UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CONTROLADOR DE PERIFERICOS GRAFICOS

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A

MIGUEL CHIN AUYON

DIRECTOR: ING. JUAN BOSCO MARTINEZ

CIUDAD UNIVERSITARIA, D.F., JUNIO 1985

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Tabla de Contenidos

Capitulo 1

Introducción.

En el presente trabajo, se describe el desarrollo e implementación de un sistema para el empleo de un osciloscopio de memoria y de un graficador incremental X-Y, como dispositivos de salida gráfica de alta calidad para equipo de cómputo.

El trabajo se desarrolló en el Laboratorio de Automatización del Instituto de Ingeniería, en él se contaba con un osciloscopio de memoria y con un graficador incremental X-Y para acoplamiento a computadoras. Estos dispositivos formaban parte del equipo periférico de una minicomputadora modelo NOVA de la compañía DATA GENERAL. Dicha minicomputadora entró en desuso y fue dada de baja, sin embargo, tanto el graficador como el osciloscopio constituyen excelentes periféricos gráficos y aún se encuentran en condiciones satisfactorias de trabajo.

También dentro del Laboratorio se cuenta con varias microcomputadoras APPLE II que ofrecen capacidad gráfica media (280 x 196 pixeles) y la salida es básicamente al monitor, con serios inconvenientes y dificultades para producir copia permanente en papel de la gráfica. Estas microcomputadoras sin embargo, dan amplias posibilidades de expansión de la circuitería.

Se detectó la necesidad de proporcionar a los usuarios de dichas microcomputadoras, de una salida gráfica de mejor calidad y con mayor facilidad de manejo con respecto a lo que ofrecía la máquina. Fue entonces, cuando se llegó a la conclusión de que se podían satisfacer estas necesidades y a la vez darle uso de nuevo tanto al osciloscopio de memoria como al graficador incremental, si se contruían las interfaces adecuadas tanto de programación como de circuitería.

El sistema que se elaboró permite accesar al osciloscopio de memoria y al graficador incremental como dispositivos de salida gráfica de alta calidad y alta resolución, ofreciendo extensiones gráficas al lenguaje Pascal que permiten al programador trabajar en coordenadas virtuales sin las preocupaciones normales, como escalas, traslaciones y rangos inválidos. El paquete básico permite seleccionar al dispositivo de salida, trazar líneas y escribir caracteres de diferentes tamaños y orientaciones.

Otra parte del sistema permite, con mucha facilidad, el trazado de gráficas de datos o fórmulas (en dos variables) con encabezados, leyendas, etiquetas, etc., en varias opciones. Finalmente existe un programa que puede manipular (rotar, trasladar, escalar, proyectar en perspectiva, etc.) objetos tridimensionales cuya definición reside en archivos de disco, permitiendo la generación de estos objetos por medio de programas externos.

Capitulo 2

Descripción de los periféricos gráficos.

2-1 El osciloscopio de memoria.

El osciloscopio de memoria es un osciloscopio con pantalla de memoria capacitiva, marca TEKTRONIX, modelo 611 destinado a despliegues gráficos por computadora. Tiene tres modos de operación:

a) ***WRITE-THRU*** (escribe a través). En este modo se comporta como un osciloscopio normal sin almacenamiento de las señales de voltaje que direccionan el haz de electrones hacia la pantalla.

b) ***NON-STORE*** (sin almacenamiento). Este modo de operación permite ver solamente el haz de electrones instántaneo, reduciendo la luminosidad de la memoria capacitiva que registra la posición del haz en todo momento.

c) **Normal.** Es este el modo de operación en el que se permite ver la pantalla de memoria capacitiva a luminosidad plena, reduciéndose ésta a un 10% después de haber transcurrido 90 segundos de inactividad del disparador del haz de electrones o señal del eje Z.

Tiene un panel frontal mostrado en la figura 2.1 con:

a) Control de encendido.

b) Regulación de la intensidad del haz de electrones.

c) Generación de patrones de prueba.

d) Activación del modo de borrado.

e) Reactivación del modo de protección en baja luminosidad, al modo normal de observación.

En la parte trasera (ver figura 2.2), tiene tres conectores tipo BNC hembra, destinados respectivamente al canal del eje X, al canal Y y al habilitador del haz o eje Z. Cuenta además con un conector tipo DB25 hembra a través del cual es posible suministrar las señales X, Y y Z, activar cualquiera de los modos de operación del osciloscopio, activar el estado de borrado y detectar el intervalo durante el cual el osciloscopio no se encuentra disponible. Cabe mencionar que las entradas X, Y y Z son entradas diferenciales (sin tierra común).

FRENTE DEL OSCILOSCOPIO

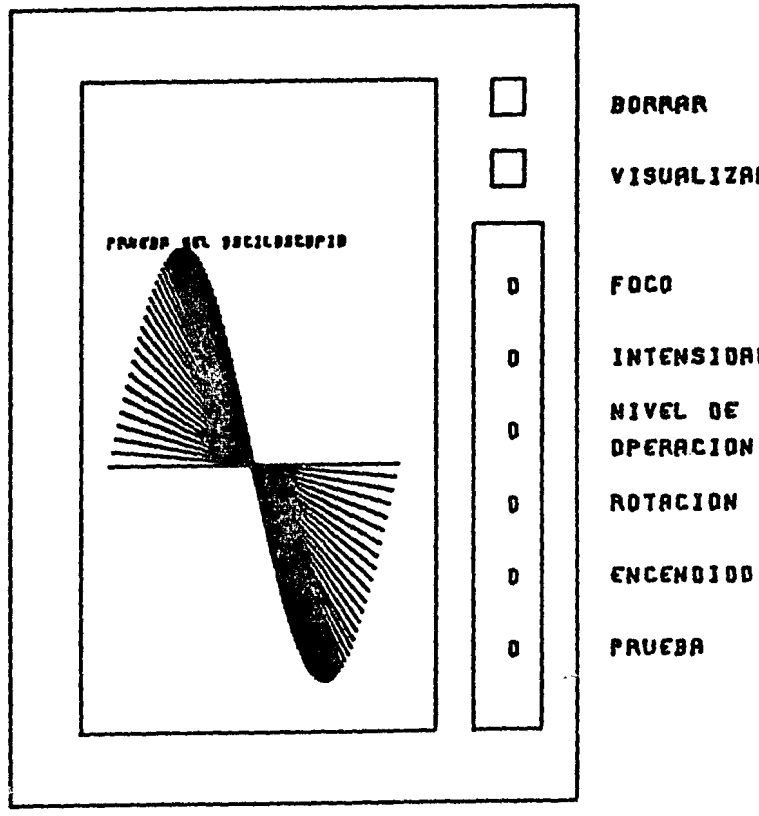
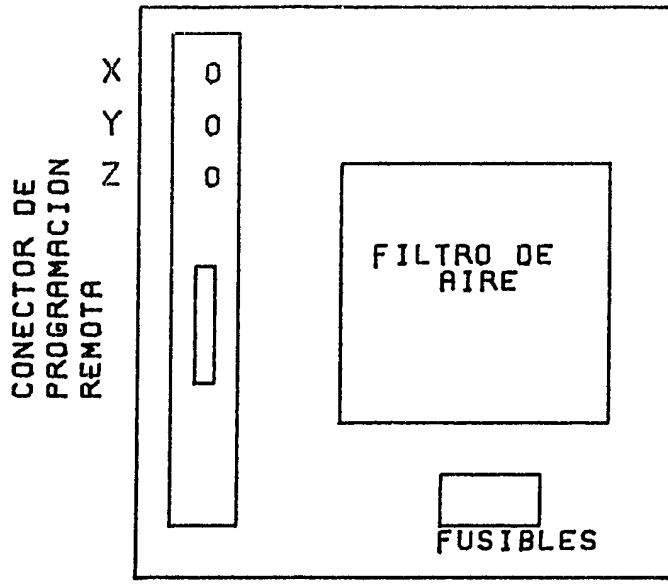


FIGURA 2.1

PARTE TRASERA DEL OSCILOSCOPIO



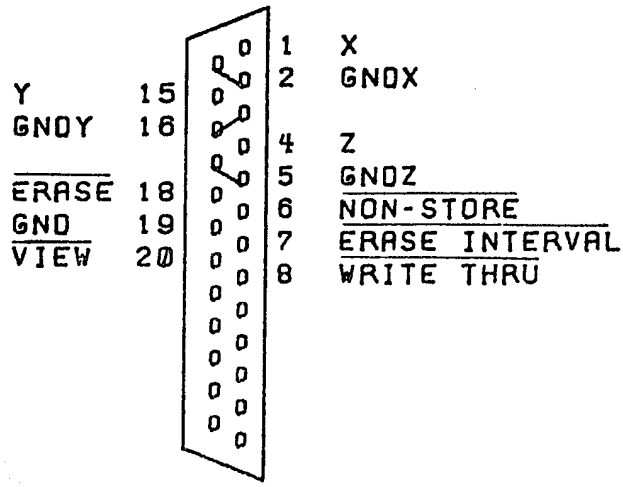


FIGURA 2.2

Aunque no se ha mencionado, se nota que externamente el osciloscopio 611 no tiene controles para adecuar la escala y posición relativa de las señales X o Y, estos controles se localizan en el interior del aparato y son poco prácticos si se tienen que realizar cambios continuamente. Tiene dos switches de tres posiciones que permiten a cada canal localizar la señal de voltaje cero a la izquierda, centro o derecha de la pantalla en el caso del eje X, y arriba, centro o abajo en el caso del canal Y. El control de atenuación esta bajo la forma de un circuito al cual se le montan resistencias de precisión para lograr el factor de atenuación deseado; estas resistencias deben ser soldadas a la placa y adicionalmente el acceso a dicho circuito es bastante restringido.

El formato de la pantalla puede ser de 162mm x 210mm o de 162mm x 162mm seleccionables a través de un switch localizado en el interior del osciloscopio. El osciloscopio fue acondicionado para trabajar con un formato de 162mm x 162mm, con la referencia de ambos canales al centro y con un factor de atenuación que permite 10v de variación en cada eje.

Las razones por las que se eligió este arreglo son las siguientes: en primer lugar, se prefiere el formato cuadrado sobre el rectangular para procurar que el tratamiento a cada canal sea idéntico y así detectar rápidamente las fallas, y principalmente porque el aspecto de 1:1 evita deformaciones al trazar, por ejemplo, círculos. Se eligió la señal a una amplitud de 10v con referencia en el centro de la pantalla para lograr requerimientos de voltaje entre -5V y 5V fácilmente asequibles de prácticamente cualquier computadora y, esperando una señal de ruido de aproximadamente 5mV, sería de esperarse una direccionabilidad en la pantalla de 0.081mm sobre cualquier eje; esto es, se lograría distinguir un par de puntos como diferentes, si se encuentran a más de 81 micras de separación, logrando así una resolución cercana a los 2000 pixeles en cada eje.

Como referencia, un dispositivo gráfico tipo raster equivalente, es decir, sólo con los atributos de encendido y apagado para cada pixel, necesitaría para su refrescado, de 512 KB de memoria y de procesadores gráficos muy sofisticados para lograr imágenes sin parpadeo.

2-2 El Graficador Incremental X-Y.

El graficador incremental X-Y, que de aquí en adelante se le mencionará como plotter, es básicamente un dispositivo electro-mecánico capaz de colocar una montura en un punto determinado en el plano, esto es en ciertas coordenadas [X,Y]. Dicha montura debe de poder desplazarse perpendicularmente al plano XY, es decir, en el eje Z, y aceptar al menos un tipo de elemento de escritura o contar ya con éste, para poder registrar un trazo sobre papel o acetatos.

El plotter con el que se cuenta es el modelo DP-1 marca Houston Instruments. Para lograr la posición de la montura (ver figura 2.3), tiene un par de motores de paso, uno para el eje X y otro para el eje Y. El motor para el eje X está conectado a través de engranes, a dos ejes con ruedas dentadas en cada extremo, las cuales aceptan papel de 11 pulgadas de ancho con perforaciones a los lados. En este caso en especial, entonces, la montura no se desplaza sobre el eje X sino que es el papel el que se mueve. Los dos ejes dan la tensión suficiente al papel para que éste no se despegue, hasta donde sea posible, del plano XY.

