sistema se considere confiable debe cumplir con las siguientes propiedades:

- Sin fallas o el mínimo de ellas.
- Capaz de detectar errores e identificarlos.
- Manejar información válida.
- Fácil de rastrear. En caso de detectarse errores de diseño o de lógica, poderlos corregir fácilmente.

1.2 Diseño y Desarrollo de Programas

Como hemos visto, es de suma importancia desarrollar programas eficientes para un sistema de cómputo. En esta sección planteamos algunos principios básicos que nos permiten desarrollar programas confiables, modificables y eficientes, que además sean sencillos y entendibles.

Para el desarrollo de programas de cómputo es muy importante hacer uso de la programación estructurada, con el objeto de utilizar lógicas sencillas, comprensibles y modificables. El uso de la programación estructurada permite que la estructura del programa modele la estructura natural del problema. Para alcanzar este objetivo podemos hacer uso de varias herramientas:

ESTRUCTURAS DE CONTROL

Las estructuras de control determinan el flujo de información durante la ejecución del programa, este flujo debe ser simple y visible, de ser posible con tan sólo una entrada y una salida. Las estructuras de control que son válidas en el uso de la programación estructurada son las siguientes:

- Desarrollo lineal (sequence). Ejecuta instrucción tras instrucción, sin saltos en la ejecución del programa.
- Estructura IF-THEN-ELSE.
- Clasificación en casos (CASE).
- Estructura WHILE-DO.
- Estructura REPEAT-UNTIL.
- Ciclos (LOOPS).
- Alguna Salida Anormal

MODULARIDAD

Llamaremos modularidad al proceso de dividir el programa en varios grupos de instrucciones, los módulos. El lenguaje de programación debe permitir ejecutar cada uno de los módulos de manera independiente, en ocasiones es suficiente nombrarlos para que esto ocurra. Cada módulo consta de un conjunto de instrucciones, algunas de las cuales pueden ser llamadas a nuevos módulos, entonces el programa ejecuta el primero de los módulos al cual se hace referencia durante la ejecución del programa, y se pueden crear ejecuciones anidadas. Un módulo puede ser llamado a ejecución desde uno o varios módulos mas, a veces su ejecución puede ser recursiva, es decir, puede llamarse a si mismo.

Para que exista un buen diseño modular, cada módulo debe tener una sola función completa. La división en módulos se hace de acuerdo a la estructura lógica del problema a resolver, así se consigue además aislar los efectos de las modificaciones posteriores que se realicen al programa. Si la función que va a desempeñar un módulo es muy extensa, se divide en subfunciones, sin alterar su propósito original, buscando que su tamaño no complique su comprensión.

La comunicación entre módulos debe de ser por medio de llamadas claras y bien justificadas. La relación entre módulos es muy importante, dos módulos con funciones totalmente ajenas no deben llamarse entre sí; deben tenerse bien definidos los módulos que en conjunto realizan una tarea específica, para no cruzarlos con otros en el programa general.

Al crear un módulo, la definición de este debe especificar de manera clara su nombre, la función que realizará, así como cuál es su interface externa con otros módulos, es decir que es lo recibe de entrada y que es lo que regresa de salida. No es fácil determinar cuando un diseño modular es mejor que otro

Existe un tipo de módulos que son utilizados desde cualquier parte del programa y que pueden ser ejecutados por cualquier otro módulo, se les llama *módulos compartidos.* Generalmente se ocupan de realizar funciones que deben repetirse continuamente a lo largo de todo el programa o de todo el sistema, generalmente realizan funciones como pedir algún valor numérico, alguna cadena de caracteres, validar la existencia de información, etc.

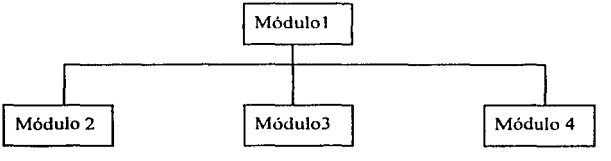
Una buena organización de los módulos a usar en un programa, se puede lograr con un diagrama que represente la jerarquía de los módulos y las relaciones entre ellos, a este diagrama le llamaremos *árbol de módulos.* En el árbol de módulos cada uno de ellos se representa con un rectángulo, y tiene conectados hacia abajo los módulos que dependen exclusivamente de él. En la parte inferior se representan de manera independiente y sin tener conexión hacía arriba, los módulos compartidos.

DISEÑO DESCENDENTE (TOP-DOWN)

En esta metodología para estructuras de control se propone resolver el problema abordándolo desde sus aspectos más generales hacía los particulares.

Los módulos que se utilizarán en el programa se diseñan ignorando sus detalles, es decir, en esta etapa lo importante es qué harán los módulos, no cómo lo harán. Usando el diseño "top-down" solamente se diseñará con detalle un módulo a la vez, junto con su comunicación con los otros que estamos definiendo. Como todos los módulos se diseñan en forma semejante, se llega por último a los módulos que sólo contienen instrucciones y llamadas a módulos compartidos.

Esta metodología es útil para cualquier tipo de problema, particularmente con los que no conocemos a fondo o que son muy extensos. Nos permite esbozar cómo ir resolviendo el problema, incluso podría dividirse el trabajo entre varias personas y reunir después las soluciones parciales del problema global (ver diagrama).



Ejemplo : El presente diagrama muestra la relación jerárquica entre módulos utilizando el diseño descendente (top-down), en donde el módulo 1 es el que se describe a detalle y los módulos 2, 3, y 4, solamente se definen sin entrar a detalles, indicando tan sólo lo qué harán, no como lo harán. Al terminar de diseñar el módulo 1 a detalle, se procede a diseñar en forma semejante cada uno de los módulos que se definieron (2, 3, 4), hasta llegar a los módulos que sólo tienen instrucciones.

DISEÑO ASCENDENTE (BOTTOM-UP)

El diseño ascendente es una metodología que aprovecha las ideas de modularidad y de estructuras de control. Aborda el diseño resolviendo primero los aspectos particulares y yendo hacia los generales.

Se identifican primero las funciones básicas o elementales que nos ayudarán a resolver el problema en general. Estas funciones básicas generalmente sólo contienen instrucciones, y no llamadas a otros módulos; una vez identificadas, se define un módulo para cada función. Se diseña entonces cada uno de estos módulos a detalle y su comunicación con los demás. En seguida se identifican otras funciones que requieran de los módulos que ya han sido diseñados, se define uno que haga llamadas a los módulos con los cuales está relacionado y que en conjunto realizarán una tarea específica. De esta manera se van atacando las funciones identificadas hasta lograr un módulo que tenga como función solucionar el problema planteado.

Esta metodología puede acarrear ciertos problemas, como llevar a definir módulos inútiles. Este tipo de diseño puede servir en problemas muy bien conocidos, lo cual no sucede normalmente. Es particularmente útil si se usa de manera combinada, es decir, parte del diseño es descendente y parte ascendente. Su uso continuo e indiscriminado puede acarrear conflictos en la solución del problema.

2. SISTEMA PARA ADQUISICIÓN Y ANÁLISIS DE SEÑALES FISIOLÓGICAS

Hemos hablado acerca de diferentes métodos para realizar adquisiciones de datos de señales externas a la computadora mediante una interface de laboratorio (Capitulo I). Se ha señalado que esto se hace a través de una tarjeta que transforma señales de voltaje a información digital o viceversa, también hemos dicho que ASYST es un "software" científico, y que una de sus especialidades es precisamente el uso de tarjetas para adquisiciones de datos. Además, en el presente capítulo se dieron algunos principios acerca de como puede desarrollarse un sistema de cómputo de manera eficiente, tanto en el manejo de los recursos materiales, como al realizar los programas necesarios.

El objetivo de este trabajo fue desarrollar un sistema de cómputo que permite realizar las siguientes funciones:

- Capturar datos experimentales, producto de registros electrofisiológicos, mediante una tarjeta de adquisición.
- Almacenar los datos obtenidos en archivos, para posteriormente analizarlos, sumando, restando, dividiendo, superponiendo, o calculando áreas de registros de señales electrofisiológicas, así como midiendo tiempos y amplitudes.
- Graficar los registros, así como el resultado de las operaciones realizadas entre ellos, con los títulos y leyendas que desee el usuario.
- Transferir los archivos generados en ASYST a archivos compatibles con otros programas, como Lotus o Excell.

2.1 Requerimientos del Sistema

El sistema requiere de un conjunto de instrumentos que permitan que las señales lleguen a la tarjeta de adquisición en condiciones apropiadas para la captura. Ya hemos hablado de que es muy importante utilizar de manera eficiente los recursos materiales con los cuales se cuenta, es por eso que en esta sección hablaremos de los diversos instrumentos, las características de cada uno de ellos, y la función o funciones que desempeñan. El siguiente esquema indica los instrumentos que son necesarios para la digitalización de las señales electrofisiológicas por medio de la computadora, junto con la relación entre estos instrumentos (*fig. 2*).

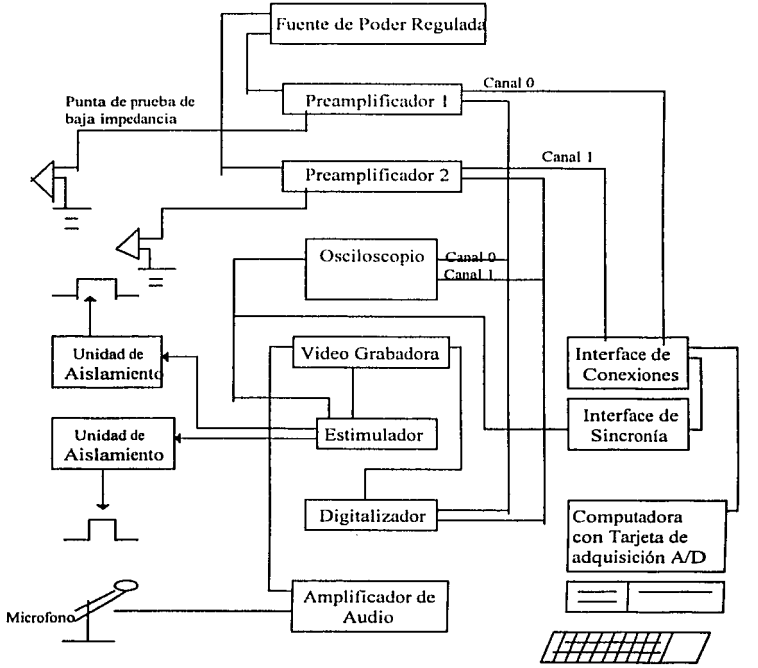


Fig. 2. Diagrama del montaje experimental que muestra los diversos instrumentos de registro y sus conexiones, necesarios para la digitalización de señales electrofisiológicas.

FUENTE DE PODER REGULADA
Suministra la energía necesaria a los preamplificadores. Marca GRASS RPS 107.

PREAMPLIFIFICADORES
Amplifica las señales de voltaje recibidas, según las necesidades del usuario. Generalmente en un rango de 2 000-10 000 veces, con una calibración para las señales de entrada de 500 µvoltios. Incluye dos filtros, pasa-bajas y pasa-altas. Marca GRASS, serie P5. Se cuenta con dos preamplificadores con salida en paralelo hacia el osciloscopio y hacía la interface de conexiones.

OSCILOSCOPIO
Nos permite un registro continuo de la señal para ajustar los equipos y eliminar posibles fuentes de ruido eléctrico. Marca Techtronic T912 a 10 Mhz.

ESTIMULADOR
Genera pulsos eléctricos que se dirigen hacía la preparación biológica para estimularla y obtener la respuesta fisiológica y detectarla mediante los preamplificadores. Se puede regular la frecuencia del pulso, su duración, y el voltaje. Marca GRASS S88.

VIDEO GRABADORA
Se usa para grabar las señales generadas y realizar su análisis posteriormente. La video grabadora también puede emplearse para transmitir a la computadora las señales grabadas previamente. Recibe o transmite (in-out) la señal del/al digitalizador según lo operación en curso (grabar o reproducir), también recibe los pulsos de sincronía suministrados por el estimulador, y la voz en caso de estar realizando una grabación.

DIGITALIZADOR
Se trata de un modulador de pulsos. Convierte la señal analógica a digital o digital a analógica para posteriormente ser grabada o reproducida por la vídeo grabadora, según sea el caso. Este digitalizador se usa solamente cuando las señales van a ser grabadas o reproducidas por la vídeograbadora. Si sólo se van a grabar las señales en la computadora, el uso del digitalizador no es necesario.

AMPLIFICADOR DE AUDIO
Se usa solamente cuando las señales van a ser grabadas por la vídeo, en cuyo caso se encarga de amplificar y grabar la señal de audio en una pista de la cinta de vídeo.

INTERFACE PARA SINCRONÍA
Como ya se explicó en el Capítulo I, el inicio del registro digital debe obtenerse sincronizándolo con un evento externo, como puede ser la estimulación de un nervio o simplemente un comando de cambio de voltaje. Esta interface liga el comienzo del reloj de muestreo A/D (el de la tarjeta) a un pulso de entrada llamado disparo externo (external trigger). La tarjeta de adquisición DAS40-G2 detecta un disparo externo al cambiar una señal de voltaje desde un nivel alto a uno bajo (High a Low). El voltaje constante que se le suministra es de 5 voltios, el cual debe de disminuir hasta 2.5 Volts mínimo para detectarlo

como disparo externo. Este estándar (protocolo TTL) se estableció para evitar que un ruido eléctrico puda disparar accidentalmente el sistema, es muy poco probable que por efecto del ruido ocurra un cambio desde un nivel alto hacia abajo. Para cumplir esta condición, el usuario se obliga a alimentar un voltaje constante de 5 volts, y el disparo ocurrirá solamente cuando disminuya hasta los 2.5 volts.

Dado que los equipos más viejos no incluyen este estándar para los pulsos de sincronía, se hace necesario construir una pequeña interface, un circuito que se encarga de montar el voltaje enviado por los otros intrumentos de registro, sobre un nivel de 5 volts e invertirlo, para que a partir de ahí baje hasta 2.5 volts y pueda ser detectado por la tarjeta, la cual automáticamente comienza a tomar muestras.

INTERFACE DE CONEXIONES
Es una pequeña caja que permite realizar conexiones y facilita su arreglo, pues una tarjeta de adquisición puede manejar hasta 16 canales de entrada y 8 de salida.

TARJETA DE CONVERSIÓN A/D
La tarjeta de adquisición que se utilizó para la captura de datos fue una Keithley-Metrabyte ***DAS40-G2***. La cual debe instalarse en la computadora y desde el sistema operativo darle la siguiente configuración a su manejador externo:

- Board Type (DAS40-G2). Tipo de tarjeta que se desea usar, en nuestro caso Das40-G2, también puede darse de alta la Das40-G1.
- Adress (240h). Puerto de entrada/salida
- IRQ Chanel (10). Canal de interrupciones
- DMA buf A/B (5/6). Uso de la memoria y canales para el modo DMA
- A/D.Mode (Single Ended). Referencia de voltaje, monopolar o diferencial
- A/D.Range (+/- 10 V). Rango de voltaje de entrada
- Digital I/O (8 in / 8 out). Puertos digitales de entrada/salida activos

Una vez instalada la tarjeta y configurado el manejador externo DAS, será identificada cada que se arranque la computadora con dar desde el teclado la instrucción *DAS40*, tomando la última configuración establecida. Es recomendable que esta instrucción se incluya dentro del programa autoejecutable de arranque (autoexe.bat). Esta configuración puede cambiarse cuando sea necesario escribiendo la instrucción *DAS40 -Menú* desde el sistema operativo. A continuación se presentará una lista con las últimas especificaciones, y nos presenta además las diferentes opciones que existen. Con la nueva configuración definida, se salva el programa y al salir nuevamente al sistema operativo se ejecutar la instrucción *DAS40* para que la computadora reconozca la nueva configuración.

Es recomendable que sólo se hagan cambios a la configuración de la tarjeta en contadas ocasiones, pues una vez configurada para el sistema por conveniencia se incluye en el autoejecutable y la identificación de su configuración es automática. Es preciso mencionar que antes de eso debió de correrse primero el programa *Memmaker* desde el sistema operativo MS-DOS.

COMPUTADORA

La computadora deberá tener como mínimo las siguientes características:

- Coprocesador 80X87 (SX) instalado.
- Monitor SVGA o VGA.
- 640 KBytes de memoria en RAM.
- ASYST instalado (como ya se especifico en la sección 2.1) y crear o tener una versión de ASYST que esté configurada de la siguiente manera:

A) Seleccionados los siguientes "overlays":
Data Files.- Este "overlay" permite almacenar arreglos numéricos en disco como archivos compatible en DOS. Los archivos de datos en ASYST han sido diseñados para permitir la lectura y escritura de sus datos.

123 File Interface.- Para transferencias de datos de ASYST a archivos Lotus o Excell (.WKS o .WK1) y viceversa. Los datos pueden ser tomados de un archivo o generados en el momento.

DOS File Sharing.- Permite realizar interfaces de MS-DOS con Asyst.
HP Plotter Driver.- Permite dirigir las gráficas de salida a impresoras HP.

Waveform Operations (I y II).- Contiene palabras que son usadas para el análisis y manipulación de las formas de onda. Funciones tales como integración, diferenciación, suavización y transformadas de Fourier (FFT).

Baby Driver Interface.- Soporte para un gran número de impresoras

Data Acq Master.- Para hacer uso de todos los comandos utilizados en las conversiones A/D y D/A y digital I/O.

Ext.Das Driver Support.- Soporta la especificación del manejador externo DAS hecha desde ASYST.

B) Configuración de la memoria:
Kbytes free for new arrays: 51
Kbytes free for new expanded memory tokens: 1696
Symbol Table size in kbytes: 32
String Segment size in kbytes: 20
Das buffer size in kbytes: 1
GPIB queue size in kbytes: 0
User Dictionary size in Kbytes: 48
Token Heap size in kbytes: 100
Unnamed Array size in kbytes: 100
Keyboard buffer size in bytes: 64992
Disallow expanded memory tokens? N
Recall default values (y/n): N

C) DAS Configuración debe encontrar la siguiente tarjeta:
(1) Das40 Board #N I/O Adr:20

D) Hardware Configuration, en activo:
Display (VGA Card and Monitor) (no. 11)
Printer (for 480 vertical pixel display mode) (no. 2)
Text Colors, File Sharing y **Display Mode** como el usuario lo desee.

E) En Graphics Mode que el modo sea el número **18**, con 640 en HorizResol (resolución horizontal), 480 en VertResol (resolución vertical), 16 colores Plot Style activado Solid.

- Contar con los programas Program1.Dmo y Program2.Dmo (Ver Apéndice) en caso que se desee crear por primera vez el sistema de corrido de este sistema. Es conveniente entrar a la versión de ASYST que se ha creado con las características mencionadas, y cargar con la instrucción *Load* los programas Program1.Dmo y en seguida Program2.Dmo, al término de lo cual nos saca al sistema operativo pero ha creado un sistema de corrido que arroja dos archivos: *Sistema.Com y Sistema.Ovl*. Para ejecutar el sistema de corrido basta con escribir desde el sistema operativo la palabra *SISTEMA* y seguir las instrucciones que ahí se especifican. Es preciso mencionar que este procedimiento se realiza una sola vez, ya que los archivos *Sistema.Com y Sistema.Ovl* quedarán de manera permanente en el Disco Duro y basta con emitir su nombre para entrar al sistema.

IMPRESORA
La impresora debe estar instalada, el sistema reconoce como dispositivo PRN a la HP LaserjetII o algún otro modelo más reciente como son la LaserjetIII o la Laserjet4.

2.2 Estructura del Sistema

Como ya especificamos un sistema debe cumplir con ciertas características fundamentales, ser comprensible, modificable y confiable. Para lograr lo anterior los programas que componen el sistema deben de aprovechar las herramientas ofrecidas por la programación estructurada. El sistema para adquisición y análisis de señales fisiológicas que hemos desarrollado, hace uso de dichas herramientas utilizando principalmente la modularidad y un diseño descendente.

El sistema define primeramente el ambiente en el cual se va a trabajar, y todas las áreas de trabajo que van a utilizarse. A continuación damos una lista siguiendo el orden en el cual fueron definidos en el programa *Program1.dmo* y explicamos brevemente cada una (si se desea ver el código ir al Apéndice).

- Se define la impresora de trabajo
- Se definen dos plantillas de adquisición: una para manejar solamente un canal de adquisición, que es identificado como el 0; y la otra para manejar dos canales de adquisición, el 0 y el 1.
- Se definen los arreglos de datos que van a utilizarse, tanto los de tamaño y descripción fijos como los variables (tokens).
- Se definen las variables. Variables enteras, reales y cadenas de caracteres (strings), se especifica su precisión, y tamaño.
- Se definen las ventanas que van a utilizarse. Tanto en modo normal, como las que se usan en modo gráfico.
- Definición de "vuports" de graficación. Se definen los tamaños y nombres de las diferentes áreas de graficación, lo cual debe hacerse en modo gráfico.
- Se inicializan las variables. Se inicializan las muestras, repeticiones, canales, y los colores de cada uno de los elementos de las gráficas (ejes, fondos etc.).
- Se especifican los colores que van a tener el fondo, los ejes, las etiquetas, la línea de graficación, y el cursor en cada uno de los "vuports" que se definieron anteriormente.

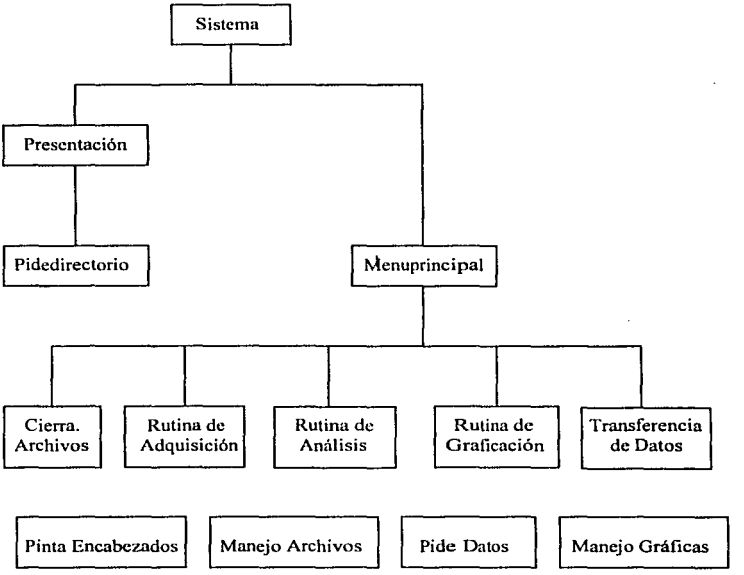
En seguida se procede a crear las definiciones de dos puntos que utilizará el sistema. Todas las definiciones de dos puntos que se crean a continuación de las definición de variables, arreglos , "vuports", etc., pueden hacer uso de ellas, modificando sus valores, estructura y formato cuando así se especifique.

El sistema se encuentra estructurado en 6 módulos que a su vez se subdividen en palabras o definiciones de dos puntos que se relacionan entre sí y son llamadas por los procedimientos más generales. La estructura es la siguiente:

1. **Definiciones Generales** . Aquí se encuentran todas las definiciones de dos puntos utilizadas por las rutinas de adquisición, análisis, graficación y transferencia de datos.
2. **Rutina de Adquisición**. Es la parte básica del sistema y se encarga de capturar los datos producto de señales electrofisiológicas.
3. **Rutina de Análisis de Resultados**. Permite realizar diversos análisis de la información capturada por la rutina anterior, tal como mediciones, captura de áreas etc.,.
4. **Rutina de Graficación**. Realiza la graficación de los datos capturados, permitiendo darles una presentación determinada por el usuario, y su impresión.
5. **Transferencia de Datos**. Transfiere archivos creados por el sistema a archivos que pueden usarse por Lotus-123 o Excell.
6. **Definiciones que Enlazan las Anteriores**. En esta parte se define una presentación del sistema, el menú principal manda llamar a cada una de las rutinas anteriores, contiene una definición de dos puntos que arranca todo el sistema, una definición Banner, y las instrucciones que se encargan de crear el sistema de corrido.

ÁRBOL DE MÓDULOS

En el siguiente diagrama se explica de manera general cómo se encuentra estructurado el sistema, a continuación se explicará de manera detallada cada uno de los módulos que aquí se especifican.



SISTEMA

Este módulo está compuesto por cinco definiciones de dos puntos, que se encargan de enlazar todas las rutinas y hace una pequeña presentación del sistema. Las definiciones de dos puntos hechas en este módulo son las siguientes:

Cierra.Archivos .- Esta definición de dos puntos se encarga de revisar que al arrancar el sistema no se encuentren archivos abiertos, en cuyo caso los cierra para poder empezar la ejecución del sistema.

Pidedirectorio .- Pide al usuario que indique el directorio en el desea archivar la información, incluyendo la unidad de disco. Hace el cambio de directorio, y es donde se realizarán todas las operaciones con los archivos que maneje ASYST.

Presentación .- Despliega una pequeña presentación del sistema, especificando la hora y la fecha de ese momento, la institución en la que se encuentra, el nombre del sistema, y la persona responsable de dicha institución. Después de realizar lo anterior, se manda llamar a la definición de dos puntos *pidedirectorio*.

Menuprincipal .- Ofrece al usuario las diferentes opciones que tiene para utilizar el sistema. Antes de seleccionar alguna de las opciones,manda llamar a *Cierra.Archivos*. Las opciones que ofrece son: la rutina de adquisición de datos, la rutina de análisis de resultados, la rutina de graficación, la de transferencia de datos o salir del sistema. Una vez seleccionada una opción, se realiza una presentación explicativa de cada una de ellas, y se enlaza con la palabra correspondiente para que se ejecute dicha rutina, al término de la cual nos regresa a este menú principal. Si la opción escogida es la de salir, se despliega un mensaje de despedida, y nos regresa al directorio de trabajo en el cual mandamos a llamar al sistema

Sistema .-Arranca todo el sistema. Se encarga de guardar el directorio desde el cual fue llamado el sistema, para regresar a él cuando se ha acabado de ejecutar. Manda a ejecutar primero la *Presentación* y a continuación *Cierra.Archivos* y *Menuprincipal* para enlazarse con las rutinas.

MÓDULOS INDEPENDIENTES

Las definiciones de dos puntos especificadas en estos módulos independientes, pueden ser utilizadas por cualesquiera de las otras definiciones de dos puntos que se encuentran en otros módulos que aparecen en el árbol descrito anteriormente. Los módulos independientes son: *Pide Datos, Pinta Encabezados, Manejo Archivos*, y *Manejo Gráficas*.

Pide Datos

En este módulo independiente se encuentran las definiciones de dos puntos con las cuales el usuario puede dar información directamente del teclado. Las definiciones de dos puntos definidas en este módulo son las que a continuación se describen.

Espera.Una.Clave .- Esta pequeña rutina espera que se oprima una tecla para continuar con la ejecución del programa. Antes de mandar llamar esta definición de dos puntos se puede desplegar un mensaje avisando al usuario que oprima una tecla, o se puede suprimir dicho mensaje, lo cual no es conveniente.

#Entrada&Checa .- Su función principal es la de esperar algún valor numérico que se de desde el teclado. En caso de que la entrada contenga caracteres no numéricos como son letras u otros, se repetirá la acción hasta que se de un valor numérico. Para regresar el cursor a las coordenadas en donde se encontraba al pedir el valor, se salvan tanto la línea como la columna correspondiente. El valor válido dado se almacena en la pila.

"Entrada&Checa .- Es similar a la anterior con la diferencia que esta espera recibir del teclado una cadena de caracteres. En caso de que la longitud de la cadena sea cero, es decir que no se haya dado nada desde el teclado, repite la operación hasta que se de alguna cadena de caracteres. La cadena de caracteres válida se almacena en la pila de símbolos para usos posteriores.

Dar.S/N .- Esta definición de dos puntos espera que del teclado se de una sola letra, ya sea una "S" que representa un si, o una "N" que representa un no, también pueden darse en minúsculas "s" o "n" y esta rutina las reconoce. Esta rutina se encarga de validar que el caracter dado sea en realidad una "s" o una "n" en minúscula o mayúscula, de lo contrario desplegará un mensaje de error y regresará a pedir nuevamente una respuesta. Esta definición e dos puntos se utiliza principalmente cuando durante el programa en curso se desea realizar una pregunta cuya respuesta sea si o no.

Pinta Encabezados

Este módulo independiente contiene las definiciones de dos puntos que se encargan de pintar las etiquetas a las diferentes gráficas que va generando el sistema, menos a las de la rutina de graficación en las que debido a que se mandan a impresión el usuario puede ponerle las etiquetas y colores que desee. Estas definiciones de dos puntos son usadas principalmente por las rutinas de análisis y adquisición, en donde las etiquetas siempre serán las mismas. A continuación se enlistan y se describen las diferentes definiciones de dos puntos realizadas en este módulo independiente.

Pintaetiquetas1 .- Pinta las etiquetas correspondientes a los ejes en las gráficas desplegadas. Estas etiquetas son las palabras "volts" y "tiempo (msec)", y se usan cuando se está desplegando sobre el "vuport" *arriba* que es el que ocupa la parte superior de la pantalla o *abajo* que ocupa la parte inferior de la pantalla. Esta definición contiene un interruptor (sw) que se encarga de verificar si la llamada a esta definición de dos puntos se hace desde la rutina de análisis, en cuyo caso tendrá el valor 1, o de la rutina de graficación en cuyo caso tendrá el valor 0. Esto permite mantener constante el color de las etiquetas (verde) cuando se llama desde las rutinas de adquisición o de análisis, cuando el usuario se encuentra dentro de la rutina de graficación, puede modificar los colores.

Pintaetiquetas2 .- Funciona como la anterior, pero solamente se utiliza en la rutina de adquisición, cuando se realizan adquisiciones en dos canales. Estas etiquetas aparecen en los "vuports" *arribaizq y abajoizq*, donde se despliega cada uno de los canales de adquisición respectivos, y cuyas coordenadas de despliegue difieren del caso anterior.

Pintaencabezado1 .- Despliega el encabezado correspondiente de señales acumuladas, como la definición anterior, difiere de su posición dependiendo del número de canales con los que se esté muestreando. El encabezado desplegado es "*Acumulados*".

Pintaencabezado2 .- Su posición no depende del número de canales que se están muestreando. El encabezado desplegado es el siguiente "*Promedio*".

Pintaencabezado3 .-Despliega los nombres de los archivos requeridos para ser graficados. Se utiliza para desplegar el nombre de los archivos que son recuperados y graficados para su análisis. Al igual que pintaetiquetas1 utiliza el interruptor (sw), por los mismos motivos.

Pintarep .- Despliega el número de adquisición que se está realizando en el momento, se incrementa en uno y no sobrepasa el número de repeticiones que se establece al comenzar la adquisición de datos.

Manejo Archivos

Las definiciones de dos puntos que aquí se describen son las que se encargan de dar una lista de archivos creados, así como de seleccionar un archivo de entrada que previamente fue grabado en la rutina de adquisición. Existen otras definiciones de dos puntos que se encargan de grabar la información en archivos de disco, pero que son propias de la rutina de adquisición y de esas hablaremos más adelante.

Muestraarchivos .- En esta definición de dos puntos, se muestran todos los archivos tipo promedio (.prm) y los acumulados (.acu) que se encuentran en el directorio de trabajo. Para ello se realiza una interface con el sistema operativo MS-DOS.

Muestraarchivoswks .-Similar a la anterior, solamente muestra los archivos tipo (.wks) creados con anterioridad en el directorio de trabajo en el cual se encuentra operando el sistema. Utiliza la interface con MS-DOS.

Selecciona.Archivo.Entrada .- Permite seleccionar un archivo de datos que fue creado anteriormente para accesarlo. Esta operación valida el modo actual (gráfico o normal), muestra los archivos que se encuentran en ese directorio haciendo una llamada a *muestraarchivos*. A continuación pide el nombre del archivo que se desea accesar, valida que se encuentre bien escrito, y que el archivo exista. En caso de que no exista, da la opción de intentarlo de nuevo.

Manejo Gráficas

A continuación se describirán cada una de las definiciones de dos puntos que manejan mediciones de las gráficas, tanto en el momento de realizar la adquisición, como después de haber hecho ésta. En este módulo independiente se puede hacer uso de las palabras que

hacen posible el uso de gráficas interactivas, que arrojarán los principales resultados del análisis de datos.

Menumedición .- Esta rutina despliega el menú para realizar mediciones en la gráfica en uso. Para ello es necesario que ASYST se encuentre en gráficas interactivas, manejando el modo de *graphics.readout*, esto permite manejar las teclas inicio, supr, regreso de página y las flechas, ver las coordenadas en las que se encuentra el cursor, terminar la gráfica interactiva, cambiar de incremento el desplazamiento del cursor, y moverlo a través de toda la gráfica. Las coordenadas en las que se encuentra el cursor son desplegadas en la esquina superior derecha del "vuport" que en uso.

Cursores .- Permite hacer uso de las gráficas interactivas, pero mediante el modo *array.readout*, para manejar dos cursores verticales dentro de la gráfica. Puede emplear todas las funciones especificadas para *graphics.readout*, y además de las teclas *insert, fin,* y *avance de página*, con las que se puede realizar un acercamiento (zoom) de la zona delimitada por los cursores; permite mover sólo el cursor izquierdo o el derecho, según la tecla seleccionada.

Coordcurs .- Permite recobrar las coordenadas en las se encuentra el cursor, los valores se colocan en el tope de la pila, de donde se recuperan más adelante asignándoselos a las variables correspondientes. También valida que los cursores no arrojen coordenada impropias, por ejemplo cuando el cursor izquierdo sobrepase al derecho y entonces pase a a ser el derecho o viceversa.

Piderespimp .- Permite recortar la gráfica de resultados arrojada por una operación entre dos archivos. Después de realizar el recorte a la gráfica correspondiente, la vuelve a graficar ya recortada; nos permite poner títulos o leyendas a esta gráfica y enviarla a impresión. A continuación nos pide el número de copias que se necesitan, y pregunta si se desean imprimir las gráficas tal como se encuentran en la pantalla, es decir, una de ellas en la parte superior izquierda, la otra en la superior derecha, y en la parte inferior el resultado de la operación realizada. En caso de que no se deseen imprimir con este formato, da la opción de imprimir sólo la gráfica de resultados o ninguna de ellas. Si se decide imprimir toda la pantalla, se imprime tanto las gráficas fuentes como el resultado de la operación realizada entre ellas.

Menuofertas.- Ofrece el menú para realizar operaciones entre dos archivos, suma, resta, división, superposición o filtrado. Una vez seleccionada, la operación se realiza con los dos archivos, y si esta definición de dos puntos fue activada desde la rutina de graficación, se llama a *piderespimp* (ver más arriba), en caso contrario se despliegan las etiquetas y encabezados, y manda llamar a *menumedicion* y *cursores* para poder efectuar mediciones en las gráficas de resultados. Al terminar las mediciones se pregunta si se desea realizar otra operación con los archivos ya seleccionados, o si se sale del menú de ofertas.

RUTINA DE ADQUISICIÓN

La rutina de adquisición captura los datos por medio de la tarjeta DAS40. Dicha rutina cumple con las siguientes funciones:

- Tiene una presentación explicativa de la rutina
- Contiene un menú que permite seleccionar:
  a) El número de canales de muestreo (1 o 2)
  b) El número de muestras de una adquisición por cada registro
  c) El número de repeticiones o número de veces que va tomarse un registro
  d) El tiempo de retardo entre cada muestra, este dato debe de darse en milisegundos.
- Inicia la captura mostrando el despliegue de cada registro que ingresa con escalas reales de tiempo y voltaje
- Al terminar la captura de los registros, muestra el promedio de ellos
- Ofrece en el despliegue de promedios un cursor móvil que permite medir tiempos y amplitudes. Las coordenadas del cursor son desplegadas en la pantalla.
- Graba el registro promedio si el usuario lo desea. El archivo generado tiene por extensión .PRM.
- Graba los registros acumulados que dieron origen al de promedios si el usuario lo desea. El archivo generado tiene por extensión .ACU.
- Permite cambiar los parámetros dados en el punto dos si el usuario lo cree conveniente.
- Realiza las adquisiciones hasta que el usuario lo determine.

A continuación mencionaremos y describiremos brevemente las definiciones de dos puntos que se encargan de hacer posible las funciones antes mencionadas.

Presentaadquisicion .- Presenta esta rutina, especifica cada uno de los parámetros que utiliza, y el formato en los cuales son especificados a la máquina.

Iniciaplantilla1ch .- Pone en activo a la plantilla que realiza adquisiciones en un sólo canal (canal #0), asigna tamaño y tipo al arreglo en donde se van a almacenar los datos resultado de la adquisición. Especifica el modo cíclico, e inicializa la plantilla.

Iniciaplantilla2ch .- Pone en activo a la plantilla que realiza adquisiciones en dos canales (canal #0 y #1), asigna tamaño y tipo al arreglo en donde se van a almacenar los datos resultado de la adquisición en dos canales. Especifica el modo cíclico, e inicializa la plantilla.

Crea.Archivo .- Especifica el formato y crea un archivo cuyo nombre se indica con anterioridad, lee el número de comentarios que se van a realizar. La creación del archivo es en forma física, ya que a continuación se debe de abrir para poder grabar información en él.

Graba.Comentarios .- Abre el archivo indicado previamente y pide del teclado los comentarios cuyo número fue especificado también, los grabar en el archivo correspondiente.

Selecciona.Archivo.Salida .- Pide del teclado el nombre del archivo en el cual se grabarán los datos adquiridos, elimina los caracteres no válidos, valida que el archivo no exista y pide el número de comentarios que se desean realizar. Esto lo hace tanto para el canal #0 como para el canal #1, en caso de que se deseen grabar los promedios de cada canal, también puede seleccionar únicamente uno de los canales. Esta definición de dos puntos hace uso tanto de *Crea.Archivo* como de *Graba.Comentarios* las cuales son llamadas después de hacer las validaciones que se especificaron con anterioridad.

Grabaarrchiprom .-Graba los arreglos promedio de cualquiera de los canales que se utilizaron en el archivo previamente seleccionado y creado por *Selecciona.Archivo.Salida*. Toma dos arreglos que se encuentran en el tope de la pila y los graba en el archivo. El primer arreglo que debe tomar es el que contiene los tiempos de la adquisición, el segundo contendrá el promedio de las adquisiciones. Una vez hecho lo anterior cierra el archivo en uso, y en caso de que no se haya confirmado la grabación se aborta dicho proceso.

Grabaarchacum .- Graba los arreglos acumulados del canal #0 en el archivo seleccionado previamente por *Selecciona.Archivo.Salida*. Los arreglos fueron puestos en la pila durante la adquisición y de ahí son tomados para grabarse en el archivo de salida, el primer arreglo grabado es el que contiene los tiempos reales de la adquisición.

Grabaarchacumc0 .- Graba el arreglo que se encuentra en el tope de la pila y que debe de corresponder a una adquisición hecha en el canal #0, ya que abre el archivo correspondiente al canal #0 y a continuación lo cierra.

Grabaarchacumc1 .- Graba los arreglos acumulados por la adquisición hecha en el canal #1.

Capturaarreglo .- Realizar un ciclo mediante el cual se capturan los datos de un solo canal de muestreo (#0), almacenando los datos en un arreglo y con la velocidad de muestreo especificada al ser llamada esta definición de dos puntos. Se mide el tiempo real que tarda el ciclo, el cual deberá corresponder al tiempo de duración del registro.

Sumarygraficar .- Suma los arreglos que se van adquiriendo en un solo canal (canal #0), y almacena el resultado en otro arreglo. También grafica cada uno de los arreglos que se generan por cada adquisición, para lo cual, al terminar de almacenar los datos en el primer arreglo se convierten a voltios, y se calcula el arreglo que contendrá los tiempos correspondientes a los datos arrojados por la adquisición, la fórmula que nos da ese arreglo de tiempos es:

Tiempo.Data = Arreglo[1, 2, 3, 4, 5, ...,muestras] * (tiempofinal / muestras)

Donde *tiempofinal* es el tiempo que duró el evento, *muestras* es el número de datos que se tomaron durante el muestreo, y *Arreglo* es un arreglo unidimensional cuyos elementos son los números reales *1,2,3,...,muestras*. Con esta formula se obtiene el arreglo

correspondiente a los tiempos en que se obtuvieron cada una de las muestras, ya que *tiempofinal/muestras* nos da el tiempo que hay entre una muestra y la siguiente, al multiplicar este valor por *Arreglo* cuyos elementos son *1,2,3,...,muestras* nos da como resultado el tiempo en que fue adquirida cada muestra del arreglo, dependiendo del número de elemento que le corresponde. Una vez con el arreglo de tiempos, se procede a graficar contra el arreglo que arrojó la adquisición y se pintan las etiquetas correspondientes. Los demás arreglos que se obtenienen de las adquisiciones, también se convierten a voltios y se grafican sobrepuestos a los anteriores.

Capturaarreglos .- Semejante a la definición antrerior, pero funciona para dos canales de muestreo(canal #1 y #2). Es conveniente decir que al muestrear en dos canales, el resultado obtenido son dos valores, uno por cada canal y se colocan en el tope de la pila. Los datos obtenidos durante los muestreos, se guardan en un arreglo de datos.

Sumarygraficardos .- Ejecuta las mismas funciones que *sumarygraficar*, con la diferencia que lo hace para dos canales de muestreo. La fórmula que nos da el arreglo de tiempos se modifica por la siguiente:

Tiempo.Data = Arreglo[1,2,3,...,muestras/2] * (tiempofinal / (muestras / 2))

Con esta fórmula se obtiene el arreglo que contiene los tiempos de muestreo de cada canal. A continuación se obtienen los subarreglos de los arreglos originales donde están almacenados los datos de la adquisición. Así el subarreglo que contenga todos los elementos impares del arreglo original, será el correspondiente al canal #0 y el que contenga los pares corresponderá al canal #1.

Pidecanalmuestrayrep .- Esta definición de dos puntos se encarga de pedirle al usuario el número de canales de muestreo, el número de muestras por cada adquisición, el número de repeticiones de la adquisición, y el tiempo de retardo entre cada muestra. Se encarga de validar que cada uno de los valores dados se encuentre dentro de los rangos establecidos. Cuando se selecciona el número de canales estos solamente pueden ser uno o dos. En lo que respecta al número de muestras es un valor que puede escoger el usuario y que se recomienda que se encuentre entre 300 y 1500 para que se tenga una buena digitalización. El número de repeticiones es el número de veces que se van a adquirir datos por cada canal. De este número depende el arreglo promedio que se graficará a continuación de las adquisiciones.

Rutinadeadquisición .- Es la rutina principal de este módulo, relaciona las definiciones de dos puntos anteriores para lograr con ellas las funciones propuestas. Inicialmente pregunta el número de canales, de muestras, de repeticiones y el retardo entre muestras. Una vez con estos parámetros, espera a que el usuario revise que todos los aparatos y conexiones estén listos para iniciar la adquisición. Conocido el número de canales sobre los cuales se va a muestrear, manda llamar las definiciones de dos puntos correspondientes y definidas anteriormente encargadas de dichas operaciones, grafica además los datos que se van adquiriendo. Al concluir, calcula el promedio de todos los

arreglos que se adquirieron y lo gráfica. Para la graficación de un solo canal la pantalla se divide en 2 partes, en una de las cuales se grafican los datos que se van adquiriendo y en la otra el promedio de los mismos. A continuación el usuario puede tener acceso a la gráfica de promedios para realizar algunas mediciones.

Una vez terminado este procedimiento, se procede a grabar en un archivo el arreglo que contiene los datos promedio de las adquisiciones, si el usuario así lo determina. Se realiza el mismo procedimiento para los archivos acumulados, los cuales fueron almacenados en la pila conforme se fueron adquiriendo.

Cuando se trabaja con dos canales, los procedimiento de captura y graficación son los mismos, con la diferencia de que entonces se divide la pantalla en 4 regiones, dos para la adquisición en línea de cada canal, y las dos para los promedios respectivos.

A continuación se graban los archivos promedio por cada canal, si el usuario así lo solicita. Cuando se desean grabar los arreglos acumulados esta definición de dos puntos procede de la siguiente manera:

Los datos almacenados se encuentran apilados como sigue.