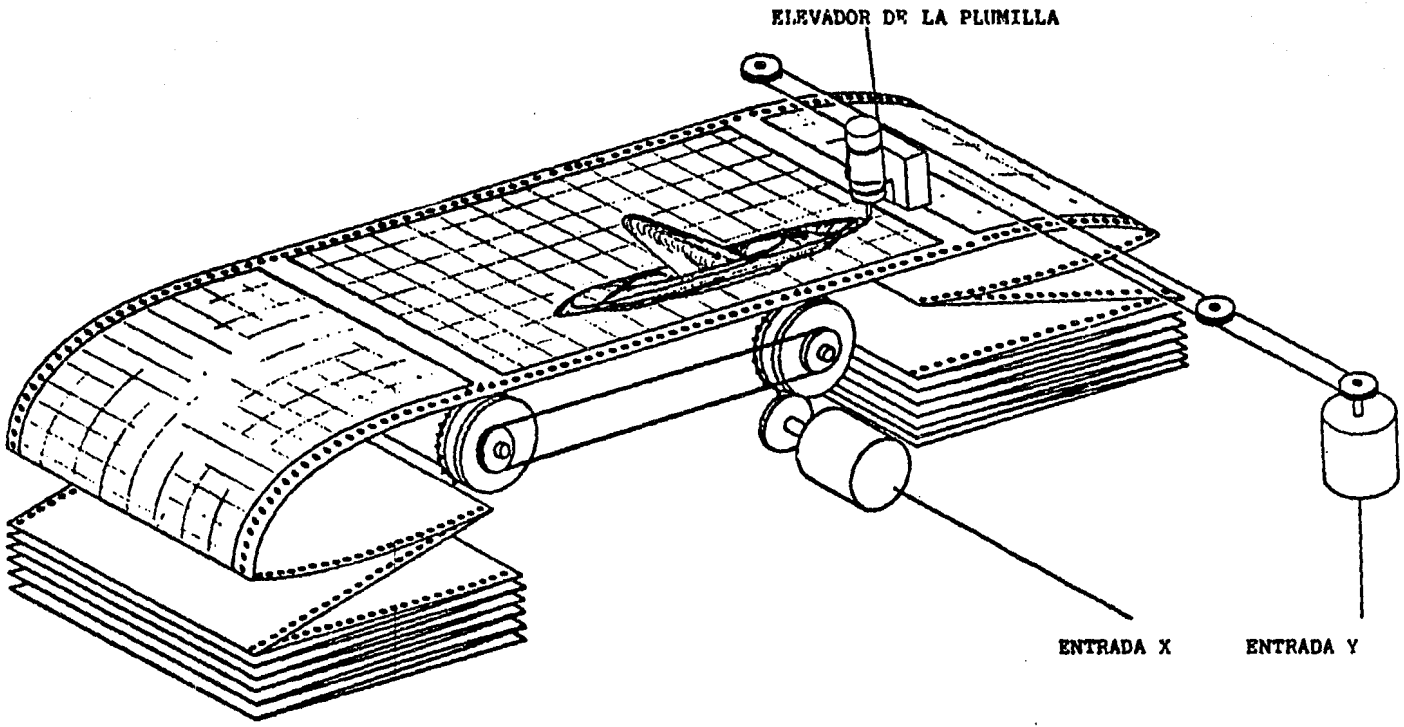


Diagrama Básico Funcional del Graficador Incremental

Figura 2.3

El motor conectado al eje Y, mueve mediante de un sistema de poleas, a la montura que se desliza a lo largo de un par de barras paralelas de acero con las que tiene contacto a través de un elemento de plástico para lograr un mínimo de fricción. A ambos extremos de las barras, existe un par de sensores que avisan al circuito manejador del motor que la montura se encuentra en alguno de los extremos y que no puede moverse más en el mismo sentido, esto evita dañar la montura contra los soportes de las barras. Posee una transmisión a base de poleas porque la montura está alojada en una tapa del plotter que debe ser alzada para poder colocar el papel en las ruedas dentadas.

Finalmente, la montura se desplaza en el eje Z impulsada por un solenoide. Este solenoide, al ser energizado, hace que la montura baje y se acerque al papel. Si está desenergizado, la montura está arriba; entonces, la posición normal de la montura, o de inicio, es hacia arriba. Por otro lado, es de notarse que el control sobre el eje Z es de dos posiciones únicamente.

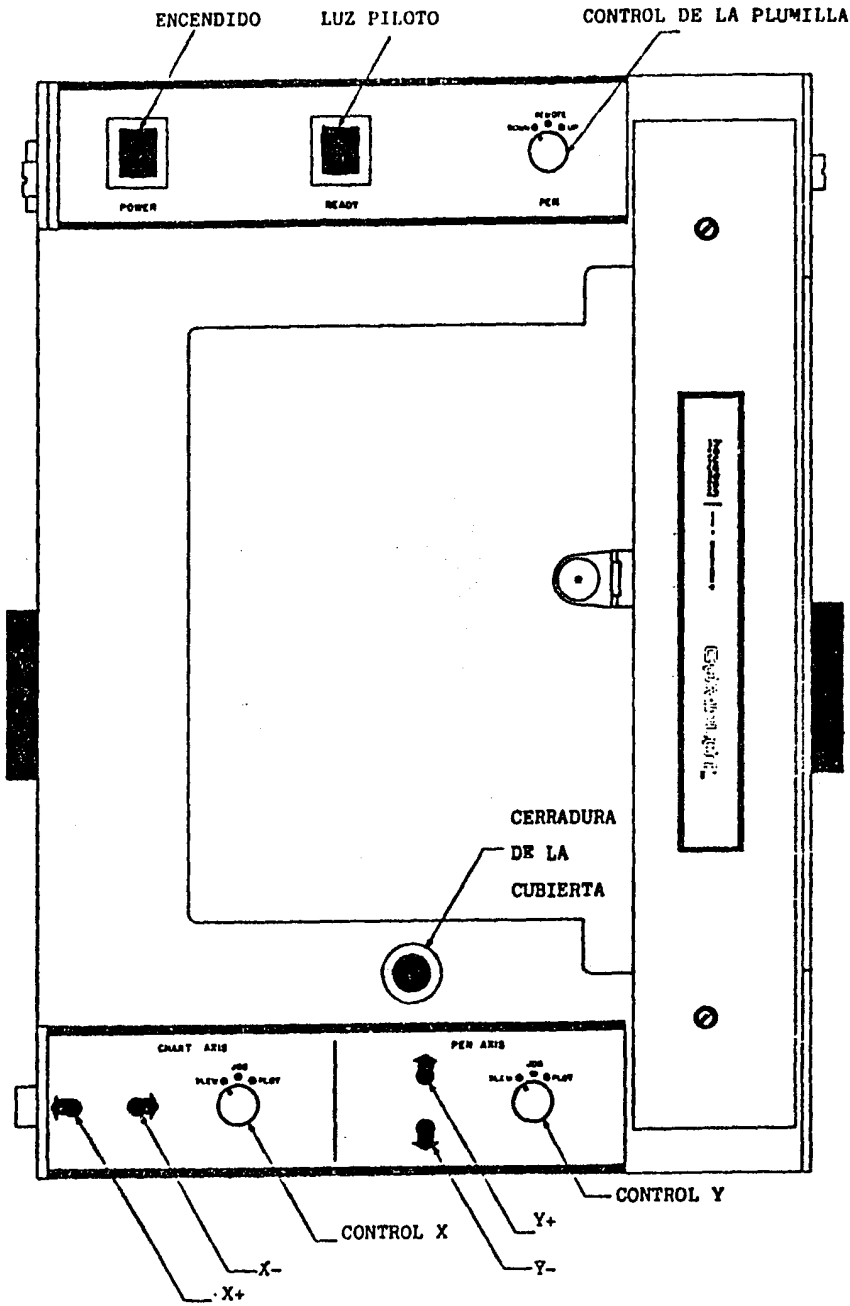
La montùra de este plotter acepta tres tipos de adaptadores. Existe un adaptador de repuestos de bolígrafo, estos deben ser de 62 mm de longitud aproximadamente, ser completamente lisos en sus superficie y de preferencia ser de metal. El adaptador de plumones, acepta plumones comerciales, no especiales, facilmente acequibles. Además, se tiene un adaptador de conos o *graphos* para emplear tinta

china, o tintas similares, comúnmente empleadas en los conos con los que se alcanzan trazos de alta calidad.

Todos los adaptadores tienen un cilindro interior donde se aloja el elemento de escritura, que se desliza dentro del adaptador. La posición de este cilindro puede ser fijada con un tornillo de sujección y con esto calibrar la altura a la que podrá bajar nuestro elemento de escritura. La montura cuenta tiene un anillo con tornillo que permite ajustes finos (menos de 1mm) de la posición de la punta del elemento de escritura.

El plotter tiene externamente tres grupos de controles, uno por cada eje mostrados en la figura 2.4. Para el eje Z (el solenoide), hay un control de tres posiciones para subir y bajar la montura y para otorgar el control remoto. Al pasar a modo remoto, la posición de la montura es la que tiene almacenada la tarjeta del manejador del solenoide, que fue inducida por control remoto o bien es el estado otorgado al encender el equipo.

Para los ejes X y Y los controles son similares. Hay un control de tres posiciones: modo remoto, modo pulso y modo tren de pulsos.



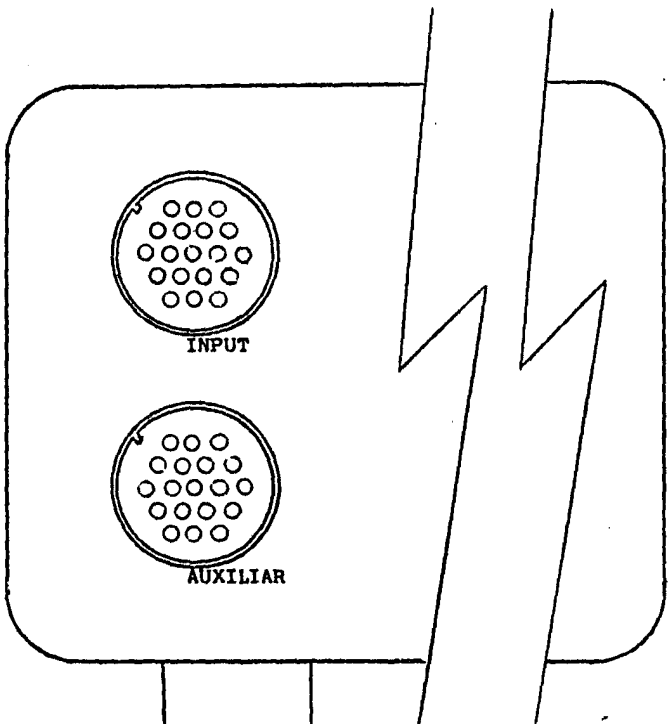
Controles del Graficador

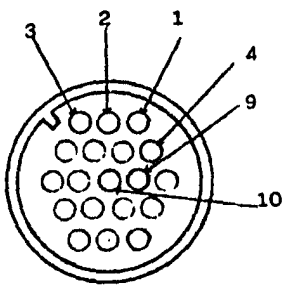
Figura 2.4

En el modo remoto, las señales son externas; en el modo pulso oprimiendo cualquiera de los dos botones asociados a cada control, manda el pulso adecuado al manejador del motor y provoca un solo paso en el sentido pedido. En el modo tren de pulsos, oprimir el botón provoca que un tren de pulsos (el número de pulsos depende del tiempo que el botón está presionado) sea enviado al manejador del motor de pasos correspondiente y origina un movimiento continuo en el sentido pedido de longitud proporcional al número de pulsos enviados (o al tiempo que se mantuvo oprimido el botón).

Al estar los tres controles en el modo remoto, se enciende una luz que indica que el plotter se encuentra listo para recibir comandos remotos.

Cuenta con dos conectores machos MIL-5015 cilíndricos de 19 puntas marca ITT-Cannon (ver figura 2.5), con los que el plotter recibe comandos de algún dispositivo remoto (por ejemplo una computadora). Las señales eléctricas que capta son de cuatro posibles tipos: acopladas en AC con respuesta al borde negativo o respuesta al borde positivo, o acopladas en DC con pulsos de voltaje positivo o negativo. Uno de los conectores MIL-5015 recibe las señales AC, mientras que el otro recibe las señales en DC.





INPUT

1 Y+
2 Y-
3 X+
4 X-
9 Z+
10 Z-

Entradas DC+

Figura 2.5 Conectores del Graficador

Específicamente se tomó el acoplamiento en DC con pulsos de voltaje positivo y en especial las características deben ser:

a) La impedancia de salida de la fuente debe ser menor a 500 ohms.

b) El voltaje activo debe ser mayor a 10 volts.

c) El ancho del pulso debe ser mayor a 15 microsegundos de acuerdo al manual de operaciones.

El plotter tiene motores de paso con capacidad para mover a la montura o al papel en 1/200 de pulgada, esto quiere decir que hay una resolución de 200 pasos por pulgada. La velocidad a que pueden trabajar estos motores es de 300 pulsos/seg, lo que da una rapidez máxima de 1.5 pulgadas/seg o 38.1 mm/seg en cualquier eje y hasta 53.88 mm/seg en diagonales.

El solenoide de la montura, entra en estado estable en menos de 100 milisegundos después de ser energizado o desenergizado. El manual recomienda que se permitan 100 milisegundos por cada cambio de estado del solenoide, esto permite 5 ciclos de energizado-desenergizado por segundo o 5 veces subir y bajar alternadamente la montura.

Comparando el plotter contra el osciloscopio de memoria, podemos decir que el plotter es un dispositivo de direccionamiento relativo a su posición en cualquier momento; en contraste, el osciloscopio posee direccionamiento absoluto comandado por por el valor de los voltajes aplicados a las entradas. Además, en el plotter la unidad mínima de trazo es un segmento de recta, mientras que para el osciloscopio es un punto.

Estas diferencias fundamentales en las características de los periféricos que se van a emplear, marcarán la pauta en la implementación de las rutinas básicas para el manejo de estos dispositivos.

Capitulo 3

Descripción de la tarjeta de acoplamiento.

3-1 Generalidades.

Se diseñó y construyó una tarjeta de acoplamiento entre la microcomputadora APPLE II y los dos periféricos gráficos que son, el osciloscopio de memoria y el plotter. Básicamente la tarjeta es capaz de generar, en el caso del osciloscopio, las señales analógicas para cada uno de los canales, la activación de los modos de operación, la señal de borrado y detección del intervalo de borrado y la activación del eje Z. Para el caso del plotter, generará con las características adecuadas, las seis señales para mover la montura del plotter en los seis sentidos básicos posibles y los retardos necesarios.

La fuente de la APPLE II provee niveles de voltaje de +5V, -5V, +12V y -12V que como se describió, se amolda perfectamente a los requerimientos globales para la conexión con los periféricos, mencionados en el capítulo anterior. Sin embargo, la corriente que entrega es reducida por lo que se debe tener en cuenta esta restricción en el diseño de la tarjeta.

3-2 Conexión al osciloscopio.

Se deseaba tener más de 2000 pixeles de resolución en cada eje, en la pantalla del osciloscopio, lo cual, como se describió en el capítulo pasado, es perfectamente viable. Para esto, se requiere una señal digital de al menos 11 bits por cada canal que serían convertidos en una señal analógica en un convertidor D/A. En total se requiere un mínimo de 22 bits para manejar las señales de los canales X y Y.

Hay además 4 señales de salida y una de entrada, el intervalo de borrado, esta última resultaría conveniente manejarla a través de interrupciones porque su período es muy grande (500ms).

Las entradas de selección de modo en el osciloscopio, responden a nivel, y están diseñadas para ser manejadas por un circuito en colector abierto, o con más de 10V.

Las entradas para la señal de borrado y del eje Z en el osciloscopio, responden a pulso, con salida de colector abierto. Por el período que requieren, es conveniente que sean manejadas por medio de un monoestable.

La salida del osciloscopio del intervalo de borrado, tiene un nivel de 10V, por lo que se necesita de un transistor para acoplar esta señal a niveles TTL.

En la figura 3.1 se muestra un diagrama de bloques del circuito mínimo para acoplamiento al osciloscopio.

Diagrama de bloques del circuito para el osciloscopio

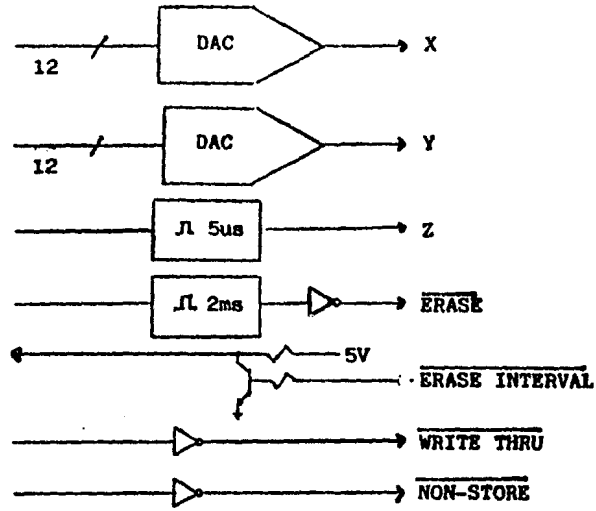


Figura 3.1

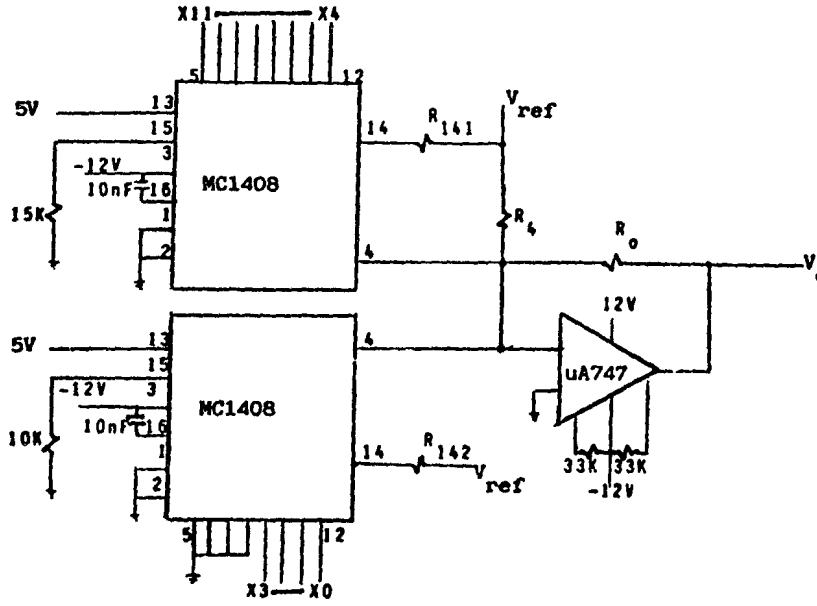


Figura 3.2 Configuración de un convertidor D/A

3-3 Descripción de los convertidores DA

En la sección de los convertidores DA donde se generan las señales para el osciloscopio, se seleccionaron 12 bits de capacidad de resolución que permitirían 4048 pixeles en cada eje. Se pensó en la posibilidad de emplear convertidores de 12 bits pero su costo era elevado.

Se optó entonces por emplear un par de circuitos MC1408 que son DAC de 8 bits que se consiguen fácilmente en México; uno de ellos manejando los 8 bits más significativos y el otro manejando los 4 bits menos significativos. Ambos quedan acoplados por medio de un sumador diseñado alrededor de un amplificador operacional, que por simplicidad, se escogió un uA747 que contiene dos AmOp, precisamente lo requerido para manejar nuestros dos canales del osciloscopio.

Ya que se quería una señal bipolar entre +5V y -5V se empleó la configuración mostrada en la figura 3.2. La primera consideración que se hizo fue siguiendo el manual, $R_o=R_4$ y fue calculado ajustando tres puntos críticos: $800=0V , $FFF=+5V , $000=-5V. Las ecuaciones son las siguientes:

$$V_o=V_{o1}+V_{o2}$$

$$V_{o1}=([x_{11}/2+x_{10}/4+...+x_4/256]/R_{14}-1/R_4)*V_{ref}*R_o$$

$$V_{o2}=[x_3/32+x_2/64+x_1/128+x_0/256]*V_{ref}*R_o/R_{14}$$

Para $800=0V

$V_{o1}=R_o*V_{ref}*[1/(2*R_{141})-1/R_4]$

$V_{o2}=0V$

$V_o=V_{ref}*[R_o/(2*R_{141})-1]=0V$

$V_{ref}<>0$

$R_o=2*R_{141}$

Para $000=-5V

$V_{o1}=-V_{ref}*R_o/R_4$

$=-V_{ref}$

$V_{o2}=0V$

$V_o=-V_{ref}=-5V$

$V_{ref}=+5V$

Para $FFF=+5V

$V_{o1}=R_o*V_{ref}*[255/(256*R_{141})-1/R_4]$

$=127*V_{ref}/128$

$V_{o2}=15*R_o*V_{ref}/(256*R_{142})$

$V_o=(75*R_o/R_{142}+1270)/256=+5V$

$75*R_o/R_{142}=10$

$R_{142}=7.5*R_o$

$R_{142}=15*R_{141}$

$I_{ref1}=V_{ref}/R_{141}$

$I_{ref2}=V_{ref}/R_{142}=I_{ref1}/15$

Analizando varias combinaciones de resistencias se determinó que para usar el menor número de resistencias comerciales y mantener el consumo de corriente a niveles razonables, el circuito empleará la siguiente combinación:

$R_o=R_4=2.2K$
$R_{141}=1.1K=2.2K||2.2K$
$R_{142}=16.5K=15K+1.5K$

con
$I_{ref1}=4.5$ mA
$I_{ref2}=0.3$ mA

Realizando experimentos se observó que R_{142} podía ser simplemente 15K ahorrando una resistencia por canal y espacio de la tarjeta.

3-4 Acoplamiento al Plotter.

El graficador incremental o plotter, es manejado con seis señales y el control es de malla abierta, por ende los tiempos de respuesta deberán ser controlados por medio de la tarjeta o por programa.

Las señales que maneja son:

a) Z+ Sube la pluma.

b) Z- Baja la pluma.

c) Y+ Mueve la montura hacia arriba.

d) Y- Mueve la montura hacia abajo.

e) X+ Mueve el papel a la derecha.

f) X- Mueve el papel a la izquierda.

La pareja de señales Z comandan un solenoide y su tiempo de respuesta es de 100ms con un ancho del pulso de activación de 150us.

Las señales X y Y comandan un par de motores de paso con un circuito manejador que tiene un tiempo de respuesta de 3.3ms y un ancho de pulso de activación de 120us.

El conjunto de las seis señales fue conectado a través de la entrada DC+ del plotter que requiere voltajes de activación de más de 10V con una impedacia de salida de 500 ohms o menos.

Cuando en una pareja de señales (+ y -) ambas se encuentran activas, éstas se cancelan mutuamente. Si entonces alguna de ellas se desactiva, se detecta la activación de su contraparte en forma normal siempre que se

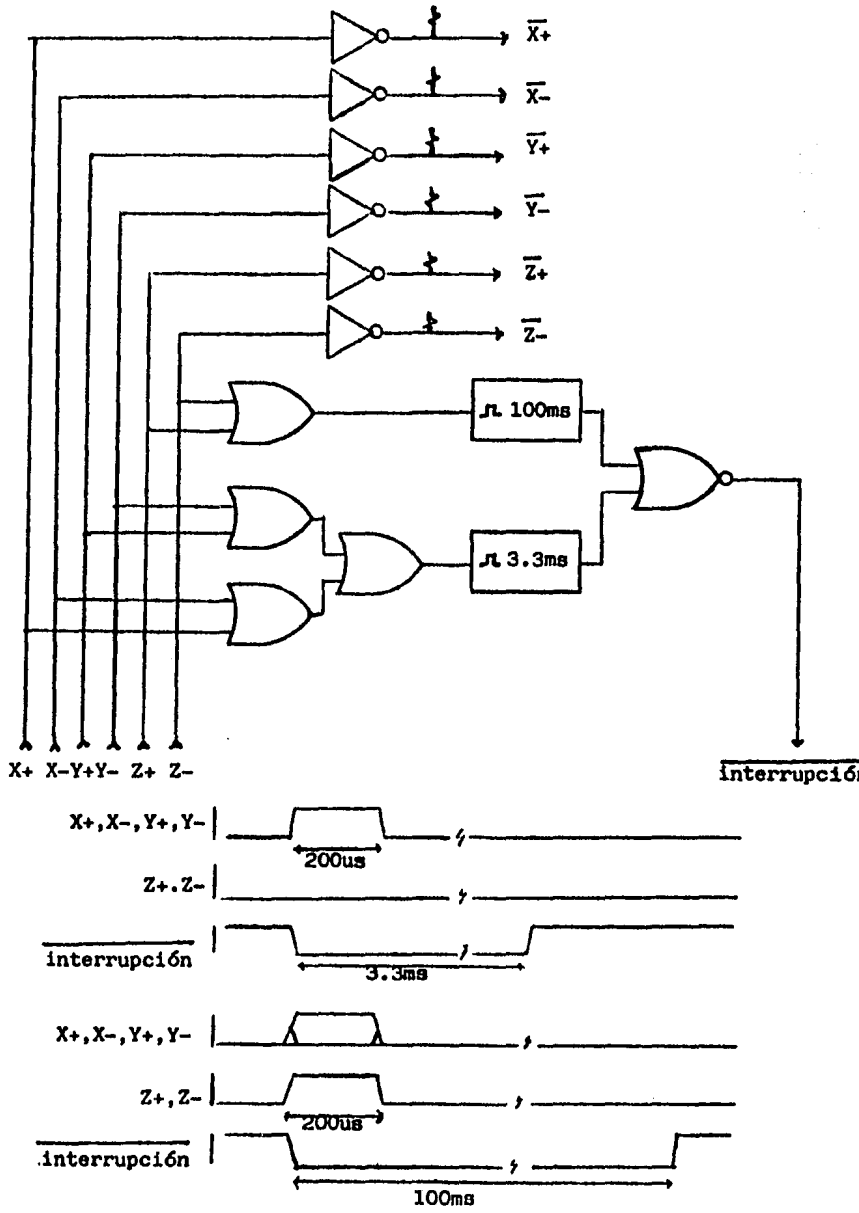
efectúe un retorno al estado activo por parte de la primera señal.