Pila
ArregloN
.
.
.
Arreglo2
Arreglo1

Hay que considerar que los arreglos contienen tanto la información del canal #0 como del canal #1. Si se desea grabar la información de los dos canales, el arreglo que se encuentra en el tope de la pila se duplica y se obtiene un subarreglo de él para el canal #0. Se llama la definición de dos puntos *grabaarchacumc0;* en seguida se obtiene un subarreglo para el canal #1 y se manda llamar *grabaarchacumc1*. Este procedimiento se repite hasta que se terminan los arreglos de datos de la pila. Si se desea grabar la información de un solo canal el procedimiento es más sencillo, ya que los arreglos no se tendrán que duplicar y solo se obtiene el subarreglo de los elementos impares o pares según sea el caso.

RUTINA DE ANÁLISIS

Esta rutina utiliza los archivos de datos que han sido creados previamente a través de la rutina de adquisición. Los análisis que realiza pueden aplicarse tanto a archivos promedio como a acumulados, salvo en el caso en que se desea realizar una operación con dos archivos. La presente rutina cumple con las siguientes funciones:

- Lee los archivos tipo promedio y acumulados y los despliega gráficamente
- Permite realizar mediciones de áreas comprendidas entre dos cursores móviles, que pueden ser manejados por el usuario
- Permite medir tiempos y amplitudes en la gráfica activa, empleando para ello cursores móviles. Los tiempos y las amplitudes se desplegarán en la pantalla.
- Despliega gráficamente dos archivos tipo promedio y presenta un menú que permite:
  a) Sumarlos
  b) Restarlos
  c) Dividirlos
  d) Superponerlos
  e) Filtrarlos
- Realiza acercamientos al área definida entre los cursores móviles (zoom).
- Cualquiera de las funciones anteriores pueden repetirse el número de veces que se desee.

A continuación se describirán cada una de las definiciones de dos puntos que en conjunto cumplen con las funciones antes mencionadas.

Pidearchivoygrafica .- Esta definición de dos puntos pide al usuario que seleccione el archivo que va a graficarse, ya sea tipo promedio o tipo acumulado, grafica los datos con sus respectivos comentarios.

Mideareas .- Mide el área que se encuentra entre dos cursores móviles y por debajo de la gráfica. Para ello se necesita saber el número de elemento que le corresponde al cursor derecho e izquierdo (coordenada inicial y final) en el arreglo que contiene los tiempos, para lo cual se despeja de la fórmula dada anteriormente:

coordin = (coordin * muestras) / tiempofinal
coordfin = (coordfin * muestras) / tiempofinal

donde *coordin* es la coordenada que le corresponde al cursor izquierdo, y *coordfin* el que le corresponde al cursor derecho, pero los valores que se regresan van de acuerdo al que les corresponden en el arreglo de los tiempos, no así el número de elemento. Todo esto se debe a que para calcular el área se utiliza la instrucción *integrate.data*, la cual toma un arreglo del tope de la pila y calcula su integral, dejando en el tope de la pila un arreglo que contiene el valor de la integral valuada en el número de elemento que le corresponde. Por esto la coordenada inicial (*coordin*) y la coordenada final (*coordfin*) son necesarias, por medio de ellas conocemos el valor de la integral en el cursor izquierdo y en el cursor derecho, sólo basta restar esos valores para obtener el valor del área comprendida entre los dos cursores. Este procedimiento es similar a la solución de una integral definida que tiene tanto un límite inferior como un límite superior. Los resultados de dicha integración son desplegados en la pantalla

Midetiempoampl .- Realizar las mediciones correspondientes a los tiempos y amplitudes en la gráfica. Para esto se colocan los cursores en los lugares deseados, se recuperan las coordenadas en las cuales se encuentran cada uno, y se obtiene la diferencia entre ellos, obteniendo así la diferencia en las coordenadas X (tiempo) y la diferencia en las coordenadas Y (Volts).

Rutinadeanalisis .- Esta definición de dos puntos, reúne a las anteriores para realizar los análisis que se desean. Primeramente presenta un menú que permite seleccionar el tipo de análisis que se desea realizar (medir áreas entre dos cursores, tiempos y amplitudes, operaciones con dos archivos o regresar al menú principal). En caso de que se escoja entre las dos primeras, se llama al archivo y se graficar; nos permite además manejar los cursores en la gráfica y recuperar las coordenadas en las que se encuentran para realizar mediciones de áreas o tiempos y amplitudes según sea el caso. Si se escoje la opción de operación con dos archivos, pide dos archivos para trabajar con ellos, los grafica y manda llamar a la definición de dos puntos *Menuofertas* que ya fue descrita y que nos permite realizar operaciones con dos archivos (sumarlos, restarlos, dividirlos, superponerlos, y filtrarlos).

RUTINA DE GRAFICACIÓN

La rutina de graficación tiene por objetivo cumplir con las siguientes funciones:

- Seleccionar el archivo tipo promedio o tipo acumulados que se desea graficar para mandarlo a impresión.
- Permite hacer arreglos de presentación tales como escribir leyendas y títulos en cualquier lugar de la gráfica.
- Permite seleccionar los colores que han de utilizarse para los diferentes elementos de la gráfica.
- Permite hacer ajustes en las dimensiones de la gráfica antes de su impresión.
- Permite realizar operaciones con dos archivos previamente seleccionados y mandar a impresión, ya sea el resultado de la operación, o las dos gráficas y el resultado conjuntamente.
- Permite regresar al menú principal cuando el usuario así lo determine.

Esta rutina sólo contiene dos definiciones de dos puntos, ya que una de ellas realiza con la ayuda de las definiciones de dos puntos definidas en los módulos independientes, las funciones que se mencionaron anteriormente. A continuación se describen brevemente estas definiciones de dos puntos.

Opcioninc .- Se encarga de mandar un mensaje de opción incorrecta que se torna muy repetitivo cuando se están seleccionando los colores que han de usarse para las gráficas que se envian a impresión.

Rutinadegraficacion .-Realiza las funciones asignadas a esta rutina, por lo que presenta un menú de selección que permite:

1) Seleccionar el archivo a graficar
2) Ajustar los colores y las dimensiones de la impresión
3) Poner títulos, leyendas y mandar a impresión un archivo
4) Realizar operaciones con dos archivos y mandarlos a impresión
5) Regresar al menú principal

En caso de haber seleccionado la opción número uno, se selecciona un archivo de entrada y lo gráfica.

Si la opción seleccionada es la número dos, se despliega una lista de los colores disponibles (16) y su número asignado. Se puede seleccionar el color del fondo de la gráfica, el color de los ejes, el de las etiquetas, el color de los títulos que se desean escribir, el color que tendrá la línea de la gráfica, y el de los comentarios que se realizaron en el momento de la grabación del archivo. Cada opción se valida dentro de los rangos permisibles en el número de colores. A continuación permite seleccionar el número de divisiones que tendrá el eje horizontal, X, cuyo rango es de 1 a 10, lo mismo que para las divisiones del eje vertical, Y. También permite decidir si se desea imprimir la gradilla o rejilla de separación.

Si la opción seleccionada es la número tres, se ajustan los colores seleccionados anteriormente, y se grafica el archivo ya establecido. Permite recortar la gráfica hasta el punto que determine el usuario, y realiza un acercamiento (zoom) que también puede graficarse. A continuación puede colocarse el cursor en el lugar deseado para los títulos o leyendas y escribirlo. Pueden enviarse a impresión también los comentarios, indicar el número de copias requerido y el formato en que se desea imprimir (en media página o un cuarto de página). Una vez seleccionado todo lo anterior, se procede a la impresión.

Si la opción seleccionada fue la número cuatro, el procedimiento es el mismo, pero ahora con dos archivos, validando primero que el tamaño de los arreglos de cada archivo sea el mismo, de lo contrario la operación no podrá realizarse. Después se llama a la definición de dos puntos *Menuofertas* ya descrita, la cual realiza las operaciones con los dos archivos, da acceso a las áreas de graficación para poner títulos o leyendas, escoger los formatos de impresión, y seleccionar si se desea mandar a impresión sólo la gráfica de resultados o también las gráficas fuente.

TRANSFERENCIA DE DATOS

Esta rutina tiene sólo una función, transferir datos de archivos tipo promedio o tipo acumulados a archivos tipo *.wks*, para que puedan ser manejados tanto por Lotus-123 como por Excell. Consta de una definición de dos puntos que a continuación se describe:

Transferencia .- Selecciona un archivo tipo promedio o tipo acumulados con sus respectivas validaciones de existencia, nombra el archivo de salida que tendrá la terminación *.wks* donde se realiza la transferencia. La transferencia se realiza a una hoja de cálculo, empezando en la línea 3 columna 1 y los datos se graban por columnas, es decir hacia abajo. En caso de que se tenga más de un arreglo en el archivo fuente, se aumentará en uno la columna a donde se realiza la transferencia.

CAPÍTULO IV

PRESENTACIÓN DE RESULTADOS Y EVALUACIÓN

1. PRESENTACIÓN DE RESULTADOS

Una vez que se han grabado los registros obtenidos de señales electrofisiológicas, deben analizarse y preparar la presentación de resultados. Se puede considerar a esta etapa como el estado final del proceso de información que comienza con la digitalización de la señal analógica y el análisis de las diversas características de la misma.

La forma en que se van a analizar y presentar los datos depende de la información que puede ser extraída de los experimentos. Los procedimientos varían considerablemente, pero los podemos clasificar de la siguiente manera:

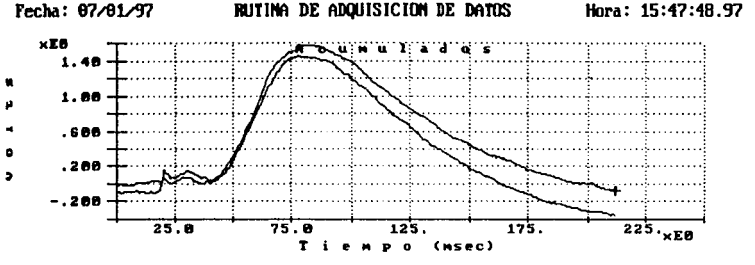
- Análisis y presentación de los efectos de un tratamiento específico en un preparado biológico
- Análisis y presentación de la evolución en el tiempo de la señal estudiada
- Presentación y análisis de la distribución de los diversos parámetros morfológicos de la señal analógica

Algunos de estos procedimientos pueden ser aplicados a series experimentales. El sistema permite en su parte de análisis de resultados realizar diversas mediciones de los diferentes atributos de la señal (tiempos, amplitudes, áreas), los cuales pueden ser grabados en los comentarios del archivo y posteriormente ser recuperados para su análisis estadístico.

1.1 Pantallas Generadas por el Sistema

En las páginas subsecuentes se presentan las pantallas principales que genera el sistema en su rutina de adquisición de datos y análisis de resultados. En cada una de ellas se explicará brevemente los elementos que la conforman, y las diferentes opciones que tiene el usuario para su manejo.

Las *figuras 1 y 2* en sus incisos a) y b), muestran diferentes momentos en la adquisición de datos, tanto para un canal como para dos. La *figura 3* en sus incisos a) , b) y c), muestra diversos aspectos de la medición de áreas. La *figura 4* en sus incisos a) y b), muestra como se realiza una medición de tiempos y amplitudes, y la *figura 5*, muestra como se realiza una operación con dos archivos. Las figuras 3, 4 y 5, se encuentran dentro de la rutina de análisis de resultados.



Espera un momento se esta realizando la adquisicion # 2 _

Fig. 1. a) La pantalla muestra como se esta realizando la adquisición de datos en el instante en que se han acabado de capturar las 2 primeras señales. Se encuentra en la rutina de adquisición de datos, mostrando hasta el momento sólo la ventana en donde se grafican los acumulados.

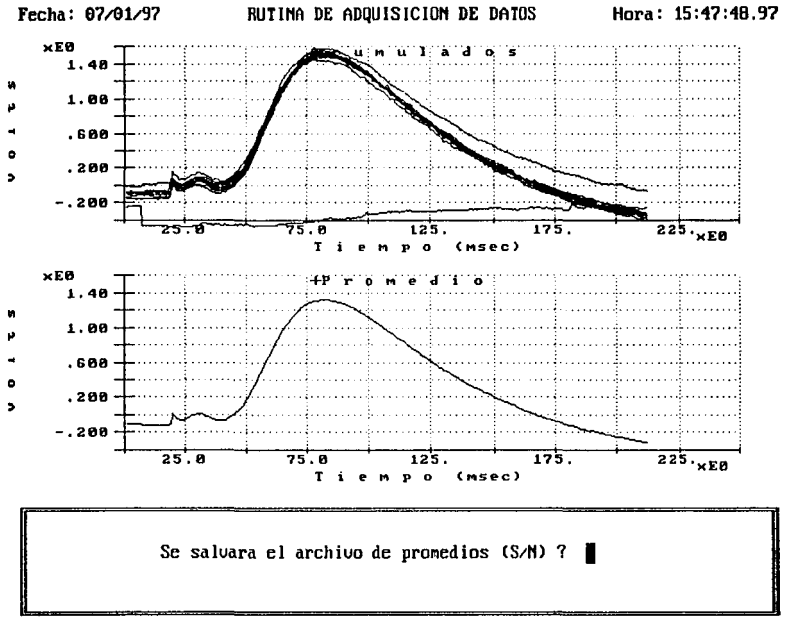


Fig. 1. b) En esta pantalla se muestra el momento en que se han acabado de adquirir datos, y muestra la gráfica de la señal promedio, que se ha obtenido de las 10 adquisiciones realizadas. Se puede ver que en las gráficas de los acumulados aparece ruido eléctrico que se elimina en la de promedios.

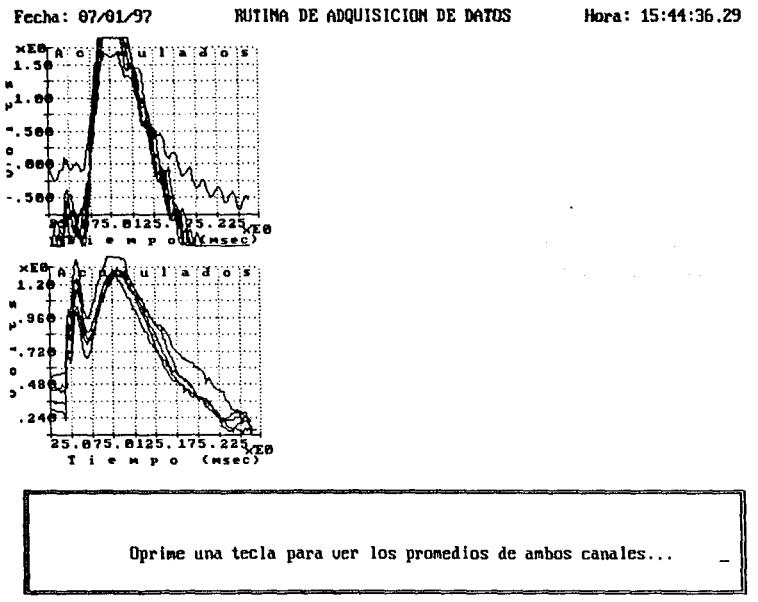


Fig. 2. a) La pantalla muestra como se realizan adquisiciones en dos canales, en el momento mismo que ha terminado de realizarlas. A diferencia de la que se realiza en un canal, esta se divide en cuatro ventanas, para poder mostrar las adquisiciones y los promedios.

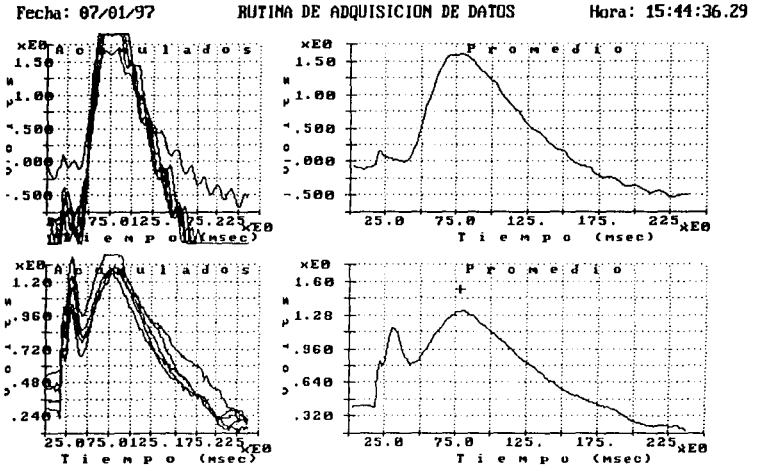


Fig. 2. b) En esta pantalla se muestran tanto las gráficas de los acumulados, como de los promedios obtenidos por cada canal. Además se puede observar que se permite el acceso a las gráficas, para poder realizar algunas mediciones que desee el usuario.

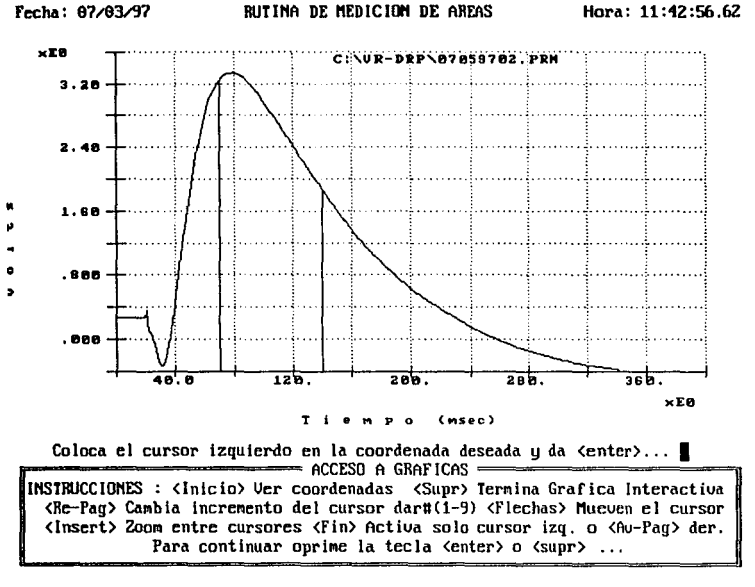


Fig. 3. a) En la presente pantalla se muestra cómo se ha recuperado un archivo grabado previamente y cuyo nombre se encuentra en la parte superior de la gráfica. Además permite el manejo de dos cursores (izquierdo y derecho), que pueden deslazarse siguiendo las instrucciones que se encuentran en el recuadro, para poder medir el área definida por ellos.

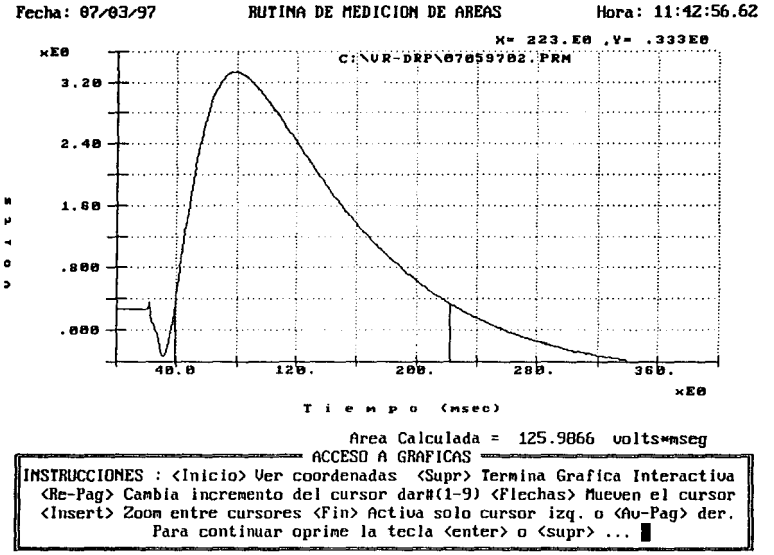


Fig. 3. b) En esta pantalla se muestra el momento en que el usuario ha elegido a través de los cursores el área que se desea calcular, y el resultado que esta operación a arrojado. El resultado se muestra entre la gráfica y el recuadro de instrucciones. También se muestran las coordenadas del último cursor que se uso, que en este caso es el derecho, y se encuentran en la parte superior derecha de la gráfica.

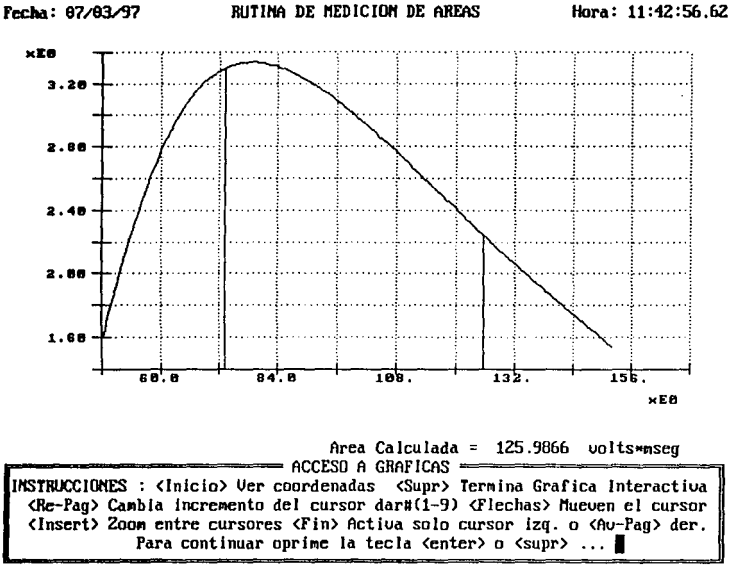


Fig. 3. c) Es esta pantalla se muestra cómo se ha elegido una región de la gráfica a la cual se desea hacer un acercamiento (zoom), y se vuelve a graficar tan sólo la región elegida, que aquí se muestra. También pueden utilizarse las teclas para accesar a la gráfica, e inclusive volver a realizar un acercamiento.

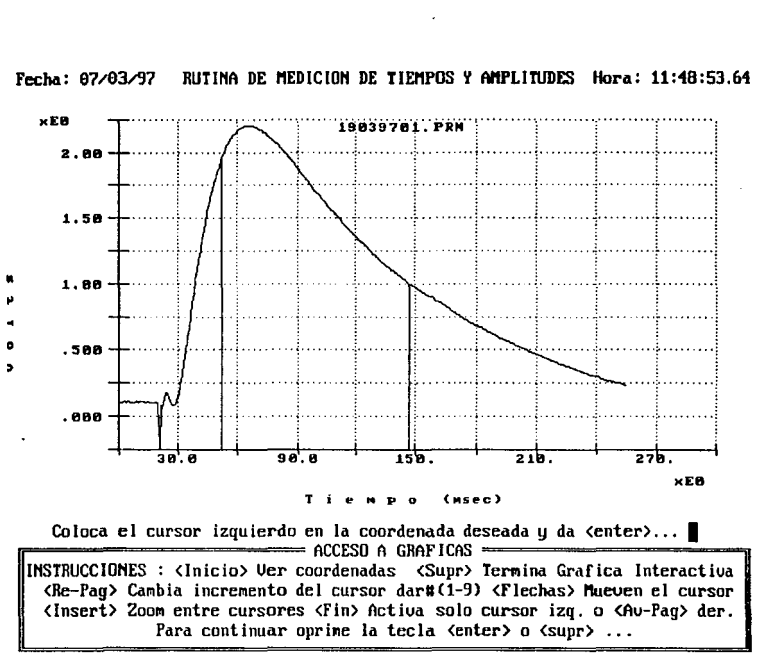


Fig. 4. a) Esta pantalla pertenece a la rutina de medición de tiempos y amplitudes, la cual permite colocar el cursor izquierdo y derecho en los lugares que desea el usuario, para realizar alguna medición.

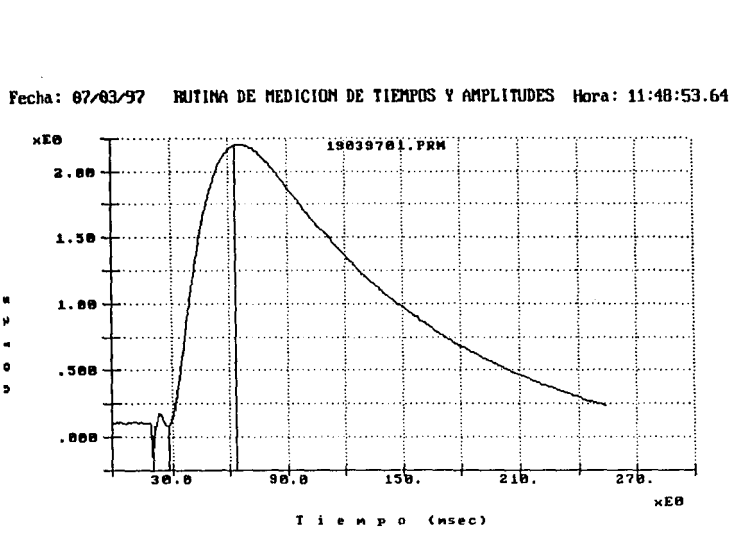


Fig. 4. b) En esta pantalla, el usuario ya ha colocado los cursores en los lugares deseados para realizar las mediciones, las cuales se despliegan entre la gráfica y el recuadro de instrucciones de acceso a gráficas.

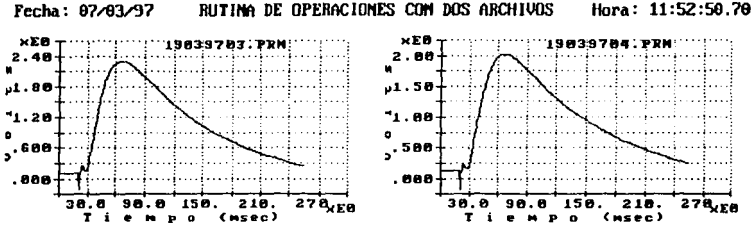


Fig. 5. a) En la rutina de análisis de resultados, en su sección de operaciones con dos archivos, aparece esta pantalla, en la cual ya se han seleccionado y graficado dos archivos, con los cuales se va ha realizar alguna operación que se muestra en el menú de ofertas y que el usuario eligirá.

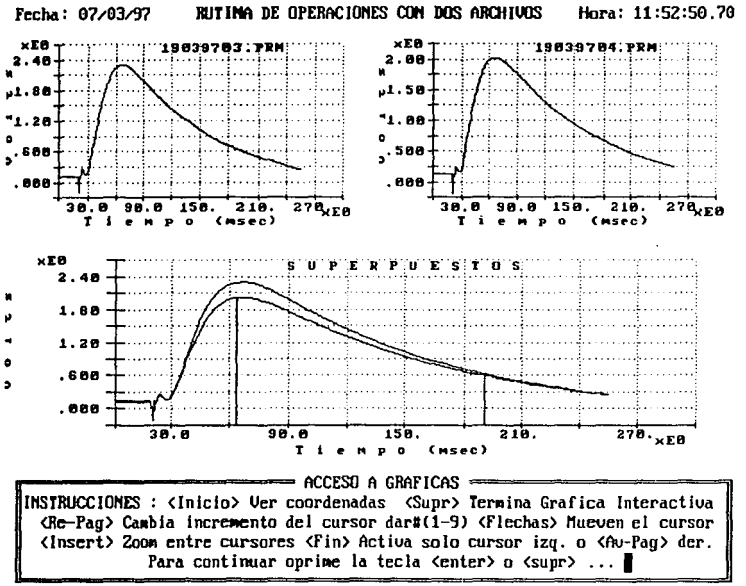


Fig. 5. b) En esta pantalla, se muestra el resultado de una operación realizada con dos archivos, que en este caso fue superponerlos. Se pueden ver las diferencias entre las dos gráficas, y también se puede tener acceso a ellas.

1.2 Estadísticas que pueden realizarse, y algunas ventajas y desventajas de la transferencia de datos a archivos en Excel o Lotus.

La estadística juega un papel importante en el proceso de análisis, al permitir el desarrollo de pruebas de hipótesis contrastando las observaciones experimentales. De la misma manera, al observar los resultados de una serie repetida de experimentos, la magnitud de la variabilidad y la estimación real cambia aparentemente. Tanto la variabilidad y el "mejor" estimador para mediciones experimentales, se obtienen usando estadística descriptiva, siendo las determinaciones más comunes:

a) La media aritmética
b) La mediana
c) La moda
d) La desviación estándar
e) El error estándar de la media

Existen también otras herramientas para el análisis estadístico que pueden aplicarse a la información obtenida del registro de las señales electrofisiológicas. Entre estas se encuentran sólo por mencionar algunas, la distribución Normal, la distribución T de Student, la significancia estadística, y las pruebas que con ellas pueden hacerse; también es común realizar estadísticas no paramétrica, como son la prueba de Wilcoxon y otras.

Debido a estos procedimientos estadísticos el sistema ofrece la oportunidad de poder transferir los datos obtenidos de un experimento a archivos que puedan ser manejados desde Lotus-123 o Excell. Esto se debe a que una hoja de cálculo como las anteriormente mencionadas nos permiten tener un fácil manejo de los datos arrojados por el experimento. Algunas de las ventajas que se obtienen al manejar los datos por medio de alguna hoja de cálculo son las siguientes:

1. Se pueden visualizar cada uno de los datos producto de la digitalización de la señal analógica.
2. Permite combinar tablas y datos numéricos, con textos, y a la vez poder graficar la información obtenida utilizando un formato que el usuario diseña.
3. Permite realizar los análisis estadísticos mencionados, de una manera sencilla y algunos de forma automática.
4. Cuando el archivo transferido haya sido un archivo de registros acumulados, la hoja de cálculo permite seleccionar uno o algunos de los registros que nos interesen para su análisis o graficación, y eliminar los datos restantes.
5. Permite realizar operaciones con los datos transferidos, tal como en la rutina de análisis del sistema.

Si bien son atractivas las ventajas que ofrece el uso de una hoja de cálculo, también tiene grandes desventajas para el análisis y presentación de resultados como son:

1. El hecho de que no pueden medir tiempos y amplitudes por medio de cursores móviles y de forma automática.

2. No permite medir áreas que se encuentran entre dos límites que son establecidos por el usuario.
3. Las operaciones entre dos archivos necesariamente necesitan de alguna instrucción propia del paquete en uso.

Hemos mencionado algunos de los procedimientos que se necesitan para presentar los resultados obtenidos del registro de señales electrofisiológicas, por medio de la computadora. Es por esto que observando las ventajas y desventajas que ofrece la transferencia de datos dentro del sistema para la adquisición y análisis de señales electrofisiológicas, podemos decir que la información obtenida por la rutina de análisis de resultados del propio sistema, y los análisis ofrecidos por Lotus o Excell, nos permiten realizar una adecuada presentación de resultados.

2. EVALUACIÓN

Para realizar una evaluación procederemos primeramente a analizar aspectos propios del sistema de cómputo para adquisición y análisis de señales electrofisiológicas, y a continuación haremos una evaluación propia del "software" utilizado para su programación, que en este caso es ASYST.

2.1 Aspectos propios del Sistema

Existen algunas características propias del sistema, de las cuales hablaremos brevemente

- Promediación entre las repeticiones de un registro

Debido al análisis estadístico ya mencionado, es preciso calcular la tendencia y la dispersión en torno a la misma, por eso es necesario realizar un promedio de las señales que se están capturando. Lo que se promedia son los valores obtenidos en los instantes correspondientes dentro de los diferentes registros. Ya que utilizamos una conversión analógica digital para que los registros puedan ser manejados por la computadora, se tiene una lista de valores en un arreglo de datos correspondientes a muestras de la señal espaciadas por intervalos iguales de tiempo.

Tomando los valores obtenidos por una conversión A/D y que corresponden a un mismo intervalo de tiempo, estos datos son sumados y acumulados para posteriormente divididos entre el número de repeticiones del evento, obteniendo así un promedio. Como este cálculo se repite para todas las muestras que componen el registro, se obtiene un registro promedio. La *Fig. 1* b) de la seección anterior, muestra la diferencia existente entre los registros que son almacenados en archivos tipo acumulados y el archivo tipo promedio.

En los registros acumulados de la *Fig. 1* inciso b) se muestran las señales mezcladas con ruido aleatorio, que es lo que se trata de evitar.

Para eliminar el ruido y dejar la señal mas limpia, se usa un filtro que elimine dicho ruido. La forma mas sencilla de hacerlo dentro de una computadora consiste en promediar a cada uno de los puntos de la señal con los valores de las otras señales, y así obtener el registro promedio (ver *Fig. 1* b)).

- Captura de datos analógicos y conversión a digitales por medio de un ciclo

Esta característica del sistema está dada por la restricción que nos marca el manejador externo DAS debido a que en modo sincrónico no se pueden realizar más de una conversión analógica a digital por ciclo.

Para resolver este problema se programó un ciclo (loop) que nos permitiera realizar la captura y conversión de información analógica a digital. Este ciclo aparece tanto en las definiciones de dos puntos llamadas Capturaarreglo y Capturaarreglos, para uno y dos canales respectivamente. Inicialmente el control de la conversión lo tiene la tarjeta de adquisición, para poder sincronizar el inicio del muestreo, y posteriormente el control lo toma la propia computadora para poder realizar las adquisiciones. En resumen la adquisición de datos se realiza bajo el reloj de la computadora.

- El sistema cumple con las características que debe de tener un sistema de cómputo.

El sistema cumple con la característica de ser comprensible, ya que esta estructurado de tal manera que su diseño sigue una lógica sencilla y clara que modela el problema en forma real. Además los programas tienen ventanas de instrucciones cuando el usuario tiene que responder a un requerimiento hecho por el mismo, y tienen múltiples validaciones en la información que este maneja. Los archivos del sistema tienen la información ordenada, primeramente se encuentra el arreglo que contiene los tiempos del registro, y a continuación el arreglo o los arreglos, según sea el caso, con los datos del voltaje obtenido de la conversión analógico a digital de cada registro o del promedio de los mismos.

Cumple con ser modificable, ya que se estructuró en base a módulos, por lo que las definiciones de dos puntos que se definieron en cada módulo pueden localizarse fácilmente así como entender la función que desempeña, por lo que su modificación se facilita y no altera el funcionamiento de los otros módulos que componen el sistema.

La última característica que cumple es la de ser confiable, pues se hicieron las pruebas y ajustes necesarios antes de liberarlo y actualmente está en uso, arrojando los resultados y cumpliendo con los objetivos para los cuales fue creado.

2.2 Evaluación del "Software" utilizado (ASYST).

Existen muchas características de ASYST que lo hacen ser un "software" confiable, sencillo de utilizar y programar. Muchas de esas características (fácil manejo de arreglos de datos, de adquisiciones A/D, etc.) ya se mencionaron en el Capítulo II por lo que en esta sección nos limitaremos a listar algunas limitantes del mismo y en los casos en que sea posible propondremos una solución alternativa.

Las principales dificultades que presenta el manejo y programación en ASYST, son las siguientes:

- Nos limita en el manejo de archivos de información, principalmente si esta se compone de información no numérica.
- El manejo de la pila puede causar contratiempos cuando no se tiene experiencia en su manejo,
- La notación para el desarrollo de fórmulas es complicada.
- No permite el uso de variables locales en caso de estar programando.
- No permite definir funciones que optimicen la programación.

Limita el manejo de Archivos de Información

Cuando se están manejando datos que son cadenas de caracteres, surge una limitante en el manejo de archivos en ASYST, ya que estos archivos están diseñados para almacenar información numérica, ya sea en forma de escalares o de arreglo de datos. Permite almacenar datos numéricos o arreglos numéricos dándoles un número de subarchivo sin importar la estructura que puedan tener. Una alternativa para el almacenamiento de información no numérica es la creación de arreglos de cadenas de caracteres, lo que puede dar una solución al problema.

El manejo de la pila puede causar confusión

Como ya se mencionó en el Capítulo II, ASYST maneja las variables y constantes por medio de una pila de escalares o de símbolos según el tipo de datos de que se trate. Lo que puede causar confusión al usuario o al programador en el manejo de la pila, es el hecho de que los elementos ya sean variables o constantes que se vayan mencionando, quedan almacenados en la pila hasta que se haga uso de ellos, de lo contrario permanecerán de forma permanente en la pila, o pueden ser utilizados por equivocación en alguna operación. Debido a esto, se debe de tener un perfecto control de los elementos que entran y salen de la pila para no crear confusión en las operaciones que ahí se realizan. Por ejemplo, si en la pila ha quedado algún numero X, y se desea realizar la operación A + B, pero el programador comete el error y sólo indica la operación A +, omitiendo a B, lo que en realidad va a realizar ASYST es la suma de X + A, y no marcará error ninguno, si además la operación se encuentra en un programa que utilizará el resultado de esa operación para realizar a su vez otras operaciones, se desencadenarán una serie de errores que no serán marcados, pero que arrojarán resultados erróneos. Es por esto que se recomienda tener un buen control de los elementos que entran y salen de la pila.

Además en la pila de símbolos almacenará en forma de valores lógicos (falso o verdadero) el resultado de todas aquellas comparaciones o validaciones que se realicen, lo que también puede causar confusión en el manejo de la pila de símbolos.

La notación para el manejo de fórmulas es complicado

ASYST maneja un tipo de notación especial para realizar cualquier tipo de operación en la pila. A este tipo de notación se le llama "posfija". En ella es necesario tener primeramente los números, ya sean variables o constantes, en la pila antes de que la operación pueda ser ejecutada. Los resultados de las operaciones que se vayan realizando serán colocados en el tope de la pila para posteriormente ser recuperados o utilizados en otras operaciones.

Algunos ejemplos en el manejo de esta notación se dan a continuación, escribiendo primero la forma normal en que se escribe la operación y en seguida la forma en como debe de indicarle a ASYST que realice la operación. En aquellos donde se manejan variables, se sobreentiende que estas fueron primeramente definidas, e inicializadas con algún valor establecido.

Notación Normal	Notación en Asyst
5 + 3	5 3 +
(13 + 4) / (2 * 4)	13 4 + 2 4 * /
45 - (12 / 4)	45 12 4 / -
- (15.3 * 4 + 16 - 5) / 3.5	15.3 4 * 16 + 5 - NEG 3.5 /
(A + B) / (X / Y) * 2	A B + X Y / 2 * /
X * Y + (((A / B) - Y) * X)	X Y * A B / Y - X * +
A + (B - (X * (Y / (A + B))))	A B X Y A B + / * - +
$-B+\sqrt{(B^2 - 4*A*C)}/2*A$	1 NEG B * B B * 4 A * C * - SQRT + 2 A * /

No permite el uso de variables locales en caso de estar programando

Una de las limitantes de ASYST cuando se está programando, es decir cuando se están creando definiciones de dos puntos, es el hecho de que no podemos crear variables locales dentro de estas definiciones de dos puntos. Lo que sucede es que si definimos alguna variable en una de las definiciones de dos puntos, y a dicha definición se le manda llamar varias veces para su ejecución, la variable definida en ella se creará tantas veces como el número de llamadas que se haga a la definición de dos puntos. La variable aparecerá en el directorio de ASYST ese número de veces, y estará utilizando área de memoria que será inútil, ya que sólo tomará en cuenta la última definición que se hizo de la variable.

Lo que pasa en este procedimiento, es que las variables no pueden ser eliminadas del directorio cuando se ha terminado de ejecutar una definición de dos puntos, para ser creadas nuevamente en la siguiente llamada. La única manera de borrar una palabra del directorio de ASYST es utilizar la instrucción Forget, pero esta se encuentra entre las instrucciones prohibidas en la creación de los sistemas de corrido.

No permite definir Funciones

ASYST no permite definir funciones que realicen una tarea específica, que regresen directamente el valor obtenido para ser utilizado por el programa principal. Podríamos decir lo mismo de los procedimientos en los cuales se mandan una serie de parámetros, ya sea en forma de constantes o de variables y se ejecuta una tarea específica pero no se regresa ningún valor obtenido durante el procedimiento. Sin embargo como veremos a continuación las definiciones de dos puntos pueden hacer la misma tarea de los procedimientos, sin utilizar variables locales que generalmente se usan en estos.

A continuación se especifica una forma alternativa para realizar procedimientos y funciones en ASYST. Las variables que en ellas se especifican, deben haberse definido con anterioridad a la llamada de el procedimiento o de la función.

La definición de dos punto que hace las funciones de un procedimiento se definiría de la siguiente manera:

```
: Procedimiento

    { Conjunto de instrucciones }

;

X1 X2 ... Xn   Procedimiento
```

La llamada al procedimiento tiene primeramente los parámetros X1,X2,..,Xn que serán almacenados en la pila, durante la ejecución de la definición de dos puntos podrán ser utilizados por la misma tomándolos de la pila. Si las variables sufren modificaciones en su valor durante la ejecución de esta definición de dos puntos, dichas modificaciones tendrán efecto durante el resto del programa. También es necesario que dicha definición de dos puntos deje la pila vacía para que pueda ser tomada como un procedimiento, ya que de esta manera ejecutaría una tarea específica y no regresaría valor alguno.

Para definir una función se necesitaría también de una definición de dos puntos como la siguiente:

```
: Función

    { Conjunto de instrucciones al final de las cuales se coloca un valor en la pila }

;

X1 X2 ...Xn Función nomvar :=
```

La llamada de la función tiene al igual que en el procedimiento los parámetros X1,X2,...,Xn que serán utilizados por la función para realizar una tarea específica. Dichos valores también serán tomados del tope de la pila durante la ejecución de Función, y podrá modificarse su valor. Para que pueda regresar un valor y se cumpla con las características de una función, antes de terminar la ejecución de la definición de dos puntos que la describe, se deberá colocar el valor en el tope de la pila correspondiente a la tarea específica que se le encomendó a la función. Dicho valor es almacenado en una variable, que en este caso lleva el nombre de nomvar y que se definió con anterioridad al llamado de la función.

CONCLUSIONES

Una de las herramientas más poderosas que ha desarrollado el hombre, en cuanto a aplicaciones científicas se refiere, sin duda alguna es la **Computadora**. La computadora ha servido para dar apoyo técnico a las diversas áreas de la ciencia al realizar análisis eficientes y veloces que permiten darle mayor dinamismo y fluidez a las investigaciones en curso, o bien al posibilitar cálculos que antes se antojaban impensables. El desarrollo tecnológico ha sido un gran apoyo a las investigaciones científicas, pero a su vez éste ha surgido, crecido y se ha expandido como consecuencia de los resultados obtenidos de esas investigaciones.

Como ya lo mencionamos anteriormente, la computadora ha servido para dar apoyo a las diversas áreas de la ciencia, en donde la **Actuaría** no podía ser la excepción. Por lo que desde el surgimiento de esta valiosa herramienta, el Actuario ha procurado integrarse a los estudios de los sistemas de cómputo.

Para poder desarrollar un sistema de cómputo, se debe de tener previamente un conocimiento general del área para la cual se va ha realizar dicho sistema, por lo que surgen diversas relaciones de la gente que se dedica al desarrollo de sistemas con otras áreas del conocimiento.

El planteamiento de una metodología para poder realizar sistemas de cómputo es de suma importancia, ya que éste debe aplicarse sin tomarse en cuenta el área para la cual se esta desarrollando el sistema.

Una de las áreas de investigación científica en la cual la computadora se ha convertido en herramienta de apoyo indispensable es la electrofisiología. Desde el surgimiento de las primeras computadoras, se han creado diversos métodos que dependiendo de la tecnología disponible en cada momento, han permitido almacenar y procesar información generada por las señales fisiológicas. Es por eso que el desarrollo de paquetes de cómputo que faciliten el control y análisis de los protocolos experimentales cobra una gran importancia.

Para que las operaciones de almacenamiento y análisis de información generada por señales fisiológicas se realicen mediante una computadora, se necesita que los equipos de registro, amplificación y medición sean compatibles y bien seleccionados. La Tarjeta de Adquisición de Datos tiene una gran importancia como interface entre los instrumentos y la computadora, pues realiza la conversión de las señales generadas por el experimento en información que pueda ser utilizada por la computadora, y viceversa.

La gran importancia de la computadora en el desarrollo de las investigaciones, ha impulsado la diversificación de las técnicas de procesamiento y de las interfaces de comunicación, entre ellas las más importantes son: programando, el acceso directo a memoria, el manejo de interrupciones y memoria compartida.