Se decidió que los tiempos de respuesta fueran administrados por la tarjeta y se establecerá un protocolo a través de interrupciones. Por otra parte, el ancho de los pulsos se dará por medio de programa y este fue fijado de modo experimental en cerca de 200us para todo el conjunto. La figura 3.3 muestra en diagrama de bloques el circuito requerido.

Para los tiempos se empleó un monodisparador 74LS123 dual. La estimación de la red RC fue casi experimental pues se encontró que las ecuaciones dadas en el manual TTL coincidían sólo para periodos menores de 100us.

3-5 Acoplamiento con la computadora.

La descripción de los acoplamientos a los periféricos gráficos menciona que se requiere al menos de 32 bits disponibles en paralelo más 2 canales de interrupción. Principalmente se tenían dos posibles enfoques, acoplar por medio de circuitos SSI y MSI o bien por circuitos LSI.



Diagramas del circuito para conexión al Graficador

Figura 3.3

En el primer enfoque, se requeriría de latches (4 de 8 bits) y de buffers aparte de varios circuitos SSI para desarrollar la lógica que hiciera viables las interrupciones y la administración de un solo bus de 8 bits que tiene la computadora.

El segundo enfoque emplearía circuitos LSI, en este caso se emplearía un par de Adaptadores de Acoplamiento Periférico o PIA por sus siglas en inglés, el PIA es un circuito con puertos paralelos, en el caso de la APPLE II con un procesador 6502, el PIA corespondiente sería un 6520. Otros PIAs que se pueden usar son el 6820 y el 6821. Para los propósitos de la tarjeta, cualquiera de los circuitos mencionados, cumple con los requisitos, pero se usaría un MC6821 por su bajo costo.

Una comparación burda en el renglón de costos, sería, tomando como unidad el costo de dos MC6821. Un diseño MSI ocuparía cuatro latches octales como el 74LS373 cada uno costando la mitad de un solo PIA. Si se continuara con el diseño MSI, rápidamente el costo llegaría al doble o tal vez más del costo de un diseño con LSI, independientemente de que al mantener el número de partes al mínimo, la confiabilidad aumenta y la construcción se simplifica.

Por otro lado, comparando con los circuitos MSI, el PIA es un circuito CMOS y tiene un consumo mucho menor, y como se mencionó anteriormente, hay restricciones con respecto a la potencia suministrada por la microcomputadora APPLE II.

Por las razones antes expuestas, la solución adecuada para el acoplamiento con la computadora, resulta ser un diseño LSI, con un par de MC6821.

3-6 Descripción del PIA.

El PIA cuenta con:

a) Dos puertos de 8 bits.

b) Dos bits de control asociados a cada puerto, para protocolos.

c) Bus de datos de 8 bits.

d) Seis registros internos mapeados a cuatro localidades de memoria.

e) Generación de interrupciones compatibles con el up6502.

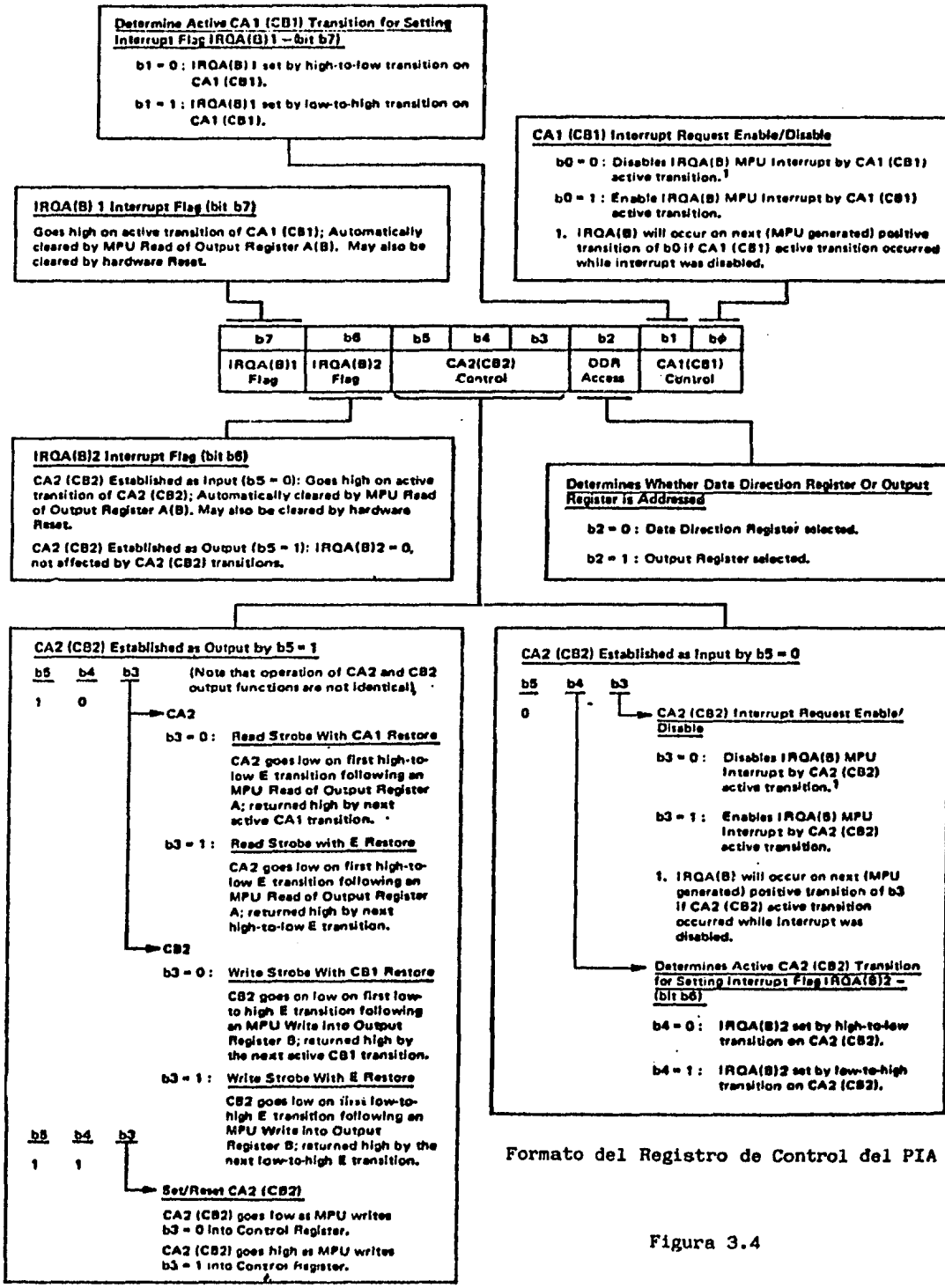
f) Bus de control compatible con up6502.

Los puertos de datos paralelos reciben el nombre de A y B y cada uno de los 16 bits puede ser programado para entrada o para salida. Los bits de manejo de protocolo se denominan CA1, CA2, CB1 y CB2, y tienen propósitos específicos.

Los puertos A y B no son exactamente iguales. El puerto B maneja corrientes de salida más altas y sus bits de protocolo se configuran de modo ligeramente diferente al de los bits de protocolo del puerto A. Por simplicidad, no se empleará en el diseño, características específicas de cada puerto sino que se procurará hacer puertos gemelos y se usarán los bits de protocolo del puerto A y los del puerto B quedarán disponibles como bits de datos.

La línea de interrupción del PIA se puede conectar a $\overline{IRQ}$ o a $\overline{NMI}$ del up6502 y puede ser activada por transiciones positivas o negativas en CA1 o CB1. La línea de lectura/escritura es igual que la línea correspondiente al up6502. Los diagramas de tiempos del MC6821 y del up6502 son compatibles. La línea del $\overline{RESET}$ en el PIA borra los seis registros internos.

Los registros internos se dividen en dos grupos de tres, un grupo por cada puerto. De cada grupo únicamente dos son seleccionados a un mismo tiempo, uno de ellos, siempre activo, es el registro de control en escritura o de status en lectura, a través del bit 2 de este registro, se selecciona cual de los dos registros restantes será accesible. La figura 3.4 muestra la programación de este registro.



Formato del Registro de Control del PIA

Figura 3.4

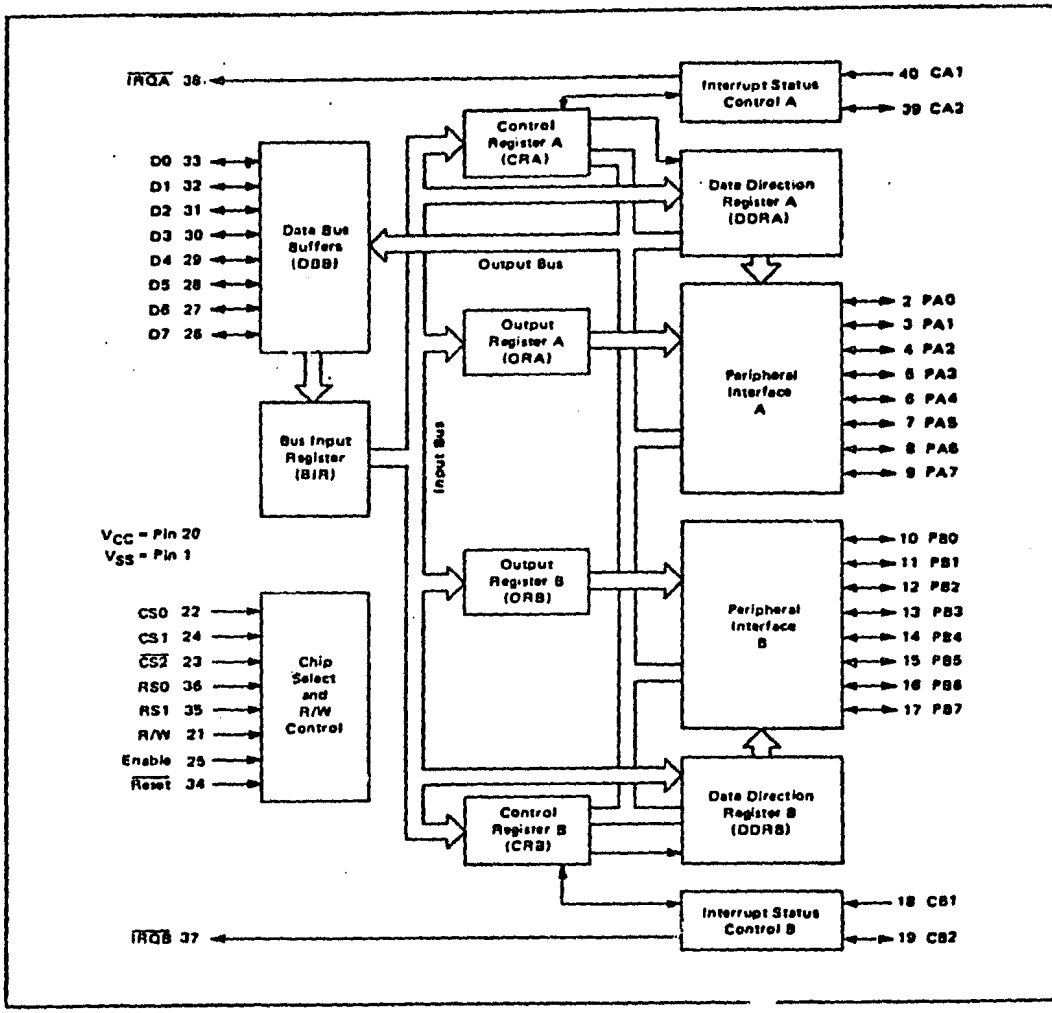


Diagrama interno de bloques del PIA

Figura 3.5

Los registros restantes son uno de datos y otro de programación de dirección de los datos, esto es, programa los bits como salidas o entradas. La figura 3.5 muestra la configuración lógica interna del PIA.

3-7 Descripción global

Se emplearon un par de PIAs con cuatro grupos de 8 bits y otros 4 bits adicionales, es decir, un total de 36 bits de entrada o salida en paralelo, con cuatro bits más de entrada exclusivamente; más que suficiente para la aplicación que demanda un mínimo de 32 bits de salida y 2 de entrada.

Para el osciloscopio, se pide al menos 22 bits para las señales de los canales o sea, 3 puertos. Para evitar perturbaciones al manipular otros registros, estos 3 puertos serán dedicados exclusivamente a las señales XY del osciloscopio y ya que la adición no cuesta prácticamente nada, se elevó el número de bits por canal a 12, aumentando la máxima resolución posible a 4096 intervalos por canal.