Existen diversos factores externos (ruido eléctrico y otros) que pueden alterar la información generada por las señales electrofisiológicas, por lo que una vez optimizadas las conexiones, es necesario utilizar alguna técnica o instrumento de filtrado. Es muy utilizada la técnica de promediación de una serie de réplicas.

Una digitalización apropiada de la información analógica generada por el experimento, nos permite confiar en la información digital que manejará la computadora, evitando pérdidas importantes. Es aquí en donde toma gran importancia la frecuencia de muestreo para una adecuada digitalización de la información analógica.

ASYST es un intérprete enfocado al desarrollo de aplicaciones científicas, entre sus grandes atractivos se hallan las facilidades que otorga para la adquisición de datos y para efectuar el análisis de las señales capturadas. Permite realizar adquisiciones de datos con una gran variedad de tarjetas de adquisición de diversas marcas y modelos, sin requerir de gran experiencia en programación, pues las adquisiciones pueden hacerse inmediatamente por medio de una plantilla, sin la necesidad de realizar un programa.

Por ser ASYST un "software" orientado a las aplicaciones científicas, tiene limitaciones en ciertos aspectos que son importantes para el área administrativa, como son el manejo de información no numérica, pero por otra parte permite el fácil manejo de arreglos de datos y el uso de algunas funciones matemáticas (en estadística principalmente) que se encuentran en bibliotecas específicas (overlays).

ASYST es un "software" que ofrece algunas aplicaciones muy atractivas para el **Actuario**, ya que se pueden desarrollar análisis estadísticos de una manera sencilla, y que en algún lenguaje de programación sería mas compleja su programación. De la misma manera el fácil manejo de fórmulas y datos numéricos es una opción muy atractiva para el desarrollo de programas en el área actuarial.

En el desarrollo de un sistema de cómputo se debe de lograr que la estructura del sistema modele la estructura natural del sistema.

Los principales problemas que debemos considerar al desarrollar un sistema de cómputo, se refieren a los costos, tiempos, mantenimiento, contabilidad y complejidad, además desde luego, de un adecuado cumplimiento de los requisitos del usuario.

Un sistema de cómputo debe reunir obligadamente varias características: ser comprensible, modificable y confiable. En la actualidad es necesario agregar una más, ser **expandible**, pues las necesidades de los usuarios aumentan constantemente, y los sistemas deben de ofrecer la oportunidad de crecer sin tener que modificar su estructura primaria.

El buen uso de las técnicas de diseño de programación, como son la modularidad y el diseño descendente (top-down), nos permiten desarrollar sistemas de cómputo que cumplan con las todas características mencionadas, incluyendo expandibilidad.