La señal del intervalo de borrado se alimentará por uno de los CA1. La activación del borrado se efectuará por su pareja CA2. La señal del eje Z se mandará por el restante CA2 y las 2 señales de selección de modo se mandarán desde bits del puerto libre hasta el momento.

Las 6 señales para el plotter encajan perfectamente en el mismo puerto de los bits de selección del modo de

operación del osciloscopio, mientras que la interrupción para permitir un nuevo pulso o pulsos al plotter, se recibirá por el bit CA1 libre.

En total se ocuparon 36 bits equivalentes a un 90% de uso de las salidas de los PIAs, con 4 bits libres para enfrentar posibles contingencias.

El diagrama total del circuito se encuentra en el Apéndice B, en el se muestra la disposición final de los registros que es como sigue:

Registro	Dirección relativa
PDRA1	0
PDRB1	1
PDRA2	2
PDRB2	3
PCRA1	4
PCRB1	5
PCRA2	6
PCRB2	7

Donde DR significa registro de datos o de dirección, CR registro de control o status, A y B, se refieren al puerto A o B respectivamente y 1 y 2 se refieren al primer PIA o al segundo PIA respectivamente.

Las direcciones son relativas a la dirección a que responda la señal de habilitación de la tarjeta. En las ranuras de la APPLE II, la señal *Device Select* ($\overline{DS}$) que es activa baja (como se requiere), se activa cada vez que se direcciona una localidad entre $C080+N0 y $C080+NF donde N es el número de la ranura seleccionada.

Empleando $\overline{DS}$, se simplifica la descodificación de las direcciones, de hecho si permitimos que las direcciones tengan gemelos en el siguiente bloque de 8, no se requiere ningún circuito adicional y la conexión es directa al bus de direcciones de la APPLE II.

Si por ejemplo la tarjeta se localizara en la ranura 3, las direcciones del bloque primario de los registros quedarían entre $C0B0 y $C0B7.

Una de las principales bondades de la tarjeta diseñada es el gran aprovechamiento de los circuitos, casi del 99% El único desperdicio que hubo fueron 4 de las 40 señales de entrada/salida los PIAs, si tomamos la conexión al exterior como el 50% del costo del PIA y dado que los PIAs forman el 27% del costo en componentes, se llega a la cifra de 1.2% de desperdicio en componentes.

Capitulo 4

Rutinas Básicas

4-1 Introducción

Las rutinas básicas del paquete gráfico se dividen en dos partes: los manejadores de dispositivos o *device drivers* y las rutinas de enlace. Las primeras, fueron programadas en ensamblador por razones de velocidad e incluyen las rutinas de inicio de la tarjeta de acoplamiento, las rutinas de servicio de interrupción, borrado de la pantalla del osciloscopio, inicio del plotter y trazado de líneas.

Las rutinas de enlace fueron programadas en UCSD Pascal porque es el sistema de desarrollo más robusto, completo y homogéneo con que cuentan las microcomputadoras Apple II. Por otro lado permite la creación de rutinas de biblioteca y agregar rutinas externas escritas en un macro-ensamblador llamado TLA que emplea el concepto de funciones y procedimientos, necesario para el acoplamiento de los manejadores de dispositivos a Pascal.

4-2 Rutinas de los manejadores de dispositivos.

Son básicamente cuatro procedimientos; las rutinas **INICIAPIA, BORRAOSC, INIPLT** y **BDDA.** Las rutinas de servicio de interrupción al tener que existir dentro de un procedimiento o función, fueron incluidas dentro del procedimiento **INICIAPIA.**

La rutina **INICIAPIA** tiene por función programar los circuitos PIA de la tarjeta de acoplamiento e iniciar algunas variables empleadas por las rutinas de este nivel que son: **PLM, OSCMOD, OSCRDY** y **PLTRDY.** **PLM** indica la posición de la pluma del plotter, si esta arriba, vale 2, abajo vale 1; estos valores corresponden directamente con las máscaras necesarias para el puerto. **OSCMOD** es la máscara de programación del modo de operación del osciloscopio, típicamente se emplea el modo normal por lo que el valor debe ser 0.

OSCRDY y **PLTRDY** tienen por función, informar acerca del estado actual de los dispositivos, **OSCRDY** se refiere al osciloscopio, mientras que **PLTRDY** se refiere al plotter. Si el valor que tienen es 0, significa que el dispositivo está libre, un valor diferente de 0 indica un estado de ocupado.

Las rutinas de servicio de interrupción que se alojan en la rutina **INICIAPIA,** tienen un punto de entrada común y detectan el origen de la interrupción entre el osciloscopio

y el plotter, cualquier otro sistema que interrumpa provocará que la microcomputadora quede atrapada en un lazo sin fin porque no se podrá atender la interrupción. Al detectar el origen de la interrupción, avisan que se está dando servicio al solicitante y ponen en 0 a **OSCRDY** o a **PLTRDY** dependiendo del origen de la interrupción. De este modo el dispositivo puede ejecutar tareas de modo concurrente a la operación del CPU, sin tener que atar este último, sólo cuando trata de accesar al dispositivo, entrando a una espera activa o *busy waiting*.

Los circuitos PIA son programados para que sus puertos paralelos sean salidas, las señales de control B (CB1 y CB2) son desactivadas y las dos líneas CA1 son programadas para recibir interrupciones. El PIA1 recibe interrupciones con borde de bajada mientras que PIA2 las detecta por borde de subida.

La rutina **BORRAOSC** espera a que la bandera **OSCRDY** baje y al momento que el osciloscopio queda libre, le manda la señal de borrado a través de CA2 de PIA1. El ancho del pulso queda controlado por un monodisparador ajustado a 2ms con un inversor de colector abierto a la salida. Sólo resta bajar la señal de borrado e indicar que el osciloscopio está ocupado por medio de **OSCRDY**. De este modo el CPU no queda inutilizado esperando a que el osciloscopio responda que ya terminó de borrar.

La rutina **INIPLT** manda las señales para levantar la montura de la pluma y hace que ésta se mueva a lo largo del eje Y hasta topar con el borde derecho y marcar dicho punto como el origen del plano gráfico para el plotter. La rutina manda 2200 pulsos, como no hay realimentación del plotter, el exceso de pulsos es ignorado por el mismo plotter una vez que llega al borde donde un switch es presionado. Por modularidad, emplea las mismas rutinas definidas en **BDDA** para la transmisión de pulsos al plotter.

Finalmente la rutina **BDDA** se divide primordialmente en tres partes; un encabezado que calcula parámetros globales, las rutinas de generación de vectores para el osciloscopio y las rutinas de generación de vectores para el plotter. Esta distinción entre las rutinas del plotter y del osciloscopio se hizo debido a la naturaleza propia del plotter, posicionamiento relativo e incremental; mientras que el osciloscopio es de direccionamiento absoluto y puntual. La selección del bloque de rutinas para un dispositivo u otro se efectúa con la variable **DISP**.

Con objeto de ocultar dichas diferencias al usuario, se definió el trazo de vectores o líneas, como el elemento gráfico más simple. Esto es consistente con el paquete gráfico **CALCOMP** y sus descendientes como PLTXY de la computadora VAX. Para trazar un punto, se deberá recurrir al trazo de vectores de longitud cero.

Las rutinas BDDA(X1,Y1,X2,Y2,TIPO) emplean el algoritmo Bresenham que a su vez está basado en el algoritmo DDA (*Diferential Digital Analysis* o Análisis Diferencial Digital) aplicado a la ecuación de la recta. Los algoritmos de generación de vectores deben de cumplir ciertos requisitos como: marcar el menor número de pixeles sin que aparezcan huecos en el trazo y comenzar el trazo en la coordenada inicial y terminarlo en la coordenada final (esto es muy importante para el plotter por su posicionamiento relativo) en este caso las coordenadas iniciales corresponden a (X1,Y1) y las finales a (X2,Y2).

Ambos algoritmos, el DDA y el Bresenham cumplen con los requisitos pero el Bresenham tiene más bondades que el DDA. Por un lado el DDA emplea números reales y Bresenham emplea enteros; por otro lado el DDA emplea divisiones que en general consumen mucho tiempo mientras que Bresenham sólo usa divisiones y multiplicaciones por 2 o lo que es lo mismo, corrimientos de bits, por lo que, se acomoda perfectamente para un programa en ensamblador como se deseaba en este caso. El algoritmo Bresenham está enfocado a aprovechar las características discretas de los dispositivos gráficos reales.

El DDA emplea las siguientes ecuaciones paramétricas:

$$x=x1+(x2-x1)*u$$

$$y=y1+(y2-y1)*u$$

donde (x1,y1) y (x2,y2) son las coordenadas extremas del

segmento de recta y u es un parámetro que varía entre 0 y 1 y la rapidez con que lo hace está en función del número de pixeles que debe de cubrir que a su vez es el máximo entre los pixeles que se cubrirían moviéndose sólo sobre X o Y.

El Bresenham siempre se mueve sobre el eje con mayor diferencia de pixeles y detecta cuando corresponde un incremento sobre el eje restante con un error máximo de 1/2 pixel empleando sólo sumas y restas. Esto significa que básicamente se podrían hacer dos rutinas, una para eje X dominante y otro para eje Y dominante como efectivamente fue codificado.

Regresando a las generalidades de la rutina BDDA se debe mencionar que el parámetro de entrada **TIPO** puede recibir un número n entre 0 y 127 que representa el tipo de línea escogido, entendiéndose como tipo de línea el trazo alternado de n pixeles encendidos seguidos de n pixeles apagados. También debe mencionarse que la variable externa **PLM** únicamente afecta la ejecución si tiene un valor diferente al que tuvo en la última llamada a **BDDA**.

La rutina para el osciloscopio contempla la posibilidad de que éste se encuentre borrando y se espera a que termine. La rutina dedicada al plotter manda los pulsos a los motores solo si los pulsos anteriores ya tuvieron el tiempo suficiente de ser ejecutados (se maneja por interrupciones).

4-3 Rutinas de enlace.

Estas rutinas escritas en Pascal proporcionan una vía de acceso a los procedimientos escritos en ensamblador de una manera más clara y coherente. Agregan una serie de conceptos como son ventanas, puertos de visión (*viewport*) o simplemente puertos, recorte de rectas al cruce con los bordes de la ventana, escritura de caracteres, administración de un archivo de eco, manejo de cursores y detección de errores.

El concepto que se maneja de la ventana es el de un marco rectangular con lados paralelos a los ejes coordenados que define la zona del mundo del usuario que será vista, esto quiere decir que se define en coordenadas virtuales o de usuario. Una ventana queda definida completamente por las coordenadas de la esquina inferior izquierda y de la esquina superior derecha. La esquina inferior izquierda siempre deberá quedar en el tercer cuadrante respecto a la esquina superior derecha.

El puerto es empleado como la porción de la pantalla del osciloscopio o la zona del plotter que servirá para representar en su totalidad a la ventana por tanto su definición queda sujeta a las mismas reglas que la ventana, a excepción que maneja coordenadas del dispositivo; para el osciloscopio queda entre 0 y 2047 en ambos ejes y para el plotter entre 0 y 2199, también en ambos ejes.

Cada vez que se invoca la rutina de iniciación gráfica, el puerto se redefine como la totalidad del dispositivo seleccionado y la ventana como un reflejo de este puerto.

Las rutinas **SEL_VENTANA** y **SEL_PUERTO** cumplen con la función de definir puerto y ventana, de filtrar los valores pedidos y de calcular los parámetros para mapear los segmentos de recta de la ventana al puerto.

La selección de la ventana implica que la pluma se encuentra ya dentro del puerto por lo que su posición mapearía a un punto dentro de la ventana. Dicho punto correspondería al cursor virtual o del mundo del usuario y se almacena en la variable **CM** y se recalcula al seleccionar una nueva ventana. **CM** es de tipo **PUNTO** que a su vez es un registro de una pareja de variables reales X y Y, se tiene acceso a esta variable desde el programa que llama a las rutinas de enlace pero se recomienda no alterar su información.

Por otro lado la selección del puerto implica la posibilidad de que el cursor del dispositivo mantenido en la variable **CD** no esté dentro del puerto pedido por tanto al seleccionar un puerto, la pluma se desplaza a la esquina inferior izquierda y ambos cursores, **CD** y **CM** se ajustan.

Existen tres rutinas para desplazamiento de la pluma en el plano: **LINEA_HACIA(X,Y)**, **LINEA(X1,Y1,X2,Y2)**, y **MUEVE_HACIA(X,Y)**. **LINEA_HACIA(X,Y)** mueve la pluma desde **CM** hasta el punto de coordenadas virtuales X,Y con la pluma subida o bajada dependiendo del último comando relacionado

con la posición de la pluma en el eje Z. El segmento de recta que se generaría es aquel que la rutina de recorte o *clipping* regresa, tal que este es un subsegmento de la recta que se pidió trazar y que está completamente dentro de la ventana previamente definida.

A pesar que se efectúe un recorte, CM conservará siempre el cursor virtual correcto o sea el del punto X,Y pedido, evitando deformaciones en la imagen que se está trazando y conservando correcta la metáfora de ventanas.

LINEA(X1,Y1,X2,Y2) es similar a **LINEA_HACIA** excepto que primero hay un desplazamiento con la pluma subida hacia X1,Y1 y después con la posición de pluma prefijada, hacia X2,Y2. Todas las coordenadas son virtuales y CM termina con el valor X2,Y2. Las restricciones del recorte se vuelven a cumplir en este caso.

MUEVE_HACIA(X,Y) es idéntico a **LINEA_HACIA** pero se garantiza la pluma levantada antes del movimiento regresando a su estado anterior al término de la rutina. De nuevo se aplica un recorte al vector y CM adquiere el valor X,Y.

Si un vector provoca, debido al recorte, una entrada a la ventana, la pluma se desplaza levantada al punto de ingreso. Si el recorte provoca una salida, al llegar la pluma al cruce con el extremo de la ventana, ésta se detiene y se levanta.

Para indicar si la pluma trazará o no, se definieron las rutinas **PLM_ABJ** que baja la pluma (hay trazo) y **PLM_ARR** que la sube (no hay trazo). Ninguna recibe parámetros y su

efecto no se nota sino hasta efectuar un movimiento con alguna de las tres rutinas mencionadas anteriormente (LINEA_HACIA, LINEA y MUEVA_HACIA).

Las rutinas de escritura de caracteres son dos, ESC_STR y ESC_CAR. ESC_STR(STR, ESC, ORIENTA) escribe la cadena de caracteres STR a escala ESC y con orientación ORIENTA; estos términos se describen a continuación. Cada carácter de la cadena se pasa a ES_CAR.

ESC_CAR(CAR, ESC, ORIENTA) escribe un solo carácter a escala ESC y orientación ORIENTA. Los caracteres tienen un tamaño de 7x7 unidades de dispositivo y la escala que es un entero, define el factor de amplificación deseado; un valor negativo provocaría una rotación de 180 grados. La orientación puede tomar los valores ARR, IZQ, ABJ y DER que provoca rotaciones de 0, 90, 180 y 270 grados respectivamente; ORIENTA es de tipo ORIENTACION. Si la rejilla de 7*7 escalada y rotada rebasa la ventana, el carácter no se traza; si hubiera trazo, CM tendría la última posición del carácter que debe ser 7,0 dentro de la rejilla, listo para trazar otro carácter.

Dos rutinas más redondean el conjunto de rutinas de enlace que el usuario emplearía normalmente, INI_GRAF y TERM_GRAF. INI_GRAF(DISP,B_ECO) inicia los parámetros para comenzar la transmisión de comandos gráficos ya sea hacia el osciloscopio o al plotter pudiendo efectuar un eco de los comandos hacia un archivo.

DISP es una variable de tipo DISPOSITIVO que puede tener los valores OSC y PLT cuyo significado es respectivamente osciloscopio y plotter; con ella se selecciona el dispositivo de salida gráfica. Si se pide OSC, la pantalla del osciloscopio es borrada con BORRAOSC, si se pide PLT se llama a INIPLT. B_ECO es la bandera que indica si se desea un eco o no y el nombre del archivo se pasa en la variable global NOM_AR_ECO (NOMbre del ARchivo para ECO de comandos); el achivo en sí es la variable tipo archivo ECO cuyos registros son de 5 bytes representando el comando y hasta dos parámetros.

TERM_GRAF(BORRA) cierra el archivo de eco y si BORRA es verdadera, cierra el dispositivo gráfico seleccionado en ese momento. Esta rutina se debe llamar obligatoriamente si se está efectuando eco, muy recomendable si la salida es a plotter y opcional en cualquier otro caso. Si se está efectuando el eco y se vuelve a llamar a INI_GRAF sin haber llamado a TERM_GRAF, se marcará error.

Al correr un programa que usa estas rutinas de enlace, automáticamente se llama INI_GRAF para OSC sin eco. También se carga la definición de los caracteres del archivo *:SYSCAR donde el * significa el disco raíz del sistema UCSD.

Hay tres rutinas más cuyo uso es poco usual; **LEE_CAR**, **PROC_GRAF** y **DAT_GRAF**. **LEE_CAR(NOMBRE)** lee un archivo con nombre **NOMBRE** que debe tener la definición de los caracteres. La carga se hace con procedimientos no convencionales que leen por bloques los archivos.

PROC_GRAF sólo puede ser llamada si no se está efectuando eco. Su objetivo es interpretar el archivo gráfico de eco generado previamente con el nombre pasado en **NOM_AR_ECO**. El archivo de eco tiene coordenadas de vectores recortados y mapeados al puerto, por tanto lleva coordenadas de dispositivo. Al llamar a **PROC_GRAF** se selecciona una ventana para representar todo el plano del dispositivo activo al momento de efectuar el eco y los caracteres que fueron trazados, son reescalados al puerto activo actualmente.

DAT_GRAF(PREG) permite la lectura de todos los parámetros que maneja el sistema seleccionándolos por el parámetro **PREG** con las equivalencias siguientes:

```
 0 : Dispositivo, 0=OSC, 1=PLT
 1 : Pluma, 0=abajo, 1=arriba
 2 : Tipo de línea
 3 : Eco, 0=desactivo, 1=activo
 4 : Máxima abscisa del osciloscopio
 5 : Máxima ordenada del osciloscopio
 6 : Máxima abscisa del plotter
 7 : Máxima ordenada del plotter
 8 : Abscisa del cursor virtual
 9 : Ordenada del cursor virtual
10 : Abscisa del cursor físico (dispositivo)
11 : Ordenada del cursor físico (dispositivo)
12 : Abscisa menor de la ventana
13 : Ordenada menor de la ventana
14 : Abscisa mayor de la ventana
```

15 : Ordenada mayor de la ventana
16 : Abscisa menor del puerto de visión
17 : Ordenada menor del puerto de visión
18 : Abscisa mayor del puerto de visión
19 : Ordenada mayor del puerto de visión
20 : Relación en abscisa puerto/ventana
21 : Relacibación en ordenada puerto/ventana
22 : 0=dentro de la ventana, 1=fuera de ella
23 : 0=sin error en el último comando, 1=con error

DAT_GRAF es función y responde con un número real representativo del parámetro correspondiente.

4.5 Definición de caracteres

Los caracteres son definidos en una rejilla de 7*7, de modo vectorial empleando vectores cortos de proyecciones X y Y de hasta 7 unidades de longitud. Las coordenadas varían entre 0 y 7 en cada eje y sólo se emplean valores enteros. Esto hace posible codificar el vector en un solo byte empleando 3 bits por cada coordenada, un bit para indicar la posición de la pluma y otro más para indicar que el vector es el último trazo del carácter, este método está ampliamente documentado en "Principles of Computer Graphics" de Newman & Sproul. Cada carácter comienza en las coordenadas 0,0 y la coordenada del vector es la posición final del vector que se convierte en la inicial del siguiente vector. Las coordenadas son absolutas respecto a la rejilla.

El paquete de las rutinas de enlace permite la definición de 128 caracteres (del 0 al 127) y hasta 1024 vectores. Se tiene una lista de índices de acceso IND_CAR

que dirige el acceso a **TAB_CAR** que contiene los vectores. Se empleó el código ASCII para los caracteres normales entre 32 y 126 que incluye los caracteres alfabéticos en mayúsculas y minúsculas, los números y los símbolos especiales de escritura como #, !, @, ^, etc. Además en la última versión del archivo **SYSCAR** se definieron los caracteres griegos más empleados y otros símbolos especiales.

Se emplearon definiciones a espacios uniformes, o sea el último trazo es a 7,0 con la pluma levantada y las minúsculas tienen rasgos descendentes a un renglón. Empleando una escala de 3 en el trazo a osciloscopio, los caracteres son bastante legibles y para el plotter depende del tipo de pluma que se emplea. La tabla de los caracteres que quedaron definidos se muestra en la figura 4.1

El archivo con la definición de los caracteres emplea 1536 bytes o 3 bloques exactamente porque la carga se hace por bloques. El primer bloque tiene los índices de acceso en los primeros 256 bytes, la segunda mitad del primer bloque está desperdiciada. Los dos bloques restantes, son los vectores (hasta 1024 vectores) comprimidos. Este archivo fue leído en bloque porque el tiempo de carga se reduce cerca de 10 veces comparada a una carga convencional empleando la lectura convencional con GET.

La creación del archivo se hace por medio del programa **GENTAB** que obtiene datos del archivo de texto **TABCAR** y genera un archivo **SYSCAR2** cuyo nombre puede cambiarse

posteriormente a *:SYSCAR o bien leerlo independientemente llamando a LEE_CAR. Este programa hace la compactación de los vectores y la generación de los índices de acceso.

El archivo TABCAR tiene el siguiente formato:

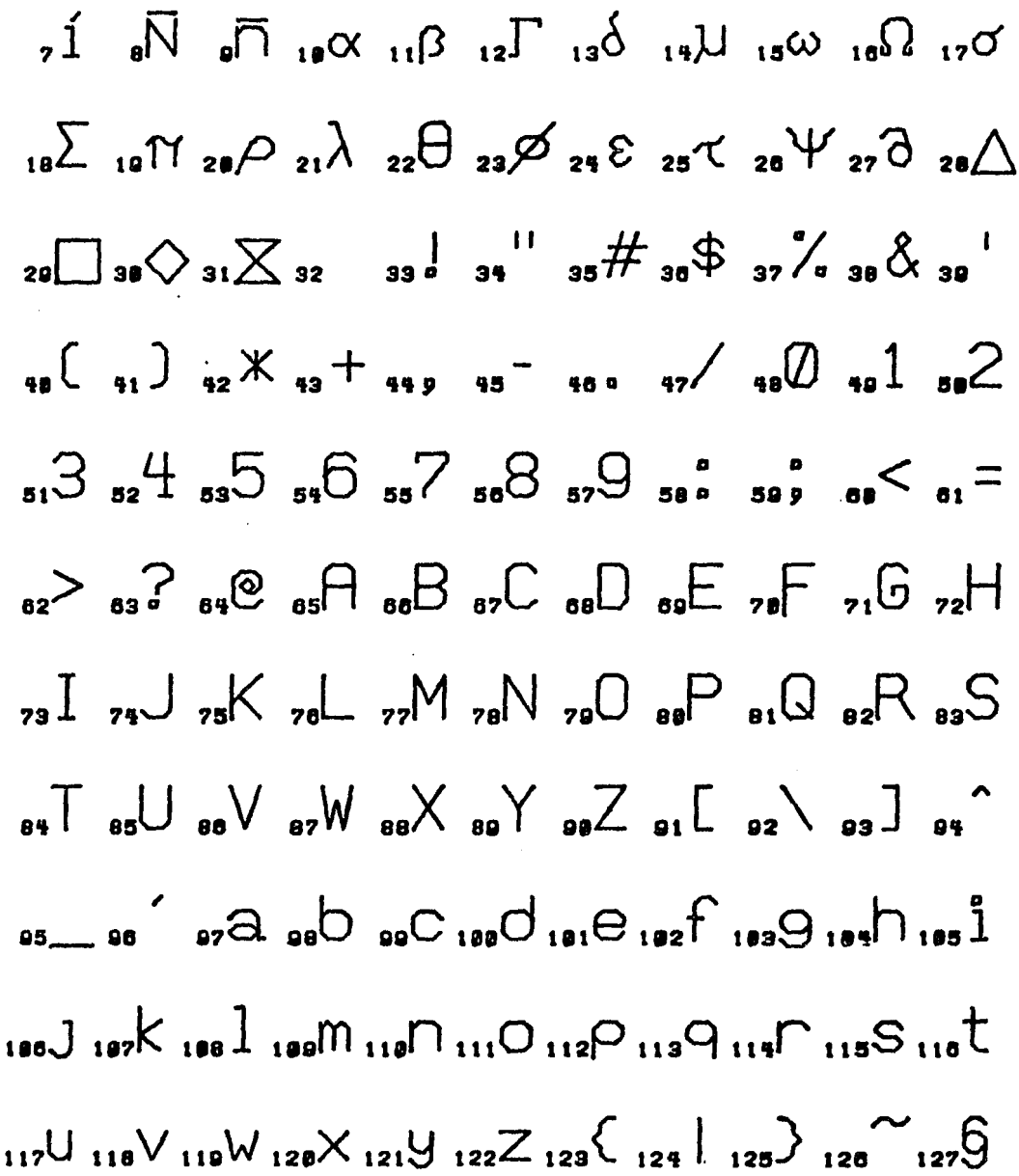
```
<ASCII del primer carácter definido>
<ASCII del último carácter definido>

    por cada carácter
    <mensaje de control>
        por cada vector
        <coord.X> <coord.Y> <pluma> <EOC>
```

Las coordenadas son números entre 0 y 7 que marcan el punto final del vector, pluma es la posición de la pluma, 0 es bajada y 1 levantada, EOC marca el último vector del cara cter, 0 es falso y 1 es verdadero. El mensaje es una cadena de caracteres para mantener el control de los vectores lei dos y para efectos de edición.