APÉNDICE

```
\*************************************************************************
\ ************************************************************************
\
\  Universidad Nacional Autonoma de Mexico
\  Facultad de Ciencias
\  Laboratorio de Biofisica del Control Neuromuscular
\  Profesora responsable del laboratorio: Dra Hortensia Gonzalez Gomez
\
\   ---------------------------------------------------------
\      SISTEMA DE REGISTRO ANALISIS Y GRAFICACION DE EXPERIMENTOS
\            ELECTROFISIOLOGICOS EN MEDULA ESPINAL
\   ---------------------------------------------------------
\
\  El presente sistema se desarollo en ASYST, traductor que permite
\  realizar adquisiciones de datos por medio de la tarjeta DAS40-G2.
\
\            Programó: CALLEJAS CHAVERO ALFREDO
\
\
\ ************************************************************************
\*************************************************************************

\                                                    *** **********************
\                                                    *     PROGRAM1.DMO     *
\                                                    **********************

\*************************************************************************
\     Definicion de arreglos, templados y del tipo de impresora
\*************************************************************************

1024 stack pila   \ Define 1 Kbyte de memoria para el uso de la pila

\ Selecciona el tipo de impresora
142 " PRN" BD.PRINTER
portrait bd.page


\ templados y tarjeta
das40
0 0 a/d.template adquis1ch.template    \ Es para un canal
0 1 a/d.template adquis2ch.template    \ Es para dos canales

\ arreglos
integer dim[ 5 ] array datos.buffer
token arreglo1ch.buffer        \ Aquí se almacenan los datos capturados por un canal
```

```
token arreglo2ch.buffer     \ Aquí se almacenan los datos capturados por los dos canales
token tiempo.data           \ Se almacenan los tiempos correpondientes a cada dato capturado
token array1.data           \ Son dos arreglos auxiliares para recuperar solo algunos datos del
token array2.data           \ arreglo original
token suma.data             \ Guarda la suma acumulada de los datos que se van capturando
token promedio.data         \ Almacena los promedios correspondientes a los datos capturados

\ ************************************************************************
\       Definicion de variables enteras, reales y strings
\ ************************************************************************

\ enteras
integer scalar ?escribe.archivo     \ Son banderas que nos indican si se van a grabar los
integer scalar ?escribe.archivo2    \ archivos promedio y acumulados
integer scalar numopc               \ Guarda la opción que selecciona el usuario en los diferentes menús
integer scalar repeticiones         \ Sirve para especificar el número de capturas que se realizarán
integer scalar numcan               \ Guarda el número de canales que van a utilizarse para muestrear
integer scalar varlog               \ Es una bandera que permite saber cuando se ha ejecutado una rutina
integer scalar cont                 \ Cuenta las capturas que se van realizando dato por dato
integer scalar numcom               \ Especifica el número de comentarios en un archivo
integer scalar numsub               \ Especifica el número de subarchivos que contiene un archivo
integer scalar numarch              \ Guarda el número de archivos que se van a grabar
integer scalar numrep               \ Cuenta el número de capturas finalizadas que se han realizado
integer scalar espmsec              \ Especifica el tiempo de retardo entre muestras
integer scalar colorfondo           \ Especifica el color para el fondo de la pantalla,
integer scalar coloraxisas          \                   para los ejes en la gráfica,
integer scalar coloretiquetas       \                   para las etiquetas de los ejes,
integer scalar colortitulos         \                   para los títulos de la gráfica,
integer scalar colorlinea           \                   para la línea de graficación,
integer scalar colorcomentarios     \                   para los comentarios realizados.
integer scalar copias               \ Guarda el número de copias en una impresión
integer scalar switch               \ Son banderas que nos permiten saber cuando una palabra ha sido
integer scalar sw                   \ llamada por la rutina de análisis o de graficación respectivamente.
integer scalar divhor               \ Guarda el número de líneas horizontales que se desean aparezcan en
integer scalar divver               \ la gráfica, y también las verticales.
integer scalar swdesp               \ Nos permite saber si la operación con dos archivos va a impresión
dp.integer scalar muestras          \ Especifica el número de muestras por registro
dp.integer scalar coordin           \ Guardan las coordenadas tan to inicial como final que nos permiten
dp.integer scalar coordfin          \ limitar las áreas seleccionadas por el usuario

\ reales
real scalar tiempofinal             \ Almacena el tiempo de duración de un registro
real scalar area                    \ Almacena el área calculada entre dos coordenadas
real scalar coordenadax1            \ Guardan los valores de las coordenadas que en ese momento
real scalar coordenadax2            \ tienen los cursores que puede manejar el usuario, tanto
real scalar coordenaday1            \ para el eje X,
real scalar coordenaday2            \ como para el eje Y.

\ strings
40 string nombre.archivo            \ Guarda los nombres que da el usuario de los archivos en los que se
40 string nombre.archivo1 \ desea grabar información, o de los que se desea recuperar información,
40 string nombre.archivo2 \ así como dos sirven de auxiliares par confirmar la existencia o no
40 string nombre.archivo3 \ existencia de un archivo.
70 string comentario                \ Almacena los comentarios realizados o guardados en un archivo
80 string titulo                    \ Guarda los títulos que desean se coloquen en las gráficas
```

```
70 string directorio                \ Almacena el directorio de trabajo.

\ ********************************************************************************
\       Definicion de ventanas de uso en modo normal y grafico
\ ********************************************************************************

normal.display
1 1  29 79 window presenta
17 2  20 77 window ventana1
15 2  20 77 window ventana2
1 2  15 77 window ventana3
graphics.display
25 2 29 79 window recuadro
24 0 29 80 window recuadro2
2 1 22 77 window recuadro3
1 0 23 80 window recuadro4

normal.display
stack.clear


\ ********************************************************************************
\ Definicion de Vuports de graficacion para su uso posterior en modo grafico
\ ********************************************************************************

graphics.display
vuport.clear

\ El vuport ocupa toda la parte correspondiente al modo grafico.
  vuport general
  0. .25 vuport.orig
  1. .70 vuport.size
  vuport.clear

\ Ocupa el lugar superior de la pantalla.
  vuport arriba
  0. .60 vuport.orig
  1. .35 vuport.size
  vuport.clear

\ Ocupa el lugar inferior de la pantalla.
  vuport abajo
  0. .23 vuport.orig
  1. .35 vuport.size
  vuport.clear

\ Ocupa el lugar superior izquierdo de la pantalla.
  vuport arribaizq
  0. .60 vuport.orig
  .35 .35 vuport.size
  vuport.clear

\ Ocupa el lugar superior derecho de la pantalla.
  vuport arribader
  .38 .60 vuport.orig
```

```
.60 .35 vuport.size
vuport.clear

\ Ocupa el lugar inferior izquierdo de la pantalla.
vuport abajoizq
0. .23 vuport.orig
.35 .35 vuport.size
vuport.clear

\ Ocupa el lugar inferior derecho de la pantalla.
vuport abajoder
.38 .23 vuport.orig
.60 .35 vuport.size
vuport.clear

\ Ocupa el lugar superior izquierdo de la pantalla.
vuport supizq
0. .62 vuport.orig
.48 .33 vuport.size
vuport.clear

\ Ocupa el lugar superior derecho de la pantalla.
vuport supder
.52 .62 vuport.orig
.48 .33 vuport.size
vuport.clear


\ ************************************************************************
\                  Inicializacion de variables
\ ************************************************************************

500 muestras :=
10 repeticiones :=
1 numcan :=
1 varlog :=
4 numopc :=
0 ?escribe.archivo :=
0 colorfondo :=
6 coloraxisas :=
2 coloretiquetas :=
7 colortitulos :=
15 colorlinea :=
7 colorcomentarios :=
1 copias :=
1 sw :=
0 swdesp :=


\ ************************************************************************
\ Inicializacion de colores de el fondo y cursores para los diversos vuports
\                  definidos con anterioridad.
\ ************************************************************************

general colorfondo vuport.color    coloraxisas axis.color  14 cursor.color
```

```
     coloretiquetas label.color colorlinea color
arriba colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
abajo  colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
arribaizq colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
abajoizq  colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
arribader colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
abajoder  colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
supizq  colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
supder colorfondo vuport.color   coloraxisas axis.color  14 cursor.color
     coloretiquetas label.color colorlinea color
normal.display



\*************************************************************************
\        MODULOS   INDEPENDIENTES
\
\      Palabras generales usadas en todas las rutinas del sistema
\*************************************************************************

\*****************************
\ MODULO PIDE CLAVES
\*****************************

\ Entra en un loop hasta que se de una clave o llave.
: espera.una.clave
 key 0 =
 if
   key drop
 then
;

\ Espera un numero real que se de del teclado y repite si hay una
\ entrada invalida como es alguna letra u otro caracter diferente.
: #entrada&checa
 BEGIN
  ?rel.col ?rel.row      \ salva la columna y linea en caso de error
  #input                 \ recibe un numero
  not                    \ lo valida ?
 WHILE
  bell                   \ Da un beep! si no es un numero
  ?rel.col 3 pick - 1 +  \ da la actual col
  3 pick 3 pick goto.xy  \ pone la posicion de origen
  spaces                 \ borra la entrada dada
  goto.xy                \ regresa a la posicion de origen
 REPEAT
 unrot 2 *drop
;
```

```
\ Espera un string que se de del teclado y repite si hay una
\ entrada invalida.
: "entrada&checa
  BEGIN
    "input              \ recibe un string
    "len 0 =            \ es un string de longitud 0 ?
  WHILE
    bell "drop              \ Da un beep! si no es un string
  REPEAT
;

\ Espera una respuesta de Si o No
: DAR.S/N
BEGIN
  ?REL.COL ?REL.ROW
  PCKEY NOT
  IF
    DUP 84 >
    IF
      32 -
    THEN
    DUP 83 = NOT
    DUP 78 = NOT AND
  ELSE
    TRUE
  THEN
WHILE
  BELL
  DROP
  60 3 goto.xy  ." <escribe S o N>"
  2000 MSEC.DELAY
  60 3 goto.xy  ."                "
  ?REL.COL 3 PICK - 1 +
  3 PICK 3 PICK GOTO.XY
  SPACES
  GOTO.XY
REPEAT
UNROT 2 *DROP
83 =
;


\ ******************************
\  MODULO PINTA ENCABEZADOS
\ ******************************

\ Esta rutina pinta etiquetas1 para las graficas de adqisicion
: pintaetiquetas1
  normal.coords
  sw 1 =
  if
    3 color
  else
    colortitulos color
```

```
  then
  .00  .350 position 90 char.dir 90 label.dir " V o l t s" Label
  .405 .030 position 0 char.dir 0 label.dir " T i e m p o  (msec)" label
  15 color
  world.coords
;

\ Esta rutina pinta etiquetas2 para las graficas de adqisicion
: pintaetiquetas2
  normal.coords
  3 color
  .00  .350 position 90 char.dir 90 label.dir " V o l t s" Label
  .230 .030 position 0 char.dir 0 label.dir " T i e m p o  (msec)" label
  15 color
  world.coords
;

\ Esta rutina pinta los encabezados de las graficas de adquisicion para Acum.
: pintaencabezado1
  11 color
  normal.coords
  numcan 1 =
  if
    .400 .930 position " A c u m u l a d o s" label
  else
    .200 .930 position " A c u m u l a d o s" label
  then
  world.coords
  15 color
;

\ Esta rutina pinta los encabezados de las graficas de adquisicion para Prom.
: pintaencabezado2
  11 color
  normal.coords
  .400 .930 position "  P r o m e d i o " label
  world.coords
  15 color
;

\ Esta rutina pinta los nombres de los archivos que se recuperan
: pintaencabezado3
  sw 1 =
  if
    11 color
  else
   colortitulos color
  then
  normal.coords
  .450 .930 position nombre.archivo label
  world.coords
  15 color
;

\ Pinta el numero de repeticiones que se estan realizando
```

```
: pintarep
  numrep 1 + numrep :=
  10 foreground
  70 1 goto.xy numrep .
  15 foreground
;

\ **********************************
\  MODULO MANEJO ARCHIVOS
\ **********************************

\ Muestra los archivos tipo WKS que ya han sido
\ grabados
: muestraarchivoswks
  15 foreground
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  7 foreground
  3 3 goto.xy ." A continuacion se despliegan los archivos tipo (.WKS) "
  3 4 goto.xy ." creados con anterioridad en el directorio " .dir
  3 5 goto.xy ." Si deseas parar el desplegado utiliza la tecla <pausa>... "
  3 6 goto.xy ."       Da una tecla para continuar ... "
  espera.una.clave
  dir *.wks
;

\ Muestra los archivos tipo promedio y tipo acumulados que ya han sido
\ grabados
: muestraarchivos
  15 foreground
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  7 foreground
  3 3 goto.xy ." A continuacion se despliegan los archivos promedio(.prm) y "
  3 4 goto.xy ." acumulados(.acu) del directorio " .dir
  3 5 goto.xy ." Si deseas parar el desplegado utiliza la tecla <pausa>... "
  3 6 goto.xy ."       Da una tecla para continuar ... "
  espera.una.clave
  dir *.prm
  dir *.acu
;

\ Selecciona un archivo de datos ya creado y que se desea accesar a los
\ subarchivos que contiene.
: selecciona.archivo.entrada
?graphics.display
if
  swdesp 0 =
  if
    recuadro3
    muestraarchivos
  then
  recuadro
else
  ventana3
```

```
    muestraarchivos
    ventana1
  then
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  7 foreground
  8 0 goto.xy ." Dar: Nombre(8 car)+Extension(.PRM o .ACU segun sea el caso)"
  15 foreground
  BEGIN
    8 2 goto.xy ." Escribe el nombre del archivo : " "entrada&checa
    nombre.archivo1 ":=
    ascii , nombre.archivo1 "compress
    nombre.archivo1 ":=
    ascii ; nombre.archivo1 "compress
    nombre.archivo1 ":=
    nombre.archivo1 nombre.archivo ":=
    nombre.archivo defer> file.sizes
    0 = 0 = and
  WHILE
    10 3 goto.xy ." Archivo inexistente, revisa tu lista de archivos creados..."
    3000 msec.delay
    10 3 goto.xy ."                                                    "
    8 2 goto.xy ."                                                     "
  REPEAT
  ?graphics.display
  if
    swdesp 0 =
    if
      recuadro3
      screen.clear
      recuadro4
      screen.clear
    then
    recuadro
  else
    ventana3
    screen.clear
    presenta
  then
  ;

\ ******************************
\  MODULO MANEJO GRAFICAS
\ ******************************

\ Rutina que despliega las instrucciones para poder realizar mediciones
\ en las graficas de promedios.
: menumedicion
  15 foreground
  screen.clear
  186 205 201 200 188 187 border.chars {BORDER}
  7 foreground
  0  0 goto.xy ." INSTRUCCIONES :"
  1 foreground
  16 0 goto.xy ." <Inicio> "
```

```
 42 0 goto.xy ." <Supr> "
 7  1 goto.xy ." <Re-pag> "
 7  2 goto.xy ." <Flechas> "
 6 foreground
 25 0 goto.xy ." Ver coordenadas "
 49 0 goto.xy ." Termina Grafica Interactiva"
 17 1 goto.xy ." Cambia distancia de incremento en cursores, dar #(1 a 9)"
 17 2 goto.xy ." Mueven el cursor arr,abj,der,izq, segun sea el caso"
 7 foreground
 14 3 goto.xy ." Para continuar oprime la tecla <enter> o <supr> ... "
 {def}
 9 foreground
 32 25 goto.xy ."  ACCESO A GRAFICAS "
 15 foreground
 recuadro
 66 3 goto.xy
 15 foreground
 graphics.readout
 normal.coords
 numcan 1 =
 if
  .66 .980 readout>position
 else
  .5 .980 readout>position
 then
 world.coords
;

\ Palabra que activa los dos cursores verticales
: cursores
 recuadro
 1 foreground
 2  1 goto.xy ." <Re-Pag> "
 49 1 goto.xy ."  <Flechas> "
 2  2 goto.xy ." <Insert> "
 30 2 goto.xy ."  <Fin> "
 62 2 goto.xy ."  <Av-Pag> "
 6 foreground
 11 1 goto.xy ." Cambia incremento del cursor dar#(1-9)"
 60 1 goto.xy ." Mueven el cursor"
 11 2 goto.xy ." Zoom entre cursores"
 37 2 goto.xy ." Activa solo cursor izq. o"
 72 2 goto.xy ." der."
 15 foreground
 66 3 goto.xy
 array.readout
 normal.coords
 .66 .980 readout>position
 world.coords
;

\ Palabra que pide o graba las coordenadas de los cursores
: coordcurs
  BEGIN
   5  23 goto.xy ." Coloca el cursor izquierdo en la "
```

```
   38 23 goto.xy ." coordenada deseada y da <enter>... "
   "input
   "drop
   5  23 goto.xy ."                                          "
   50 23 goto.xy ."                        "
   ?cursor
   coordenaday1 := coordenadax1 :=
   coordenadax1
   coordin :=
   coordin 0 <
  WHILE
   37 23 goto.xy ." Coordenada invalida ... "
   2000 msec.delay
   37 23 goto.xy ."                        "
  REPEAT
  BEGIN
   5  23 goto.xy ." Coloca el cursor  derecho  en la "
   38 23 goto.xy ." coordenada deseada y da <enter>... "
   "input
   "drop
   5  23 goto.xy ."                                          "
   50 23 goto.xy ."                        "
   ?cursor
   coordenaday2 := coordenadax2 :=
   coordenadax2
   coordfin :=
   coordfin coordin < coordfin coordin = or coordfin 1 < or
  WHILE
   37 23 goto.xy ." Coordenada invalida ... "
   2000 msec.delay
   37 23 goto.xy ."                        "
  REPEAT
;

\ Palabra que pregunta al usuario  si desea relizar impresiones y de que tipo
: piderespimp
   recuadro
   screen.clear
   186 205 201 200 188 187 border.chars {border}
   5 1 goto.xy ." Deseas recortar la grafica de resultados (S/N) ? "
   DAR.S/N
   if
    83 emit
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    menumedicion
    cursores
    10 3 goto.xy ."  Recortando coordenadas en X para realizar un ZOOM..."
    {def}
    coordcurs
    tiempo.data []size
    swap drop
    muestras :=
    tiempo.data [ muestras ] tiempofinal :=
    coordin muestras *  tiempofinal / coordin :=
```

```
  coordfin muestras *  tiempofinal / coordfin :=
  switch 2 =
  if
     abajo
     colorlinea color
     array1.data []Max
     array2.data []Max
     >
     if
       array1.data
       sub[ coordin , coordfin , 1 ]
       tiempo.data
       sub[ coordin , coordfin , 1 ]
       swap  xy.auto.plot
       9 color
       array2.data
       sub[ coordin , coordfin , 1 ]
       tiempo.data
       sub[ coordin , coordfin , 1 ]
       swap xy.data.plot
     else
       array2.data
       sub[ coordin , coordfin , 1 ]
       tiempo.data
       sub[ coordin , coordfin , 1 ]
       swap  xy.auto.plot
       9 color
       array1.data
       sub[ coordin , coordfin , 1 ]
       tiempo.data
       sub[ coordin , coordfin , 1 ]
       swap xy.data.plot
     then
  else
     abajo
     promedio.data
     sub[ coordin , coordfin , 1 ]
     tiempo.data
     sub[ coordin , coordfin , 1 ]
     swap xy.auto.plot
  then
else
    78 emit
    7 foreground
    45 3 goto.xy ." <Enter> para continuar"
    15 foreground
    espera.una.clave
then
recuadro
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {border}
1 0 goto.xy ." TITULOS.  Instr: "
7 foreground
19 0 goto.xy ." Manejando las <flechas> coloque el cursor en el lugar en"
```

```
 10 1 goto.xy ." donde desea poner el titulo o leyenda o de <N> si "
 60 1 goto.xy ." desea terminar."
abajo
 colortitulos foreground
 begin
   graphics.readout
   75 1 3 goto.xy
   spaces
   1 3 goto.xy " T1 > " "entrada&checa
   titulo ":=
   titulo " n" "= titulo " N" "= or not
   if
    colortitulos color titulo label
   then
   titulo " n" "= titulo " N" "= or
until
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {border}
10 0 goto.xy ." Cuantas copias necesitas : " #entrada&checa
copias :=
10 1 goto.xy ." Deseas imprimir tal como estan las graficas (S/N) ? "
DAR.S/N not
if
 78 emit
 10 2 goto.xy ." Solamente imprimiras el resultado de la operacion (S/N)? "
 DAR.S/N
 if
  83 emit
  15 foreground
  1 3 goto.xy ." Que formato deseas:  1)Media Pag  2)Un Cuarto Pag "
  9 foreground
  54 3 goto.xy ." Opcion : "
  Begin
    63 3 goto.xy #entrada&checa
    numopc :=
    numopc 1 = not numopc 2 = not and
  while
    63 3 goto.xy ." Opc. invalida"
    2000 msec.delay
    63 3 goto.xy ."              "
  repeat
  15 foreground
  numopc 1 =
  if
    1200 2000 bd.scale
    100 100 bd.res
  else
    1200 1000 bd.scale
    100 100 bd.res
  then
  1 3 goto.xy ."                                        "
  54 3 goto.xy ."         "
  63 3 goto.xy ."           "
  3  3 goto.xy ." Checa que este lista la impresora y da <enter>"
```

```
      52 3 goto.xy ." para imprimir..."
      espera.una.clave
      3 3 goto.xy ."                                    "
      52 3 goto.xy ."                "
      copias 1 + 1 do
        BD.VUPORT
        30 3 goto.xy ." Reporte No. " 1 . ."  mandado a impresion"
        3000 msec.delay
        30 3 goto.xy ."                                 "
        30 3 goto.xy ." Espera que salga la impresion y da <enter>..."
        espera.una.clave
        30 3 goto.xy ."                                 "
      loop
    else
      78 emit
    then
  else
    83 emit
    1200 1000 bd.scale
    100 100 bd.res
    3  3 goto.xy ." Checa que este lista la impresora y da <enter>"
    52 3 goto.xy ." para imprimir..."
    espera.una.clave
    65 3 3 goto.xy
    spaces
    copias 1 + 1 do
       recuadro2
       screen.clear
       0 foreground
       80 0 goto.xy
       BD.SCREEN
       15 foreground
       recuadro
       screen.clear
       186 205 201 200 188 187 border.chars {border}
       30 3 goto.xy ." Reporte No. " 1 . ."  mandado a impresion"
       3000 msec.delay
       30 3 goto.xy ."                                 "
       30 3 goto.xy ." Espera que salga la impresion y da <enter>..."
       espera.una.clave
       30 3 goto.xy ."                                 "
    loop
  then
  1 switch :=
;

\ Palabra que despliega el menu de ofertas de analisis
: menuofertas
BEGIN
 sw 0 =
 if
  abajo colorlinea color
 else
  abajo 15 color
 then
```

```
1 switch <>
if
 {DEF}
 7 foreground
 1  0 goto.xy ." Fecha: " .date
 63 0 goto.xy ." Hora: " .time
 15 foreground
then
recuadro
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {BORDER}
7 foreground
25 0 goto.xy ." M e n u  d e  O f e r t a s"
15 foreground
10 1 goto.xy ." 1) Sumarlos"
10 2 goto.xy ." 2) Restarlos"
30 1 goto.xy ." 3) Dividirlos"
30 2 goto.xy ." 4) Superponerlos"
50 1 goto.xy ." 5) Filtrarlos"
0 background
3 foreground
50 2 goto.xy ." Escribe una opcion : "
Begin
 3 foreground
 71 2 goto.xy #entrada&checa
 15 foreground
 numopc :=
 numopc 1 = not numopc 2 = not and numopc 3 = not and numopc 4 = not and
 numopc 5 = not and
while
 30 3 goto.xy ."       Oprime una opcion valida..."
 3000 msec.delay
 30 3 goto.xy ."                                  "
 71 2 71 2 goto.xy
 spaces
repeat
numopc
case
1 of
 array1.data array2.data +
 becomes> promedio.data
 promedio.data
 tiempo.data swap
 abajo xy.auto.plot
 switch 1 =
 if
  piderespimp
 else
  pintaetiquetas1
  11 color
  normal.coords
  .485 .925 position " S  U  M  A"  label
  world.coords
  15 color
```

```
   menumedicion
   cursores
   "input
   "drop
  then
 endof
 2 of
  array1.data array2.data -
  becomes> promedio.data
  promedio.data
  tiempo.data swap
  abajo xy.auto.plot
  switch 1 =
  if
   piderespimp
  else
   pintaetiquetas1
   11 color
   normal.coords
   .495 .925 position " R E S T A"  label
   world.coords
   15 color
   menumedicion
   cursores
   "input
   "drop
  then
 endof
 3 of
  array1.data array2.data /
  becomes> promedio.data
  promedio.data
  tiempo.data swap
  abajo xy.auto.plot
  switch 1 =
  if
   piderespimp
  else
   pintaetiquetas1
   11 color
   normal.coords
   .420 .925 position " D I V I D I R L O S"  label
   world.coords
   15 color
   menumedicion
   cursores
   "input
   "drop
  then
 endof
 4  of
   array1.data []Max
   array2.data []Max
   >
   if
```

```
    array1.data
    tiempo.data swap
    abajo xy.auto.plot
    9 color
    array2.data
    tiempo.data swap
    xy.data.plot
   else
    array2.data
    tiempo.data swap
    abajo xy.auto.plot
    9 color
    array1.data
    tiempo.data swap
    xy.data.plot
   then
   switch 1 =
   if
    2 switch :=
    piderespimp
   else
    pintaetiquetas1
    11 color
    normal.coords
    .393 .925 position " S U P E R P U E S T O S" label
    world.coords
    15 color
    menumedicion
    cursores
    "input
    "drop
   then
endof
5 of
   array1.data array2.data + 2 /
   becomes> promedio.data
   promedio.data
   tiempo.data swap
   abajo xy.auto.plot
   switch 1 =
   if
    piderespimp
   else
    pintaetiquetas1
    11 color
    normal.coords
    .420 .925 position " F I L T R A D O S" label
    world.coords
    15 color
    menumedicion
    cursores
    "input
    "drop
   then
endof
```

```
 endcase
 14 3 goto.xy ."                                    "
 14 3 goto.xy ."  Deseas realizar otra operacion (S/N) ? "
 DAR.S/N not
UNTIL
3 numopc :=
;

\ ********************************************************************************
\  M O D U L O    A D Q U I S I C I O N   D E   D A T O S.
\ ********************************************************************************

\ Presenta la Rutina de Adquisicion
: presentaadquisicion
screen.clear
presenta
1 background
15 foreground
186 205 201 200 188 187 border.chars {border}
7 foreground
10 0 goto.xy ." U.N.A.M.                          Fac. CIENCIAS"
10 1 goto.xy ."  Laboratorio de Biofisica del control neuromuscular"
10 2 goto.xy ."              Rutina de Adquisicion"
10 3 goto.xy ."                PRESENTACION"
5 4  goto.xy ." ____________________________________________________________________"
15 foreground
5 6  goto.xy ."   La presente rutina realiza la adquisicion de datos por medio"
5 7 goto.xy ." de la tarjeta DAS40, la cual se  encarga de  digitalizar la in-"
5 8 goto.xy ." formacion para poder ser utilizada posteriormente."
5 9 goto.xy ."   Para dicha tarea se usan solamente dos canales, el  #0  y el"
5 10 goto.xy ." #1.En el siguiente menu se preguntara cuantos canales se desean"
5 11 goto.xy ." usar,  en caso de dar  1  por default la rutina asumira  que el"
5 12 goto.xy ." canal a ser usado es el canal #0,  y en caso de dar 2,  asumira"
5 13 goto.xy ." que se usaran conjuntamente los canales #0 y #1."
5 14 goto.xy ."   Se pide tambien el numero de muestras de una adquisicion, en"
5 15 goto.xy ." cuyo caso se debe dar el numero  de veces que  la  maquina debe"
5 16 goto.xy ." de tomar informacion dentro de un mismo trigger(disparo)."
5 17 goto.xy ."   A continuacion pedira el numero de repeticiones de la adqui-"
5 18 goto.xy ." sicion, en donde debe  de  darsele  cuantos  triggers(disparos)"
5 19 goto.xy ." tienen que llevarse a cabo para que de ahi pueda obtenerse un"
5 20 goto.xy ." promedio de todos ellos, eliminando asi el ruido electrico."
7 foreground
10 22 goto.xy ."                Presione cualquier tecla para continuar..."
15 foreground
espera.una.clave
;

\ INICIALIZAR PLANTILLAS
\ ----------------------
\ Inicializa la plantilla que va a ser usada para la adquisicion
\ de datos en un solo canal.
: iniciaplantilla1ch
adquis1ch.template
datos.buffer template.buffer
```

```
muestras integer ramp
becomes> arreglo1ch.buffer
arreglo1ch.buffer template.buffer
cyclic
a/d.init
;

\ Inicializa la plantilla que va a ser usada para la adquisicion
\ de datos en un solo canal.
: iniciaplantilla2ch
adquis2ch.template
datos.buffer template.buffer
muestras integer ramp
becomes> arreglo2ch.buffer
arreglo2ch.buffer template.buffer
\ load camarray
\ integer dim[ muestras , 2 ] array arreglo2ch.buffer
\ arreglo2ch.buffer template.buffer
cyclic
a/d.init
;


\ CREAR ARCHIVOS Y GRABARLOS
\ ----------------------
\ Crea un archivo ya seleccionado
: crea.archivo
 file.template
  numcom comments
 end
 nombre.archivo defer> file.create
;

\ Pide los comentarios y los graba en un archivo previamente abierto y
\ seleccionado.
: graba.comentarios
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 30 0 goto.xy ." Grabando Comentarios"
 nombre.archivo defer> file.open
 numcom 0 = not
 if
  numcom 1 + 1 do
   5 I goto.xy I . ." > " "input
   I >comment
  loop
 else
    15 2 goto.xy ."     No se realizara ningun comentario ... "
    4000 msec.delay
 then
 nombre.archivo defer> file.close
;

\ Selecciona el archivo de salida en el cual se van a almacenar los datos
\ adquiridos.
```

```
: selecciona.archivo.salida
screen.clear
186 205 201 200 188 187 border.chars {border}
7 foreground
8 0 goto.xy ." Dar: Nombre(8 car)+Extension(.PRM o .ACU segun sea el caso)"
numarch 2 =
if
  8 1 goto.xy ."                    Para el canal 0"
then
15 foreground
BEGIN
 8 2 goto.xy ." Escribe el nombre del archivo : " "entrada&checa
 nombre.archivo1 ":=
 ascii , nombre.archivo1 "compress
 nombre.archivo1 ":=
 ascii ; nombre.archivo1 "compress
 nombre.archivo1 ":=
 nombre.archivo1
 nombre.archivo ":=
 nombre.archivo defer> file.sizes
 0 = 0 = and not
WHILE
 8 3 goto.xy ." Archivo ya existente , no podemos regrabarlo ... "
 4000 msec.delay
 8 2 goto.xy ."                                          "
 8 3 goto.xy ."                                "
REPEAT
8 3 goto.xy ." El Archivo no existe.Creamos uno nuevo (S/N)? "
DAR.S/N
if
 83 emit
 8 3 goto.xy ." Cuantos  comentarios  deseas  realizar :  ? " #entrada&checa
 numcom :=
 crea.archivo
 graba.comentarios
 1 ?escribe.archivo :=
else
 78 emit
 0 ?escribe.archivo :=
then
numarch 2 =
if
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 7 foreground
 8 0 goto.xy ." Dar: Nombre(8 car)+Extension(.PRM o .ACU segun sea el caso)"
 8 1 goto.xy ."                    Para el canal 1"
 15 foreground
 BEGIN
  8 2 goto.xy ." Escribe el nombre del archivo : " "entrada&checa
  nombre.archivo2 ":=
  ascii , nombre.archivo2 "compress
  nombre.archivo2 ":=
  ascii ; nombre.archivo2 "compress
  nombre.archivo2 ":=
```

```
  nombre.archivo2
  nombre.archivo ":="
  nombre.archivo defer> file.sizes
  0 = 0 = and not
 WHILE
  8 3 goto.xy ." Archivo ya existente , no podemos regrabarlo ... "
  4000 msec.delay
  8 2 goto.xy ."                                    "
  8 3 goto.xy ."                                  "
 REPEAT
 8 3 goto.xy ." El Archivo no existe.Creamos uno nuevo (S/N) ?"
 DAR.S/N
 if
  83 emit
8 3 goto.xy ." Cuantos  comentarios  deseas  realizar : ? " #entrada&checa
  numcom :=
  crea.archivo
  graba.comentarios
  1 ?escribe.archivo2 :=
 else
  78 emit
  0 ?escribe.archivo2 :=
 then
then
;


\ Graba los archivos promedio
: grabaarchprom
 0 ?escribe.archivo :=
 seleccioná.archivo.salida
 ?escribe.archivo 1 =
 if
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  10 2 goto.xy ."  Grabando  Informacion .... "
  tiempo.data
  nombre.archivo defer> file.open
  append.array>file
  append.array>file
  nombre.archivo defer> file.close
  3000 msec.delay
 else
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  10 2 goto.xy ."  Proceso Abortado, regresamos al sistema .... "
  3000 msec.delay
  drop
 then
;

\ Graba los archivos acumulados
: grabaarchacum
 0 ?escribe.archivo :=
```

```
  selecciona.archivo.salida
  ?escribe.archivo 1 =
  if
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ." Grabando Informacion .... "
    tiempo.data
    nombre.archivo defer> file.open
    repeticiones 2 + 1 do
      append.array>file
    loop
    nombre.archivo defer> file.close
    3000 msec.delay
  else
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ." Proceso Abortado, regresamos al sistema .... "
    3000 msec.delay
    stack.clear
  then
;

\ Graba el primer arreglo en los archivos acumulados del canal 1
: grabaarchacumc0
 nombre.archivo1 defer> file.open
 append.array>file
 nombre.archivo1 defer> file.close
;

\ Graba el primer arreglo en los archivos acumulados del canal 2
: grabaarchacumc1
  nombre.archivo2 defer> file.open
  append.array>file
  nombre.archivo2 defer> file.close
;


\ CAPTURAR ARREGLOS, SUMARLOS Y GRAFICARLOS
\ ------------------------------------------
\ Este ciclo realiza la captura de datos con un solo canal en un solo arreglo.
: Capturaarreglo
ext.trig
a/d.init
a/d.in arreglo1ch.buffer [ 1 ] :=
int.trig
a/d.init
rel.time
muestras 1 + 1 do
   a/d.in arreglo1ch.buffer [ I ] :=
   espmsec msec.delay
loop
rel.time
swap -
tiempofinal :=
;
```

```
\ Esta palabra se encarga de sumar y graficar los datos adquiridos
\ en el (los) canal(es) correspondiente(s).
: sumarygraficar
0 numrep :=
capturaarreglo
screen.clear
15 foreground
186 205 201 200 188 187 border.chars {BORDER}
11 1 goto.xy ."    Espera un momento se esta realizando la adquisicion # "
arreglo1ch.buffer ext.a/d.scale
dup
muestras real ramp
tiempofinal muestras / *
becomes> tiempo.data
tiempo.data
swap
xy.auto.plot
pintaetiquetas1
pintaencabezado1
pintarep
dup becomes> suma.data
repeticiones 1 = not
if
 repeticiones 1 do
    capturaarreglo
    arreglo1ch.buffer ext.a/d.scale
    dup tiempo.data swap xy.data.plot
    pintarep
    dup suma.data +
    becomes> suma.data
 loop
then
;


\ Este ciclo realiza la captura de datos en dos canales
: capturaarreglos
  ext.trig
  a/d.init
  a/d.in
  swap
  arreglo2ch.buffer [ 1 ] :=
  arreglo2ch.buffer [ 2 ] :=
  int.trig
  a/d.init
  1 cont :=
  rel.time
  muestras 2 / 1 + 1 do
     a/d.in
     swap
     arreglo2ch.buffer [ cont ] :=
     1 cont + cont :=
```

```
    arreglo2ch.buffer [ cont ] :=
    1 cont + cont :=
    espmsec msec.delay
  loop
  rel.time
  swap -
  tiempofinal :=
;

\ Este ciclo lo que hace es sumar y graficar los datos de dos canales
\ correspondientes.
: Sumarygraficardos
0 numrep  :=
capturaarreglos
0 background
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {BORDER}
11 1 goto.xy ."    Espera un momento se esta realizando la adquisicion # "
arreglo2ch.buffer ext.a/d.scale
dup
muestras 2 / real ramp
tiempofinal muestras 2 / / *
becomes> tiempo.data
tiempo.data
swap
sub[ 1 , muestras , 2 ]
arribaizq
0 vuport.color
6 axis.color
2 label.color
15 color
xy.auto.plot
pintaetiquetas2
pintaencabezado1
pintarep
dup
tiempo.data swap
sub[ 2 , muestras , 2 ]
abajoizq
xy.auto.plot
0 vuport.color
6 axis.color
2 label.color
15 color
pintaetiquetas2
pintaencabezado1
dup becomes> suma.data
repeticiones 1 = not
if
  repeticiones 1 do
    capturaarreglos
    arreglo2ch.buffer ext.a/d.scale
    dup
    sub[ 1 , muestras , 2 ]
```

```
    arribaizq
    tiempo.data swap xy.data.plot
    pintarep
    dup
    sub[ 2 , muestras , 2 ]
    abajoizq
    tiempo.data swap xy.data.plot
    dup suma.data +
    becomes> suma.data
  loop
then
;

\ PEDIR CANALES MUESTRAS Y REPETICIONES
\ ------------------------------
\ Rutina de captura de canales, muestras que se desean tomar y el
\ numero de repeticiones que se van a realizar.
: pidecanalmuestrayrep
normal.display
screen.clear
presenta
1 background
15 foreground
screen.clear
186 205 201 200 188 187 border.chars (border)
7 foreground
1  0 goto.xy ." Fecha: " .date
60 0 goto.xy ." Hora: " .time
13 2  goto.xy ." U.N.A.M.                    Fac. CIENCIAS "
13 3  goto.xy ."   Laboratorio de Biofisica del control Neuromuscular"
13 5  goto.xy ."          RUTINA DE ADQUISICION DE DATOS  "
10 6  goto.xy ." ____________________________________________________________________________"
15 foreground
13 9  goto.xy ."              Menu de Seleccion"
8  12  goto.xy ." A) Escribe el numero de  canales de  muestreo  (1 o 2) :"
8  14  goto.xy ." B) Escribe el numero de  muestras  para  la  adquisicion :"
8  16  goto.xy ." C) Escribe el numero de repeticiones  de la  adquisicion :"
8  18  goto.xy ." D) Escribe el tiempo de retardo entre cada muestra(msec) :"
7 foreground
Begin
 inverse.on
 67 12 goto.xy #entrada&checa
 inverse.off
 numcan :=
 numcan 1 = not numcan 2 = not and
While
 bell
 40 18 goto.xy ." Numero de Canales invalido ... "
 4000 msec.delay
 40 18 goto.xy ."                              "
 66 12 66 12 goto.xy
 spaces
Repeat
inverse.on
10 21  goto.xy ."   Sugerencia: Utiliza el mismo numero de muestras si estas  "
```

```
10 22 goto.xy ."                realizando el mismo experimento.        "
67 14 goto.xy #entrada&checa
muestras :=
inverse.off
10 21 goto.xy ."                                                        "
10 22 goto.xy ."                                                        "
inverse.on
67 16 goto.xy #entrada&checa
repeticiones :=
67 18 goto.xy #entrada&checa
espmsec :=
inverse.off
15 foreground
;

\ RUTINA PRINCIPAL PARA LA ADQUISICION DE DATOS