Al momento de correr el programa GENTAB escribe los códigos del primer y último carácter definido en la pantalla y escribe el mensaje de control seguido del índice de acceso correspondiente al carácter. Para todos los caracteres no definidos, se asigna el índice de acceso del carácter 32 correspondiente al espacio en ASCII por lo tanto el carácter 32 siempre debe quedar definido.

Gracias a estas facilidades, es factible redefinir la tabla de caracteres para adaptarla a las necesidades individuales de cada usuario (caracteres especiales, itálicos, en espacios proporcionales, etc.).

Figura 4.1 Tabla de caracteres



4.6 Uso de las rutinas de enlace.

Las rutinas de enlace quedaron incluidas en el archivo de paquetes de biblioteca *:SYSTEM.LIBRARY y para llamarlo se pone la siguiente declaración en el programa de aplicación:

```
USES GRAFICAS;
```

que al compilar, carga las definiciones de los procedimientos, funciones, tipos de variables y variables previamente mencionadas y que se muestran en los listados.

Las definiciones restantes por describir son **DEF_CAR**, que es el archivo de la definición de caracteres que por su forma peculiar de carga, es definida como tipo **FILE** sin tipo de registro; y **COM_GRAF** que es el tipo de registro del archivo de eco.

Al momento de correr, se lee la definición de caracteres de *:SYSCAR y se selecciona al osciloscopio como dispositivo de salida sin eco de comandos.

Este paquete emplea 6000 bytes de memoria principal a diferencia del paquete **TURTLEGRAPHICS** del sistema UCSD que emplea 18000 bytes, esto permite programas de aplicación más grandes sin necesidad de hacer *overlays*.

El sistema de bibliotecas requiere que el programador del paquete defina explícitamente las rutinas, variables y tipos de variable a las que el usuario del paquete tendrá acceso, dentro de un bloque llamado *INTERFACE* guardado en forma literal y vaciado en forma de texto al programa de aplicación. Por esta razón, se hicieron dos bloques *INTERFACE*, uno completamente documentado que se presenta en el apéndice de listados y otro compacto sin comentarios, que finalmente se incluyó en la versión de trabajo del paquete para reducir el consumo de espacio en disco y el tiempo de compilación de los programas de aplicación.

Capítulo 5

Rutinas de aplicación

5-1 Sistema de representación gráfica de datos.

Se programó un sistema para la representación gráfica de datos en dos dimensiones, que emplea el paquete gráfico descrito en el capítulo anterior. Este sistema quedó instalado en el archivo *:SYSTEM.LIBRARY como paquete de biblioteca que puede ser llamado con la palabra reservada **USES**, llamando a **GR2**.

El paquete emplea el sistema **GRAFICAS** y también emplea el paquete de funciones trascendentales **TRANSCENDS** por lo tanto la declaración debe ser:

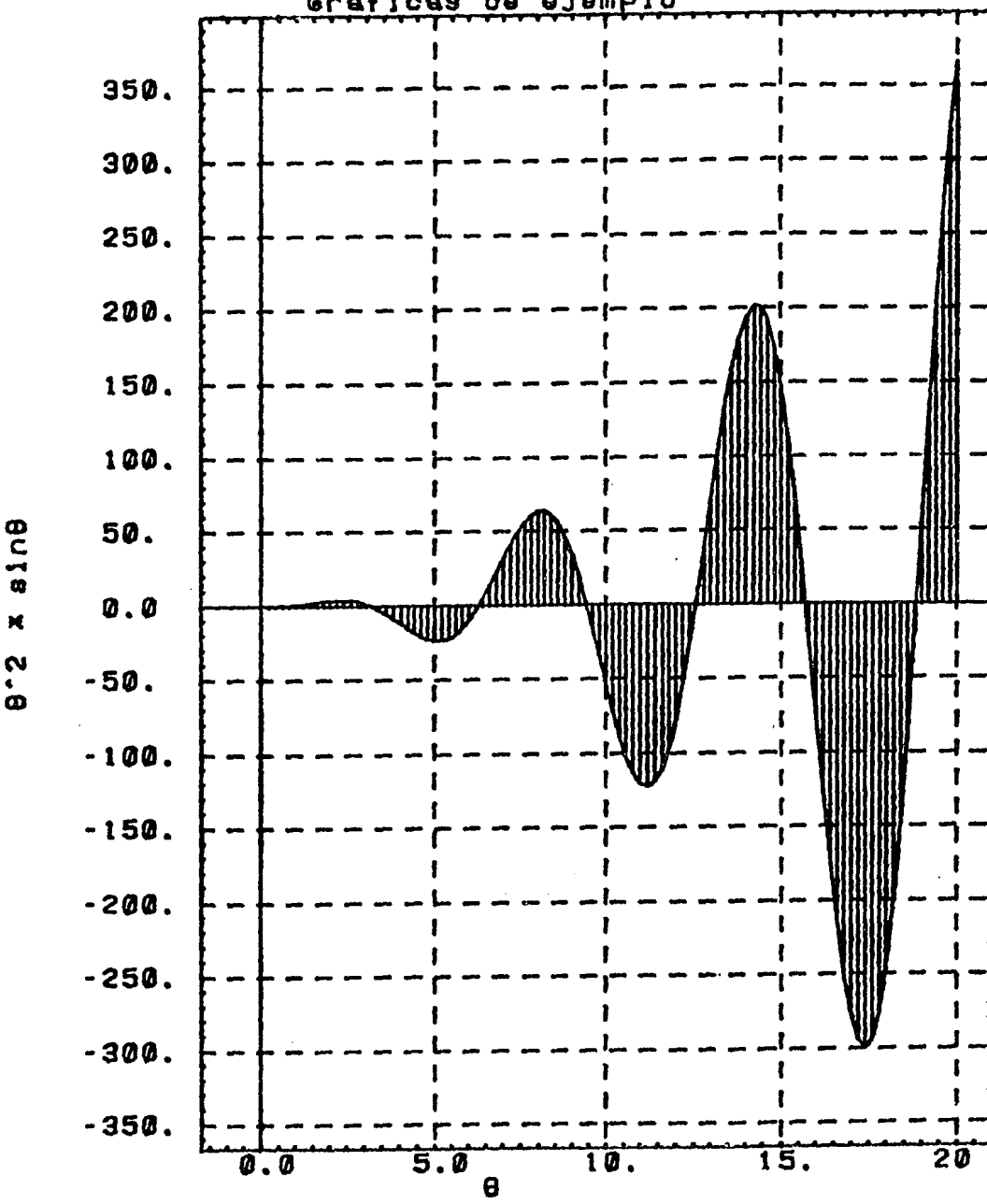
```
USES GRAFICAS, TRANSCENDS, GR2;
```

GR2 pretende simplificar el trazado de datos y por esta razón sólo tiene dos rutinas desde el punto de vista del usuario, **DIREC** y **GRAF2**.

DIREC es una función escrita en ensamblador, cuyo objetivo es obtener la dirección de cualquier variable del programa de aplicación que emplea a **GR2**. La dirección la regresa en forma de un entero y la llamada lleva por parámetro la variable en cuestión.

Figura 5.1

Graficas de ejemplo



Esta rutina es por lo tanto una implementación local, no solo desde el punto de vista de UCSD Pascal sino desde el punto de vista del microprocesador 6502 que contiene la APPLE. La técnica usada fue hacer el paso de la variable por dirección usando la palabra reservada **VAR**. UCSD Pascal tiene en sus reglas de producción la variante al Pascal común de permitir el paso de parámetros por dirección sin verificación de tipo. Esta dirección es entonces regresada a través del *stack* como un entero con signo, de 16 bits.

GRAF2(XAPT,YAPT,N,TGRAF,TLIN,OPCION) es una rutina escrita en Pascal que efectúa el trazado de la serie de datos. El parámetro **XAPT** es la dirección del arreglo con los datos de las abscisas (si existen), el parámetro **YAPT** es la dirección del arreglo de la serie de datos de las ordenadas, que deben de existir y **N** es el número de datos que se trazarán.

XAPT y **YAPT** se obtienen aplicando la función **DIREC** a los arreglos unidimensionales que contienen las series de datos que deberán ser de tipo real y pueden ser de cualquier longitud, restringidos únicamente por la memoria disponible. Es factible que la serie X no exista si se desea que Y sea trazada contra su ordinalidad; en este caso la dirección que se debe pasar como la serie X es irrelevante.

Otra opción respecto al paso de las series es que no inicien en el primer dato del vector. Normalmente al llamar a **DIREC**, el arreglo no va subindizado, al subindizar

se provee de un punto de inicio que puede ser diferente al primer dato. Manejando los arreglos con cuidado, es factible emplear arreglos multidimensionales y obtener flexibilidad en el trazado de los datos.

No se realiza verificación respecto a la longitud declarada de los vectores, ni si en efecto son vectores, por tanto la veracidad de la existencia de X, Y y N es responsabilidad del usuario, pero las consecuencias en general no pasarán de tener una gráfica equívoca.

Los tres parámetros restantes determinan las características de la gráfica deseada. **TGRAF** tiene definidos tres valores enteros válidos:

0 equivale a una gráfica con los puntos consecutivos unidos por una línea.

1 la gráfica será sólo los puntos determinados por las parejas de datos, sin unirlos.

2 para cada punto (x,y) se traza una línea desde (x,0) a (x,y); es útil para funciones discretas que se trazan con impulsos, o para lograr efectos de sombreado.

Cualquier valor fuera de los tres mencionados, no generará gráfica pero el uso de estos valores adicionales está en la prefijación de las características del área de trazado (ventana), cuando se desea hacerlo de modo manual, pues el sistema lo puede hacer de modo automático.

TLIN es el tipo de línea que se empleará en los casos de TGRAF 0 y 2, y está restringido a los tipos de línea que GRAFICAS maneja, esto es, enteros entre 0 y 127.

OPCION es un entero en el cual se agrupan tres parámetros booleanos como bits individuales; entonces los valores válidos son entre 0 y 7, para cualquier otro valor, sólo se emplearán los tres primeros bits.

Estos parámetros que maneja OPCION son:

bit 0 si la gráfica es X-Y o bien si es o-Y donde o es la ordinalidad. 0 indica X-Y y 1 indica o-Y.

bit 1 si es 0 se recalculan los parámetros, si es 1 se emplean los anteriores.

bit 2 si es 0 se traza la malla o rejilla, si es 1 no se traza dicha malla.

La siguiente tabla es un resumen de los valores que puede tomar OPCION:

Valor	X-Y,o-Y	Parámetros	Malla
0	X-Y	nuevos	se traza
1	o-Y	nuevos	se traza
2	X-Y	anteriores	se traza
3	o-Y	anteriores	se traza
4	X-Y	nuevos	no se traza
5	o-Y	nuevos	no se traza
6	X-Y	anteriores	no se traza
7	o-Y	anteriores	no se traza

Los parámetros que el sistema calcula automáticamente, corresponden a las esquinas de la ventana que abarque los datos que se deseen trazar. Cada vez que estos parámetros

son calculados, se traza un marco en la ventana, si el cero de alguna o de ambas coordenadas está contenido en la ventana se traza una línea correspondiente al eje, se trazan marcas de referencia; a veces llamadas *tics* y se escriben los títulos de los ejes y los títulos de la gráfica y las etiquetas numéricas de los ejes.

La gráfica se traza en el puerto de visión que esté definido,y se recomienda que al menos sea de 1000x1000 pixeles porque hay zonas del puerto que son empleadas para el etiquetado de los ejes y los títulos de la gráfica; si el tamaño del puerto es inferior a 1000x1000 la zona reservada para la gráfica resultaría reducida en comparación a la destinada a etiquetas que es fija en ciertas dimensiones.