\ ---------------------------------------
\ Rutina de adquisicion
: rutinadeadquisicion
stack.clear
pidecanalmuestrayrep
screen.clear
0 background
15 foreground
1 varlog :=
BEGIN
 1 sw :=
 stack.clear
 graphics.display
 {def}
 7 foreground
 1  0 goto.xy ." Fecha: " .date
 63 0 goto.xy ." Hora: " .time
 9 foreground
 25 0 goto.xy ." RUTINA DE ADQUISICION DE DATOS"
 15 foreground
 recuadro
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 0 goto.xy ." Checa que esten listos todos los aparatos y conexiones para "
 11 1 goto.xy ."              iniciar la adquisicion... "
 11 2 goto.xy ."          Teclea <ENTER>  cuando asi  sea."
 65 2 goto.xy
 espera.una.clave
 numcan 1 =
 if
  1 numarch :=
  iniciaplantilla1ch
  arriba
  0 vuport.color
  6 axis.color
  2 label.color
  15 color
  sumarygraficar
  muestras real ramp
```

```
 becomes> promedio.data
 repeticiones promedio.data :=
 suma.data promedio.data /
 becomes> promedio.data
 promedio.data
 dup
 tiempo.data swap
 abajo
 0 vuport.color
 6 axis.color
 2 label.color
 15 color
 xy.auto.plot
 pintaetiquetas1
 pintaencabezado2
 menumedicion
 "input
 "drop
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 1 goto.xy ." Se salvara el archivo de promedios (S/N) ? "
 58 1 goto.xy
 DAR.S/N
 if
  83 emit
  grabaarchprom
 else
  78 emit
  drop
 then
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 3 1 goto.xy ." Se salvara el archivo de registros acumulados (S/N) ? "
 58 1 goto.xy
 DAR.S/N
 if
  83 emit
  grabaarchacum
 else
  78 emit
  stack.clear
 then
else
 1 numarch :=
 muestras 2 * muestras :=
 iniciaplantilla2ch
 sumarygraficardos
 muestras real ramp
 becomes> promedio.data
 repeticiones promedio.data :=
 suma.data promedio.data /
 becomes> promedio.data
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 2 goto.xy ." Oprime una tecla para ver los promedios de ambos canales..."
```

```
 75 2 goto.xy
 espera.una.clave
 arribader
 0 vuport.color
 6 axis.color
 2 label.color
 15 color
 vuport.clear
 abajoder
 0 vuport.color
 6 axis.color
 2 label.color
 15 color
 vuport.clear
 tiempo.data
 promedio.data
 sub[ 1 , muestras , 2 ]
 arribader
 xy.auto.plot
 pintaetiquetas1
 pintaencabezado2
 menumedicion
 "input
 "drop
 tiempo.data
 promedio.data
 sub[ 2 , muestras , 2 ]
 abajoder
 xy.auto.plot
 pintaetiquetas1
 pintaencabezado2
 menumedicion
 "input
 "drop
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 2 goto.xy ." Se salvara el archivo de promedios del canal 0 ? "
 60 2 goto.xy
 DAR.S/N
 if
  83 emit
  promedio.data
  sub[ 1 , muestras , 2 ]
  grabaarchprom
 else
  78 emit
 then
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 2 goto.xy ." Se salvara el archivo de promedios del canal 1 ? "
 60 2 goto.xy
 DAR.S/N
 if
  83 emit
  promedio.data
```

```
 sub[ 2 , muestras , 2 ]
 grabaarchprom
else
 78 emit
then
screen.clear
186 205 201 200 188 187 border.chars {BORDER}
11 1 goto.xy ." Se salvara el archivo de registros acumulados del canal 0 ?"
71 1 goto.xy
DAR.S/N
if
 83 emit
11 2 goto.xy ." Se salvara el archivo de registros acumulados del canal 1 ?"
 71 2 goto.xy
 DAR.S/N
 if
   83 emit
   2 numarch :=
   dup
   sub[ 2 , muestras , 2 ]
   swap
   sub[ 1 , muestras , 2 ]
   0 ?escribe.archivo :=
   0 ?escribe.archivo2 :=
   selecciona.archivo.salida
   ?escribe.archivo 1 =
   if
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ."  Grabando  Informacion canal 0 .... "
    tiempo.data grabaarchacumc0
    grabaarchacumc0
   else
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ." Grabacion del Canal 0 Abortado .... "
    3000 msec.delay
    drop
   then
   ?escribe.archivo2 1 =
   if
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ."  Grabando  Informacion canal 1 .... "
    tiempo.data grabaarchacumc1
    grabaarchacumc1
   else
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 2 goto.xy ." Grabacion del Canal 1 Abortado .... "
    3000 msec.delay
    drop
   then
   ?escribe.archivo 1 = ?escribe.archivo2 1 = and
   if
```

```
      repeticiones 1 do
       dup
       sub[ 2 , muestras , 2 ]
       swap
       sub[ 1 , muestras , 2 ]
       grabaarchacumc0
       grabaarchacumc1
      loop
     else
      ?escribe.archivo 1 =
      if
       repeticiones 1 do
         sub[ 1 , muestras , 2 ]
         grabaarchacumc0
       loop
      then
      ?escribe.archivo2 1 =
      if
       repeticiones 1 do
         sub[ 2 , muestras , 2 ]
         grabaarchacumc1
       loop
      then
      stack.clear
     then
    else
     1 numarch :=
     78 emit
     0 ?escribe.archivo :=
     selecciona.archivo.salida
     ?escribe.archivo 1 =
     if
      screen.clear
      186 205 201 200 188 187 border.chars {border}
      10 2 goto.xy ." Grabando Informacion canal 0 .... "
      tiempo.data grabaarchacumc0
      repeticiones 1 + 1 do
       sub[ 1 , muestras , 2 ]
       grabaarchacumc0
      loop
     else
      screen.clear
      186 205 201 200 188 187 border.chars {border}
      10 2 goto.xy ." Grabacion del Canal 0 Abortado .... "
      3000 msec.delay
      stack.clear
     then
    then
   else
    78 emit
    1 numarch :=
   11 2 goto.xy ." Se salvara el archivo de registros acumulados del canal 1 ?"
    71 2 goto.xy
    DAR.S/N
    if
```

```
    83 emit
    0 ?escribe.archivo :=
    selecciona.archivo.salida
    ?escribe.archivo 1 =
    if
     screen.clear
     186 205 201 200 188 187 border.chars {border}
     10 2 goto.xy ."  Grabando  Informacion canal 1 .... "
     tiempo.data grabaarchacumc0
     repeticiones 1 + 1 do
      sub[ 2 , muestras , 2 ]
      grabaarchacumc0
     loop
    else
     screen.clear
     186 205 201 200 188 187 border.chars {border}
     10 2 goto.xy ." Grabacion del Canal 1 Abortado .... "
     3000 msec.delay
     stack.clear
    then
   else
    78 emit
    stack.clear
   then
  then
 stack.clear
 muestras 2 / muestras :=
 then
 screen.clear
 186 205 201 200 188 187 border.chars {BORDER}
 11 1 goto.xy ."      Deseas realizar otra adquisicion (S/N) ? "
 60 1 goto.xy
 DAR.S/N
 if
  83 emit
  1 varlog :=
11 1 goto.xy ." Deseas cambiar los parametros dados (#can,#mues,#rep) (S/N) ? "
  DAR.S/N
  if
   83 emit
   11 2 goto.xy ."        Presiona una tecla para continuar..."
   60 2 goto.xy
   espera.una.clave
   pidecanalmuestrayrep
  else
   78 emit
   11 2 goto.xy ."        Presiona una tecla para continuar..."
   60 2 goto.xy
   espera.una.clave
  then
 else
  78 emit
  0 varlog :=
  11 2 goto.xy ."        Presiona una tecla para continuar..."
  60 2 goto.xy
```

```
    espera.una.clave
   then
   0 varlog =
  UNTIL
  normal.display
  ;

  \                                                    **********************
  \                                                    *   PROGRAM2.DMO     *
  \                                                    **********************

  \*************************************************************************
  \ MODULO  ANALISIS  DE  RESULTADOS
  \*************************************************************************

  \ Presentacion de Analisis de Resultados
  : presentaanalisis
  screen.clear
  presenta
  screen.clear
  1 background
  15 foreground
  186 205 201 200 188 187 border.chars {border}
  7 foreground
  10 1 goto.xy ." U.N.A.M.                    Fac. CIENCIAS"
  10 2 goto.xy ."  Laboratorio de Biofisica del control neuromuscular"
  10 3 goto.xy ."            Rutina de Analisis"
  10 4 goto.xy ."              PRESENTACION"
  5 5  goto.xy ." ____________________________________________________________________"
  15 foreground
  5 7  goto.xy ."  La presente rutina realiza el analisis de datos que  ya han"
  5 8 goto.xy ." sido grabados previamente y que son archivos tipo  PROM o  tipo"
  5 9 goto.xy ." ACU.  "
  5 10 goto.xy ."  Dichos archivos seran  desplegados graficamente y se podran"
  5 11 goto.xy ." realizar las siguientes mediciones:"
  5 12 goto.xy ." a) Areas comprendidas entre dos cursores"
  5 13 goto.xy ." b) Dara los Tiempos y las amplitudes"
  5 14 goto.xy ." c) Podra desplegar graficamente dos archivos y mediante un menu"
  5 15 goto.xy ."  de ofertas "
  5 16 goto.xy ."          1. superponerlos"
  5 17 goto.xy ."          2. restarlos"
  5 18 goto.xy ."          3. sumarlos"
  5 19 goto.xy ."          4. dividir uno entre otro"
  5 20 goto.xy ."          5. filtrarlos"
  7 foreground
  10 22 goto.xy ."            Presione cualquier tecla para continuar..."
  15 foreground
 espera.una.clave
 ;

\ Palabra que pide los archivos de entrada y los grafica
: pidearchivoygrafica
{def}
7 foreground
```

```
 1  0 goto.xy ." Fecha: " .date
 63 0 goto.xy ." Hora: " .time
 15 foreground
 recuadro
 15 foreground
 selecciona.archivo.entrada
 nombre.archivo defer> file.open
 file.contents
 numsub :=  numcom :=
 1 subfile
 file>unnamed.array
 becomes> tiempo.data
 2 subfile
 file>unnamed.array
 numopc 3 = numopc 1 = or
 if
  dup
 then
 tiempo.data swap
 xy.auto.plot
 numopc 3 =
 if
  pintaetiquetas2
 else
  pintaetiquetas1
 then
 pintaencabezado3
 numsub 2 >
 if
  numsub 1 - 1 do
    I 2 + subfile
    file>unnamed.array
    tiempo.data swap
    xy.data.plot
  loop
 then
 numcom 0 = not
 if
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  15 0 goto.xy ." COMENTARIOS  REALIZADOS"
  numcom 1 + 1 do
   I comment> comentario ":="
   5 I goto.xy I . ." > " comentario "type
  loop
else
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 10 0 goto.xy ." NO SE REALIZO NINGUN COMENTARIO"
then
nombre.archivo defer> file.close
1 foreground
50 0 goto.xy ." <Enter> para continuar"
15 foreground
espera.una.clave
```

```
;

\ Palabra que mide las areas comprendida entre dos cursores
: midecareas
   {def}
   10 23 goto.xy ."                                    "
   tiempo.data []size
   swap drop
   muestras :=
   tiempo.data [ muestras ] tiempofinal :=
   coordcurs
   \ Algoritmo para calcular el Area
   \
   coordin muestras *  tiempofinal / coordin :=
   coordfin muestras * tiempofinal / coordfin :=
   array1.data
   integrate.data
   becomes> array2.data
   array2.data [ coordin ]
   array2.data [ coordfin ]
   swap -
   area :=
   area
   37 23 goto.xy ." Area Calculada = "  . ."  volts*mseg"
   recuadro
   66 3 goto.xy
   "input
   "drop
   {DEF}
   1 22 goto.xy ."                              "
   1 23 goto.xy ."                              "
   37 23 goto.xy ."                                    "
   recuadro
;

\ Palabra que mide los tiempos y amplitudes
: midetiempoampl
   {def}
   10 23 goto.xy ."                                    "
   coordcurs
   coordenaday2 coordenaday1 - abs
   coordenadax2 coordenadax1 - abs
   5  23 goto.xy ." Diferencia en X (tiempo) = "  .
   42 23 goto.xy ." Diferencia en Y (Volts) = "  .
   recuadro
   66 3 goto.xy
   "input
   "drop
;



\ Rutina Principal en analisis.
: rutinadeanalisis
0 vuport.color
```

```
6 axis.color
2 label.color
15 color
BEGIN
 stack.clear
 0 swdesp :=
 1 sw :=
 0 switch :=
 normal.display
 screen.clear
 presenta
 1 background
 15 foreground
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 7 foreground
 1  0 goto.xy ." Fecha: " .date
 60 0 goto.xy ." Hora: " .time
 13 2  goto.xy ." U.N.A.M.                              Fac. CIENCIAS "
 13 3  goto.xy ."   Laboratorio de Biofisica del control Neuromuscular"
 13 5  goto.xy ."          RUTINA DE ANALISIS DE RESULTADOS  "
 10 6 goto.xy ." ____________________________________________________________"
 15 foreground
 13 9  goto.xy ."                Menu de Seleccion"
8 11 goto.xy ." 1) Realizar mediciones de areas comprendidas entre 2 cursores"
8 13 goto.xy ." 2) Realizar mediciones de tiempos y amplitudes "
8 15 goto.xy ." 3) Realizar operaciones con dos archivos (+,-,/,suprpns,filt)"
8 17 goto.xy ." 4) Regresar al menu principal"
inverse.on
27 19 goto.xy ."  Escribe una opcion :  "
inverse.off
Begin
 inverse.on
 50 19 goto.xy #entrada&checa
 inverse.off
 numopc :=
 numopc 1 = not numopc 2 = not and numopc 3 = not and numopc 4 = not and
while
 30 21 goto.xy ."       Oprime una opcion valida..."
 3000 msec.delay
 30 21 goto.xy ."                                  "
 50 19 50 19 goto.xy
 spaces
repeat
numopc
case
 1 of
 graphics.display
 1 numcan :=
 {DEF}
 7 foreground
 0  0 goto.xy ." Fecha: " .date
 63 0 goto.xy ." Hora: " .time
 9 foreground
 26 0 goto.xy ." RUTINA DE MEDICION DE AREAS"
```

```
15 foreground
begin
 0 swdesp :=
 general
 0 vuport.color
 6 axis.color
 2 label.color
 15 color
 vuport.clear
 pidearchivoygrafica
 becomes> array1.data
 menumedicion
 cursores
 begin
   midareas
   {def}
   5  23 goto.xy ."                          "
   38 23 goto.xy ."                            "
   10 23 goto.xy ." Calcularas otra area en esta grafica (S/N) ? "
   DAR.S/N not
 until
 10 23 goto.xy ."                                    "
 recuadro
 14 3 goto.xy ."                                      "
 14 3 goto.xy ." Deseas realizar otra medicion de areas (S/N) ? "
 DAR.S/N not
until
endof
2 of
 graphics.display
 1 numcan :=
 {DEF}
 7 foreground
 0  0 goto.xy ." Fecha: " .date
 63 0 goto.xy ." Hora: " .time
 9 foreground
 19 0 goto.xy ." RUTINA DE MEDICION DE TIEMPOS Y AMPLITUDES"
 15 foreground
 begin
  0 swdesp :=
  general
  0 vuport.color
  6 axis.color
  2 label.color
  15 color
  vuport.clear
  pidearchivoygrafica
  menumedicion
  cursores
  begin
    midetiempoampl
    {def}
    5  23 goto.xy ."                          "
    38 23 goto.xy ."                            "
    10 23 goto.xy ." Mediras otro tiempo/ampl en esta grafica (S/N) ? "
```

```
    DAR.S/N not
   until
   10 23 goto.xy ."                                    "
   recuadro
   14 3 goto.xy ."                                     "
   12 3 goto.xy ." Se realizara otra medicion de tiemp/ampl (S/N) ? "
   DAR.S/N not
  until
 endof
 3 of
  graphics.display
  {DEF}
  7 foreground
  0  0 goto.xy ." Fecha: " .date
  63 0 goto.xy ." Hora: " .time
  9 foreground
  21 0 goto.xy ." RUTINA DE OPERACIONES CON DOS ARCHIVOS"
  15 foreground
  begin
   0 swdesp :=
   supizq 0 vuport.color 6 axis.color 2 label.color 15 color vuport.clear
   supder 0 vuport.color 6 axis.color 2 label.color 15 color vuport.clear
   abajo  0 vuport.color 6 axis.color 2 label.color 15 color vuport.clear
  = supizq pidearchivoygrafica
   becomes> array1.data
   1 swdesp :=
   supder pidearchivoygrafica
   becomes> array2.data
   array1.data []size
   swap drop
   array2.data []size
   swap drop
   =
   if
     menuofertas
   else
     recuadro
     15 foreground
     screen.clear
     186 205 201 200 188 187 border.chars {BORDER}
     20 1 goto.xy ." Numero de muestras o tiempos incompatibles..."
     20 2 goto.xy ."       revisa tus archivos."
   then
   14 3 goto.xy ."                                     "
   12 3 goto.xy ." Deseas salir de la rutina de operaciones (S/N) ? "
   DAR.S/N
  until
 endof
 endcase
 numopc 4 =
UNTIL
2 numopc :=
normal.display
;
```

```
\ ********************************************************************
\ MODULO  GRAFICACION DE RESULTADOS
\ ********************************************************************

\ Presenta Graficacion
: presentagraficacion
screen.clear
presenta
1 background
15 foreground
186 205 201 200 188 187 border.chars {border}
7 foreground
10 1 goto.xy ." U.N.A.M.                    Fac. CIENCIAS"
10 2 goto.xy ."  Laboratorio de Biofisica del control neuromuscular"
10 3 goto.xy ."          Rutina de Graficacion"
10 4 goto.xy ."            PRESENTACION"
5 5  goto.xy ." ____________________________________________________________"
15 foreground
5 7 goto.xy ."  La presente rutina realiza la graficacion de datos que ya han"
5 8 goto.xy ." sido grabados previamente y que son archivos tipo  PROM o  tipo"
5 9 goto.xy ." ACU.  "
5 10 goto.xy ."  Permitira seleccionar entre dichos archivos, para hacer arre-"
5 11 goto.xy ." glos de presentacion, como son el escribir  leyendas y  titulos"
5 12 goto.xy ." en los ejes, al pie de los registros, sobre las  curvas  de las"
5 13 goto.xy ." graficas."
5 14 goto.xy ."  Ofrece un menu para ajustar los colores y las dimensiones  de"
5 15 goto.xy ." la impresion, empleando una impresora laser."
7 foreground
10 22 goto.xy ."          Presione cualquier tecla para continuar..."
15 foreground
espera.una.clave
;


\ Mensaje de opcion incorrecta
: opcioninc
   presenta
   inverse.on
   60 21 goto.xy ." Opcion invalida."
   inverse.off
   3000 msec.delay
   60 21 goto.xy ."            "
   ventana2
;

\
\ Rutina de graficacion
: rutinadegraficacion
stack.clear
" 'Ninguno...'" nombre.archivo ":="
" 'Ninguno...'" nombre.archivo1 ":="
" 'Ninguno...'" nombre.archivo2 ":="
" 'Ninguno...'" nombre.archivo3 ":="
0 varlog :=
```

```
BEGIN
 stack.clear
 0 sw :=
 0 switch :=
 normal.display
 screen.clear
 presenta
 1 background
 15 foreground
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 7 foreground
 1  0 goto.xy ." Fecha: " .date
 60 0 goto.xy ." Hora: " .time
 13 2  goto.xy ." U.N.A.M.                              Fac. CIENCIAS "
 13 3  goto.xy ."    Laboratorio de Biofisica del control Neuromuscular"
 13 4  goto.xy ."              RUTINA DE GRAFICACION"
 10 5 goto.xy ." ____________________________________________________________"
 15 foreground
 13 7  goto.xy ."                 Menu de Seleccion"
 12 9  goto.xy ." 1) Seleccionar el archivo "
 12 10 goto.xy ." 2) Ajuste de colores y dimensiones de la impresion "
 12 11 goto.xy ." 3) Poner titulos,leyendas y mandar a impresion un archivo"
 12 12 goto.xy ." 4) Realizar operaciones con 2 archivos y mandar a impresion"
 12 13 goto.xy ." 5) Regresar al menu principal "
 7 background
 0 foreground
 29 22 goto.xy ."  Archivos selec: 1) " nombre.archivo2 "type
 ."  2) " nombre.archivo3 "type
 varlog 0 =
 if
  0 22 goto.xy ."  Colores no seleccionados..."
 else
  0 22 goto.xy ."  Colores ya seleccionados..."
 then
 1 background
 15 foreground
 inverse.on
 27 15 goto.xy ."  Escribe una opcion :  "
 inverse.off
 Begin
  inverse.on
  50 15 goto.xy #entrada&checa
  inverse.off
  numopc :=
  numopc 1 = not numopc 2 = not and numopc 3 = not and numopc 4 = not and
  numopc 5 = not and
 while
  30 17 goto.xy ."       Oprime una opcion valida..."
  3000 msec.delay
  30 17 goto.xy ."                                  "
  50 15 50 15 goto.xy
  spaces
 repeat
 numopc
```

```
case
 1 of
  7 foreground
  1  0 goto.xy ." Fecha: " .date
  60 0 goto.xy ." Hora: " .time
  15 foreground
  ventana1
  1 background
  15 foreground
  selecciona.archivo.entrada
  nombre.archivo nombre.archivo2 ":="
  screen.clear
  presenta
 endof
 2 of
  7 foreground
  1  0 goto.xy ." Fecha: " .date
  60 0 goto.xy ." Hora: " .time
  15 foreground
  ventana2
  1 background
  15 foreground
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  7  background
  0 foreground
  15 0 goto.xy ."   AJUSTE DE COLORES   "
  1 background
  15 foreground
  0  1 goto.xy ." 0) Negro "
  1 foreground
  0 background
  0  2 goto.xy ." 1) Azul M."
  2 foreground
  1 background
  0  3 goto.xy ." 2) Verde"
  3 foreground
  0  4 goto.xy ." 3) Azul C."
  4 foreground
  0  5 goto.xy ." 4) Rojo"
  5 foreground
  12 1 goto.xy ." 5) Rosa"
  6 foreground
  12 2 goto.xy ." 6) Oro"
  7 foreground
  12 3 goto.xy ." 7) Blanco"
  8 foreground
  12 4 goto.xy ." 8) Negro Int"
  9 foreground
  12 5 goto.xy ." 9) Azul Int"
  10 foreground
  26 1 goto.xy ." 10) Verde Int."
  11 foreground
  26 2 goto.xy ." 11) Azul C. Int"
  12 foreground
```

```
26 3 goto.xy ." 12) Rojo Int"
13 foreground
26 4 goto.xy ." 13) Rosa Int"
14 foreground
26 5 goto.xy ." 14) Amarillo"
15 foreground
42 1 goto.xy ." 15) Blanco Int"
7 foreground
60 0 goto.xy ." Fondo  : "
60 1 goto.xy ." Axisas : "
60 2 goto.xy ." Etiq.  : "
60 3 goto.xy ." Titulo : "
60 4 goto.xy ." LinGraf: "
60 5 goto.xy ." Coment : "
15 foreground
Begin
 inverse.on
 70 0 goto.xy #entrada&checa
 inverse.off
 colorfondo :=
 colorfondo 0 < colorfondo 15 > or
while
 opcioninc
 70 4 70 0 goto.xy
 spaces
repeat
Begin
 inverse.on
 70 1 goto.xy #entrada&checa
 inverse.off
 coloraxisas :=
 coloraxisas 0 < coloraxisas 15 > or
while
 opcioninc
 70 4 70 1 goto.xy
 spaces
repeat
Begin
 inverse.on
 70 2 goto.xy #entrada&checa
 inverse.off
 coloretiquetas :=
 coloretiquetas 0 < coloretiquetas 15 > or
while
 opcioninc
 70 4 70 2 goto.xy
 spaces
repeat
Begin
 inverse.on
 70 3 goto.xy #entrada&checa
 inverse.off
 colortitulos :=
 colortitulos 0 <  colortitulos 15 > o
while
```

```
 opcioninc
 70 4 70 3 goto.xy
 spaces
repeat
Begin
 inverse.on
 70 4 goto.xy #entrada&checa
 inverse.off
 colorlinea :=
 colorlinea 0 <  colorlinea 15 > or
while
 opcioninc
 70 4 70 4 goto.xy
 spaces
repeat
Begin
 inverse.on
 70 5 goto.xy #entrada&checa
 inverse.off
 colorcomentarios :=
 colorcomentarios 0 < colorcomentarios 15 > or
while
 opcioninc
 70 4 70 5 goto.xy
 spaces
repeat
presenta
inverse.on
57 21 goto.xy ." <Ent> para cont..."
inverse.off
espera.una.clave
ventana2
screen.clear
presenta
57 21 goto.xy ."                    "
1 varlog :=
7 foreground
1  0 goto.xy ." Fecha: " .date
60 0 goto.xy ." Hora: " .time
15 foreground
ventana2
1 background
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {border}
7  background
0 foreground
26 0 goto.xy ."   TIPO DE IMPRESION   "
1 background
15 foreground
Begin
 5 2 goto.xy ." Cuantas divisiones en la linea horizontal X (1-10) ? "
 #entrada&checa
 divhor :=
 divhor 1 < divhor 10 > or
```

```
while
 40 4 goto.xy ." Opcion invalida (de 1 a 10) ..."
 3000 msec.delay
 40 4 goto.xy ."                              "
 4 58 2 goto.xy
 spaces
repeat
Begin
 5 3 goto.xy ." Cuantas divisiones en la linea vertical Y (1-10) ? "
 #entrada&checa
 divver :=
 divver 1 <  divver 10 > or
while
 40 4 goto.xy ." Opcion invalida (de 1 a 10) ..."
 3000 msec.delay
 40 4 goto.xy ."                              "
 4 56 3 goto.xy
 spaces
repeat
5 4 goto.xy ." Quieres que se imprima la gradilla (reja de sep) (S/N)? "
DAR.S/N
if
 horizontal grid.on
 vertical grid.on
else
 horizontal grid.off
 vertical grid.off
then
divhor divver axis.divisions
screen.clear
presenta
endof
3 of
 " 'Ninguno...'"  nombre.archivo2 "=
 if
    7 foreground
    1  0 goto.xy ." Fecha: " .date
    60 0 goto.xy ." Hora: " .time
    15 foreground
    ventana1
    1 background
    15 foreground
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    20 1 goto.xy ." No ha sido seleccionado ningun archivo ... "
    20 2 goto.xy ." escoge la opcion <1> antes de escoger <3>."
    5000 msec.delay
    screen.clear
    presenta
 else
    graphics.display
    general
    colorfondo vuport.color
    coloretiquetas label.color
    colorlinea color
```

```
coloraxisas axis.color
recuadro
nombre.archivo2 defer> file.open
file.contents
numsub := numcom :=
1 subfile
file>unnamed.array
becomes> tiempo.data
2 subfile
file>unnamed.array
tiempo.data swap
xy.auto.plot
numsub 2 >
if
 numsub 1 - 1 do
  1 2 + subfile
  file>unnamed.array
  tiempo.data swap
  xy.data.plot
 loop
then
numcom 0 = not
if
 15 foreground
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 15 0 goto.xy ." COMENTARIOS  REALIZADOS"
 colorcomentarios foreground
 numcom 1 + 1 do
  I comment> comentario ":=
  5 I goto.xy I . ." > " comentario "type
 loop
else
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 10 0 goto.xy ." NO SE REALIZO NINGUN COMENTARIO"
then
nombre.archivo2 defer> file.close
1 foreground
50 0 goto.xy ." <Enter> para continuar"
15 foreground
espera.una.clave
recuadro
screen.clear
186 205 201 200 188 187 border.chars {border}
5 1 goto.xy ." Deseas recortar la grafica (S/N) ? "
DAR.S/N
if
 83 emit
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 menumedicion
 cursores
10 3 goto.xy ."  Recortando coordenadas en X para realizar un ZOOM..."
 {def}
```

```
  coordcurs
  tiempo.data []size
  swap drop
  muestras :=
  tiempo.data [ muestras ] tiempofinal :=
  coordin muestras * tiempofinal / coordin :=
  coordfin muestras * tiempofinal / coordfin :=
  nombre.archivo2 defer> file.open
  file.contents
  numsub :=  numcom :=
  1 subfile
  file>unnamed.array
  sub[ coordin , coordfin , 1 ]
  becomes> tiempo.data
  2 subfile
  file>unnamed.array
  sub[ coordin , coordfin , 1 ]
  tiempo.data swap
  xy.auto.plot
  numsub 2 >
  if
   numsub 1 - 1 do
    1 2 + subfile
    file>unnamed.array
    sub[ coordin , coordfin , 1 ]
    tiempo.data swap
    xy.data.plot
   loop
  then
  nombre.archivo2 defer> file.close
 else
  78 emit
  7 foreground
  45 3 goto.xy ." <Enter> para continuar"
  15 foreground
  espera.una.clave
 then
 recuadro
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 1 0 goto.xy ." TITULOS.  Instr: "
 7 foreground
19 0 goto.xy ." Manejando las <flechas> coloque el cursor en el lugar en"
 10 1 goto.xy ." donde desea poner el titulo o leyenda o de <N> si "
 60 1 goto.xy ." desea terminar."
 colortitulos foreground
 begin
  graphics.readout
  75 1 3 goto.xy
  spaces
  1 3 goto.xy " T1 > " "entrada&checa
  titulo ":=
  titulo " n" "= titulo " N" "= or not
  if
   colortitulos color
```

```
    titulo label
   then
   titulo " n" "= titulo " N" "= or
  until
  15 foreground
1 3 goto.xy "                                                    "
1 3 goto.xy ." Desea mandar a impresion tambien los comentarios (S/N)? "
  DAR.S/N
  if
   83 emit
   nombre.archivo2 defer> file.open
   numcom 0 = not
   if
    15 foreground
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    27 0 goto.xy ." C O M E N T A R I O S "
    colorcomentarios foreground
    numcom 1 + 1 do
     1 comment> comentario ":="
     5 1 goto.xy 1 . ." > " comentario "type
    loop
   else
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    10 0 goto.xy ." NO SE REALIZO NINGUN COMENTARIO"
   then
   nombre.archivo2 defer> file.close
  else
   78 emit
   recuadro2
   screen.clear
 then
 {def}
 15 foreground
 10 23 goto.xy ." Cuantas copias necesitas : " #entrada&checa
 copias :=
 10 23 goto.xy ."                                    "
 15 foreground
 2  23 goto.xy ." Que formato deseas: 1)Media Pag   2)Un Cuarto Pag  "
 9 foreground
 55 23 goto.xy ." Opcion : "
 Begin
   64 23 goto.xy #entrada&checa
   numopc :=
   numopc 1 = not numopc 2 = not and
 while
   64 23 goto.xy ." Opc. invalida"
   2000 msec.delay
   64 23 goto.xy ."              "
 repeat
 15 foreground
 numopc 1 =
 if
   1200 1000 bd.scale
```

```
    100 100 bd.res
  else
    2400 1200 bd.scale
    200 300 bd.res
  then
  3 numopc :=
  2  23 goto.xy ."                                    "
  55 23 goto.xy ."          "
  64 23 goto.xy ."             "
  3  23 goto.xy ." Checa que este lista la impresora y da <enter>"
  52 23 goto.xy ." para imprimir..."
  espera.una.clave
  3  23 goto.xy ."                                    "
  52 23 goto.xy ."             "
  copias 1 + 1 do
    0 foreground
    80 23 goto.xy
    BD.SCREEN
    15 foreground
    30 23 goto.xy ." Reporte No. " 1 . ."  mandado a impresion"
    3000 msec.delay
    30 23 goto.xy ."                                  "
    30 23 goto.xy ." Espera que salga la impresion y da <enter>..."
    espera.una.clave
    30 23 goto.xy ."                                  "
  loop
  30 23 goto.xy ." Oprime una tecla para continuar ... "
  espera.una.clave
 then
endof
4 of
  1 switch :=
  7 foreground
  1  0 goto.xy ." Fecha: " .date
  60 0 goto.xy ." Hora: " .time
  15 foreground
  ventana1
  1 background
  15 foreground
  selecciona.archivo.entrada
  nombre.archivo nombre.archivo2 ":="
  selecciona.archivo.entrada
  nombre.archivo nombre.archivo3 ":="
  ventana1
  30 4 goto.xy ."  Presiona una tecla para continuar... "
  espera.una.clave
  screen.clear
  graphics.display
  supizq
  colorfondo vuport.color
  coloretiquetas label.color
  colorlinea color
  coloraxisas axis.color
  supder
  colorfondo vuport.color
```

```
coloretiquetas label.color
colorlinea color
coloraxisas axis.color
abajo
colorfondo vuport.color
coloretiquetas label.color
colorlinea color
coloraxisas axis.color
recuadro
supizq vuport.clear
nombre.archivo2 defer> file.open
file.contents
numsub := numcom :=
1 subfile
file>unnamed.array
becomes> tiempo.data
2 subfile
file>unnamed.array
becomes> array1.data
tiempo.data array1.data
colorlinea color
xy.auto.plot
colortitulos color
normal.coords
.450 .930 position nombre.archivo2 label
world.coords
colorlinea color
nombre.archivo2 defer> file.close
supder vuport.clear
nombre.archivo3 defer> file.open
file.contents
numsub := numcom :=
1 subfile
file>unnamed.array
becomes> tiempo.data
2 subfile
file>unnamed.array
becomes> array2.data
tiempo.data array2.data
colorlinea color
xy.auto.plot
colortitulos color
normal.coords
.450 .930 position nombre.archivo3 label
world.coords
colorlinea color
nombre.archivo3 defer> file.close
begin
  array1.data []size
  swap drop
  array2.data []size
  swap drop
  =
  if
    menuofertas
```

```
      else
        recuadro
        15 foreground
        screen.clear
        186 205 201 200 188 187 border.chars {BORDER}
        20 1 goto.xy ." Numero de muestras o tiempos incompatibles..."
        20 2 goto.xy ."       revisa tus archivos."
      then
      15 foreground
      14 3 goto.xy ."                                    "
      12 3 goto.xy ." Deseas salir de operaciones/graficadas (S/N) ? "
      DAR.S/N
    until
  endof
 endcase
 numopc 5 =
UNTIL
10 10 axis.divisions
vertical grid.on
horizontal grid.on
3 numopc :=
normal.display
;

\ ********************************************************************************
\ MODULO   TRANSFERENCIA  DE  DATOS
\ ********************************************************************************

\ Palabra que se encarga de transferir datos que se encuentran en archivos
\ creados en ASYST con extension .prm o .acu a archivos que pueden ser maneja-
\ dos por Lotus o Excel con extension .wks
: transferencia
"                          " nombre.archivo  ":=
"                          " nombre.archivo2 ":=
"                          " nombre.archivo3 ":=
0 numsub :=
0 numcom :=
stack.clear
screen.clear
presenta
1 background
15 foreground
begin
  screen.clear
  186 205 201 200 188 187 border.chars {border}
  7 foreground
  1  0 goto.xy ." Fecha: " .date
  60 0 goto.xy ." Hora: " .time
  12 2 goto.xy  ." U.N.A.M.                    Fac. CIENCIAS"
  12 3 goto.xy  ."  Laboratorio de Biofisica del control neuromuscular"
  12 4  goto.xy ." ____________________________________________________________"
  15 foreground
  12 7 goto.xy  ."          TRANSFERENCIA  DE  DATOS"
  7 foreground
8 10 goto.xy ." Esta rutina transfiere datos que se encuentran en archivos tipo"
```

```
8 11 goto.xy ." .prm  o tipo .acu  creados en  ASYST a archivos  tipo  .wks que"
8 12 goto.xy ." pueden ser manejados tanto por Lotus123 como por Excel. "
    15 foreground
    ventana1
    1 background
    15 foreground
    186 205 201 200 188 187 border.chars {border}
    12 1 goto.xy ." Deseas realizar una transferencia (S/N) ? "
    DAR.S/N
while
    stack.clear
    ventana1
    1 background
    15 foreground
    selecciona.archivo.entrada
    nombre.archivo nombre.archivo2 ":="
    ventana3
    muestraarchivoswks
    ventana1
    screen.clear
    186 205 201 200 188 187 border.chars {border}
    7 foreground
    10 0 goto.xy ."     Dar: Nombre (8 car) + Extension (.WKS) "
    15 foreground
    begin
     8 2 goto.xy ." Escribe el nombre del archivo : "  "entrada&checa
     nombre.archivo ":="
     nombre.archivo nombre.archivo3 ":="
     nombre.archivo defer> file.sizes
     0 = 0 = and not
    while
     8 3 goto.xy ." Archivo ya existente, no podemos regrabarlo ... "
     3000 msec.delay
     8 2 goto.xy ."                                                "
     8 3 goto.xy ."                                "
    repeat
    8 3 goto.xy ." El Archivo no existe. Creamos uno nuevo (S/N) ? "
    DAR.S/N
    if
     screen.clear
     186 205 201 200 188 187 border.chars {border}
     8 2 goto.xy ." Espera un momento se esta realizando la transferencia..."
     nombre.archivo2 defer> file.open
     file.contents
     numsub := numcom :=
     numsub subfile
     file>unnamed.array
     numsub  1 do
       numsub I - subfile
       file>unnamed.array
     loop
     nombre.archivo2 defer> file.close
     nombre.archivo3 defer> 123file.create
     nombre.archivo3 123file.open
     3 1 123write.down
```

```
      array>123file
      numsub 1 do
        3 1 1 + 123write.down
        array>123file
      loop
      123file.close
      2000 msec.delay
     else
      78 emit
     then
     presenta
repeat
screen.clear
presenta
;


\ ********************************************************************************
\ MODULO      SISTEMA
\ ********************************************************************************

\ Checa que no haya archivos abiertos para poder empezar el sistema
: Cierra.Archivos
 ?file.open
 if
  file.close
 then
;

\ Palabra que se encarga de pedir el directorio de trabajo
: Pidedirectorio
 15 foreground
 screen.clear
 186 205 201 200 188 187 border.chars {Border}
 7 foreground
 8 0 goto.xy ." Da el nombre del directorio de trabajo, incluyendo la unidad : "
 15 foreground
 20 2 goto.xy ." DIRECTORIO : " "Entrada&Checa
 defer> chdir
 true
onerr:
 ?error# error.message
 3000 msec.delay
 false
;

\ Palabra de Presentacion del sistema
: presentacion
screen.clear
presenta
1 background
15 foreground
screen.clear
186 205 201 200 188 187 border.chars {border}
7 foreground
```

```
1 0 goto.xy ." Fecha: " .date
60 0 goto.xy ." Hora: " .time
15 foreground
20 4 goto.xy ." UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO"
20 5 goto.xy ."       FACULTAD DE CIENCIAS"
13 6 goto.xy ." ********************************************************"
14 9 goto.xy ." Laboratorio de Biofisica del control Neuromuscular"
inverse.on
8 12 goto.xy ."  Sistema para Registro Analisis y Graficacion de Experimentos "
8 13 goto.xy ."          Electrofisiologicos en Medula Espinal          "
inverse.off
20 16 goto.xy ."    Dra.  HORTENSIA GONZALEZ GOMEZ"
20 22 goto.xy ."              Presione una tecla para continuar..."
espera.una.clave
ventana 1
begin
  pidedirectorio
until
screen.clear
presenta
;

\ Menu principal de opciones en donde se pregunta que rutina se
\ desea operar
: menuprincipal
Begin
 cierra.archivos
 presenta
 1 background
 15 foreground
 screen.clear
 186 205 201 200 188 187 border.chars {border}
 7 foreground
 1  0 goto.xy ." Fecha: " .date
 60 0 goto.xy ." Hora: " .time
 12 2 goto.xy  ." U.N.A.M.                              Fac. CIENCIAS"
 12 3 goto.xy  ."  Laboratorio de Biofisica del control neuromuscular"
 12 4  goto.xy ." ____________________________________________________________"
 15 foreground
 20 6 goto.xy  ."    M E N U     P R I N C I P A L"
 20 7 goto.xy  ." ============================================"
 20 9  goto.xy ."  1.- Rutina de Adquisicion  de  Datos "
 20 11 goto.xy ."  2.- Rutina de Analisis de Resultados "
 20 13 goto.xy ."  3.- Rutinas      de      Graficacion "
 20 15 goto.xy ."  4.- Transferir datos de Asyst a Lotus"
 20 17 goto.xy ."  5.- Salir  del  Sistema             "
 inverse.on
 27 19 goto.xy ."  Escribe una opcion :  "
 inverse.off
 Begin
  inverse.on
  50 19 goto.xy #entrada&checa
  inverse.off
  numopc :=
  numopc 1 = not numopc 2 = not and numopc 3 = not and numopc 4 = not and
```

```
    numopc 5 = not and
   while
    30 22 goto.xy ."       Oprime una opcion valida..."
    3000 msec.delay
    30 22 goto.xy ."                                  "
    50 19 50 19 goto.xy
    spaces
   repeat
   numopc
   case
    1 of
     presentaadquisicion
     rutinadeadquisicion
    endof
    2 of
     presentaanalisis
     rutinadeanalisis
    endof
    3 of
     presentagraficacion
     rutinadegraficacion
    endof
    4 of
     transferencia
    endof
    5 of
     {def}
     1 background
     screen.clear
     186 205 201 200 188 187 border.chars {border}
     7 foreground
     2  2 goto.xy ." Fecha: " .date
     61 2 goto.xy ." Hora: " .time
     15 foreground
     25 12 goto.xy ."  TERMINASTE DE USAR EL SISTEMA"
     25 14 goto.xy ."      QUE PASES BUEN DIA..."
     40 22 goto.xy ."  Presiona una tecla para finalizar..."
     espera.una.clave
    endof
   endcase
   numopc 5 =
until
directorio
defer> chdir
0 background
15 foreground
screen.clear
BYE
;


\ Palabras usadas para crear el sistema de corrido

: sistema
```

```
 "dir directorio ":=
 normal.display
 presentacion
 menuprincipal
;

load.overlay runtime.aov
\ Define un banner para mandar un mensaje de presentacion
BANNER: primer.banner
 normal.display
 screen.clear
 cr cr cr cr cr
 inverse.on
 ."              COPIA PROTEGIDA DEL SISTEMA DE ADQUISICION          "
 ."                   ANALISIS Y GRAFICACION DE DATOS.          "
 ."          "
 inverse.off
 cr cr cr cr cr cr
 ."                        Programas elaborados por :          "
 ."                             Alfredo Callejas Chavero   "
 ."                                                        "
 cr cr cr
;BANNER

install sistema in turnkey
make.turnkey sistema
```

BIBLIOGRAFÍA

Aho Alfred V., Sethi Ravi, Ullman Jeffrey D.
Compiladores, "Principios Técnicas y Herramientas"
Addison-Wesley Iberoamericana, México 1990

Awad Elias M.
Procesamiento Automático de Datos, "Principios y Procedimientos"
Escuela de Negocios para Graduados, Universidad de Paul Chicago, Illinois
Editorial Diana, México Sexta Edición 1982

Beck Leland L.
System Software, "An Introduction to Systems Programming"
San Diego State University, Segunda Edición
Addison-Wesley-Publishing Company, USA 1990

Dempster John
Computer Analysis of Electrophysiological Signals
Departament of Physiology and Pharmacology, University of Strathclyde Glasgow
San Diego CA., USA 1993
Edition Published by Academic Press Inc.

Fraser P. J.
Microcomputers in Phisiology, "A Practical Aproach"
Departament of Zoology, University of Aberdeen, Aberdeen AB9 2TN, UK
Irl Press, Oxford·Washington DC 1988

Gorsline George W.
Computer Organization: Hardware/Software
Prentice Hall, Segunda Edición

Katzan Harry Jr. (Pratt Institute)
Introduction to Computers and Data Processing
D. Van Nostrand Company, USA 1979

Keithley Data Acquisition Division
ASYST Module 1, Tutorial (Systems, Graphics, Statistics)
Keithley Instruments Inc., 3a Impresión, USA 1992

Keithley Data Acquisition Division
ASYST Module 1, Glossaries (Systems, Graphics, Statistics)
Keithley Instruments Inc. 3a Impresión, USA 1992

Keithley Data Acquisition Division
ASYST Module 3, Data Acquisition
Keithley Instruments Inc., 3a Impresión, USA 1992

Keithley Data Acquisition Division
ASYST "Up and Running"
Keithley Instruments Inc., 3a Impresión, USA 1992

Keithley Data Acquisition Division
ASYST 4.0, Your Guide to ASYST's (New Feature, Enhancements)
Keithley Instruments Inc., 3a Impresión, USA 1992

Keithley Data Acquisition Division
ASYST "Run-Time System"
Keithley Instruments Inc., 3a Impresión, USA 1992

Keithley MetraByte/Asyst/DAC
External DAS Drivers, Keithley MetraByte DAS-40
Keithley Instruments Inc., 1a Impresión, USA 1991

Morgan L. Christopher, Waite Mitchell
Introducción al Microprocesador 8086/8088 (16 Bit)
Mc Graw-Hill, México 1987

Pappas Chris H., Murray William H. III
Manual del Microprocesador 80386
Mc Graw-Hill, México 1989, 1a. Edición

Pressman Roger S.
Ingenieria de Software, "Un Enfoque Práctico"
Mc Graw Hill, Segunda Edición, España 1988

Pyster Arthur B.
Compiler Design and Construction
Van Nostrand Reinhold Company, Segunda Edición
New York, USA 1988

Rainer Bartel
Excel para todos
Computec, Marcombo México 1995

Schildt Herbert
C "Manual de Referencia"
Mc Graw-Hill, España 1989

Softtek, Educación y Tecnología
Técnicas de Ingeniería de Software aplicadas a Programación
México 1996

Stallings William
Computer Organization and Architecture: Principles of Structure and Function
Macmillan Publishing Company, Segunda Edición

Tanenbaum A. M.
Organización de computadoras, "Un enfoque estructurado"
Prentice Hall 1991, Segunda Edición