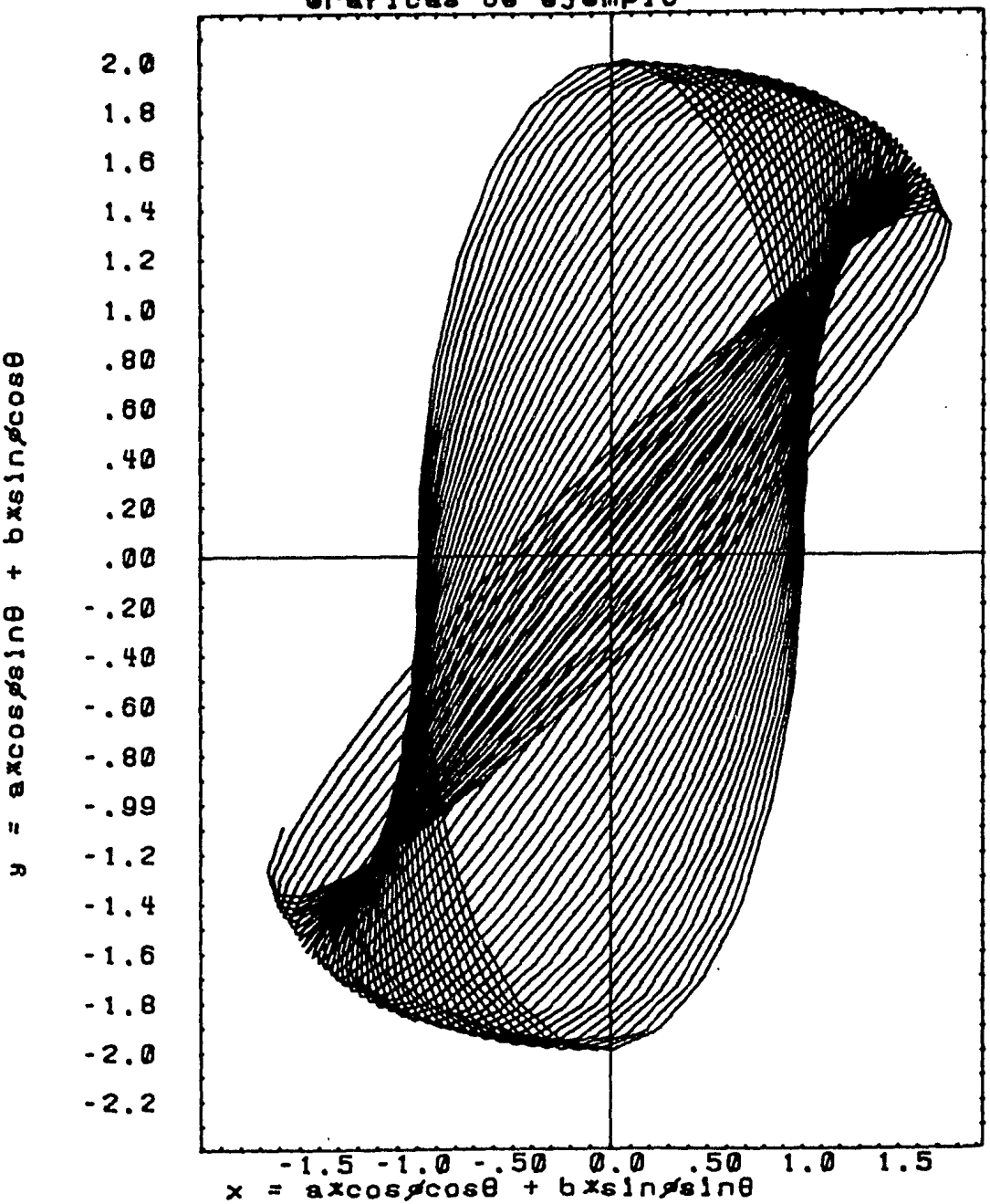
Para seleccionar el puerto de visión a emplear, se llama a la rutina **SEL_PUERTO** de la biblioteca **GRAFICAS**. Se podría entonces trazar hasta 4 gráficas en una pantalla u hoja del plotter, una en cada cuadrante.

La selección del dispositivo de salida se hace a través de **INI_GRAF** y como se mencionó en el capítulo anterior, automáticamente se selecciona la pantalla del osciloscopio como dispositivo de salida.

Los títulos que escribe en la gráfica son cuatro, para el eje X, eje Y, un título primario de la gráfica que se escribe con letras grandes y un título secundario en letra pequeña. El título del eje Y se escribe verticalmente y los títulos primario, secundario y dl eje X se escriben horizontalmente.

Figura 5.2

Graficas de ejemplo



Los títulos siempre aparecen centrados y su longitud máxima depende del puerto seleccionado, pero para el puerto mínimo recomendado, el título primario puede tener hasta 25 caracteres, y para los tres restantes hasta 35 caracteres.

Los títulos que se deseen escribir son capturados en cuatro variables tipo *string*; **TITULOX** para el eje X, **TITULOY** para el eje Y, **TPRIMARIO** para el título primario de la gráfica y **TSECUNDARIO** para el título secundario de la gráfica.

La posición de las etiquetas es calculada para que sean múltiplos de 2, 5 o 10 con el objeto de facilitar su lectura. Estas etiquetas tienen 2 dígitos, punto decimal y signo. Si la etiqueta rebasa estos límites, se hace una escala para el eje infractor y la magnitud del escalamiento se muestra, o bien en la esquina superior izquierda; para el eje Y, o en la esquina inferior derecha para el eje X, junto a una letra E.

Para el caso en que los datos tengan una componente de directa dos o más órdenes de magnitud mayor al rango dinámico de la componente de alterna, se efectúa una traslación para el etiquetado, indicando el valor de la traslación en la misma zona de las escalas, con una letra A.

Las figuras 5.1 y 5.2 muestran algunas gráficas generadas por el sistema.

5-2 Programa de representación gráfica en tres dimensiones.

La idea que motivó la creación de este programa, fue la de tener una herramienta que permitiera visualizar en forma fácil objetos en tres dimensiones.

La representación de objetos en tres dimensiones puede efectuarse de diversas maneras; se puede definir sólo el "esqueleto" de la figura en la representación conocida como armazón (*wireframe*) que es la más simple, o irle añadiendo características como líneas y caras ocultas (cuerpos sólidos), texturas, sombras, etc.

Debido a restricciones de capacidad de cómputo de la microcomputadora que se empleó, se optó por la representación más simple, la de armazón. En esta representación, se definen sólo segmentos de rectas para lo cual se deben definir los extremos de dichos segmentos.

El programa maneja un solo objeto a la vez aunque es factible hacer composiciones, y la definición de este objeto queda almacenada en disco, en un archivo de tipo real cuya estructura es la siguiente:

```
<número de puntos>
     para cada punto
          <coordenada X>
          <coordenada Y>
          <coordenada Z>

<número de líneas>
     para cada segmento de recta
          <punto inicial>
          <punto final>
          <tipo de línea>
```

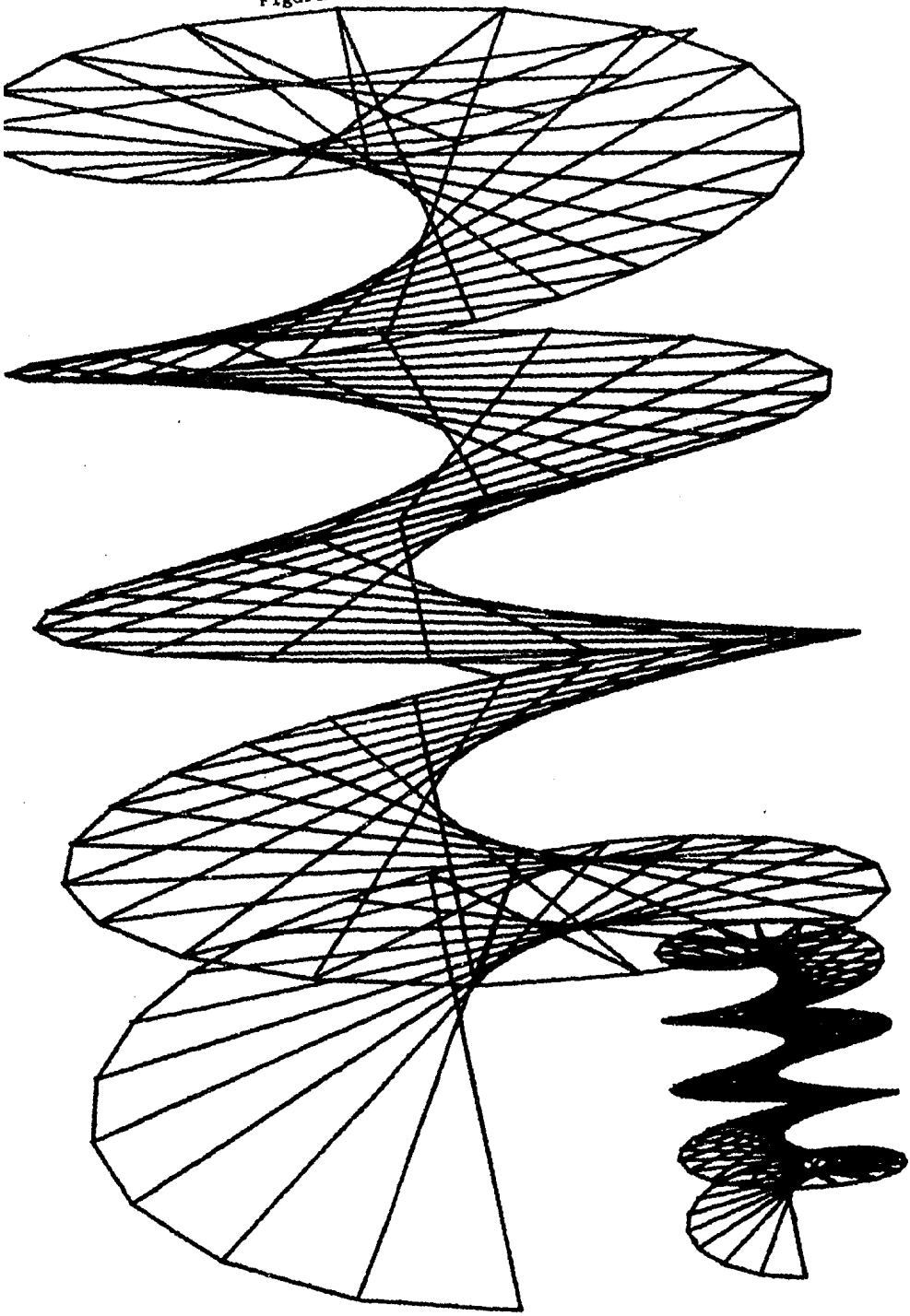
Los puntos se enumeran a partir de 1 y el tipo de línea corresponde a los tipos de línea descritos en el capítulo anterior. Por ejemplo un objeto con una sola línea tipo 10 entre las coordenadas (1,2,3) y (4,5,6) se representaría en el archivo como sigue:

2.0 1.0 2.0 3.0 4.0 5.0 6.0 1.0 1.0 2.0 10.0

El programa se maneja por menús y se ejecuta desde el menú principal del sistema operativo. El nombre del archivo en que reside es **GR3**. Al entrar, aparece un menú que indica si se desea salvar, recuperar o mantener un objeto. **SALVAR** significa dar la definición y nombre del archivo en que residirá el objeto que queremos ver; se permite la captura más no la edición de los datos. **RECUPERAR** lee el nombre del archivo con la definición del objeto. Y **MANTENER** significa retener el nombre del archivo con la última definición tratada.

El siguiente menú se refiere a las salidas y tiene cinco opciones, osciloscopio con y sin eco; plotter con y sin eco; y mantener la salida actual. Si se pide eco, se leerá el nombre del archivo al que se desea se reflejen los comandos.

Figura 5.3



A continuación se presenta el menú de las transformaciones del objeto. Las transformaciones permitidas son: rotaciones, traslaciones, escalamiento, y cualquier transformación que se pueda representar con matrices homogéneas.

Como ya se mencionó, se emplearon coordenadas homogéneas para tener transformaciones lineales, poder revertir los procesos y poder concatenar las transformaciones para almacenarlas en una sola matriz.

Las rotaciones son en contra de las manecillas del reloj (sistema derecho) y el ángulo está dado en grados. Se permiten rotaciones alrededor de los ejes coordenados y alrededor de cualquier eje, dando las componentes de un vector. Las traslaciones están dadas en unidades en las que fue definido el objeto.

El siguiente menú permite realizar las definiciones y transformaciones desde el punto de vista del usuario. Se permite hacer rotaciones, traslaciones, efectuar perspectiva de hasta tres puntos de fuga y dar la definición de ventana y puerto, si se desean alterar. Las dos opciones restantes son iniciar la matriz de transformación del usuario y salir del menú.

De nuevo, se emplean las transformaciones homogéneas y las rotaciones son en grados y contra las manecillas del reloj. La perspectiva se da con las inversas de las distancias focales en X, Y y Z. Como el usuario

normalmente ve hacia Z-, lo más normal será pasar de infinito (0) a 50 unidades (0.02) que da una perspectiva natural. Este efecto fue logrado gracias a características de las matrices de transformación homogénea porque permiten una cuarta coordenada de escalamiento global y el método pesa cada coordenada contra su distancia focal para dar el efecto de deformación por alejamiento.

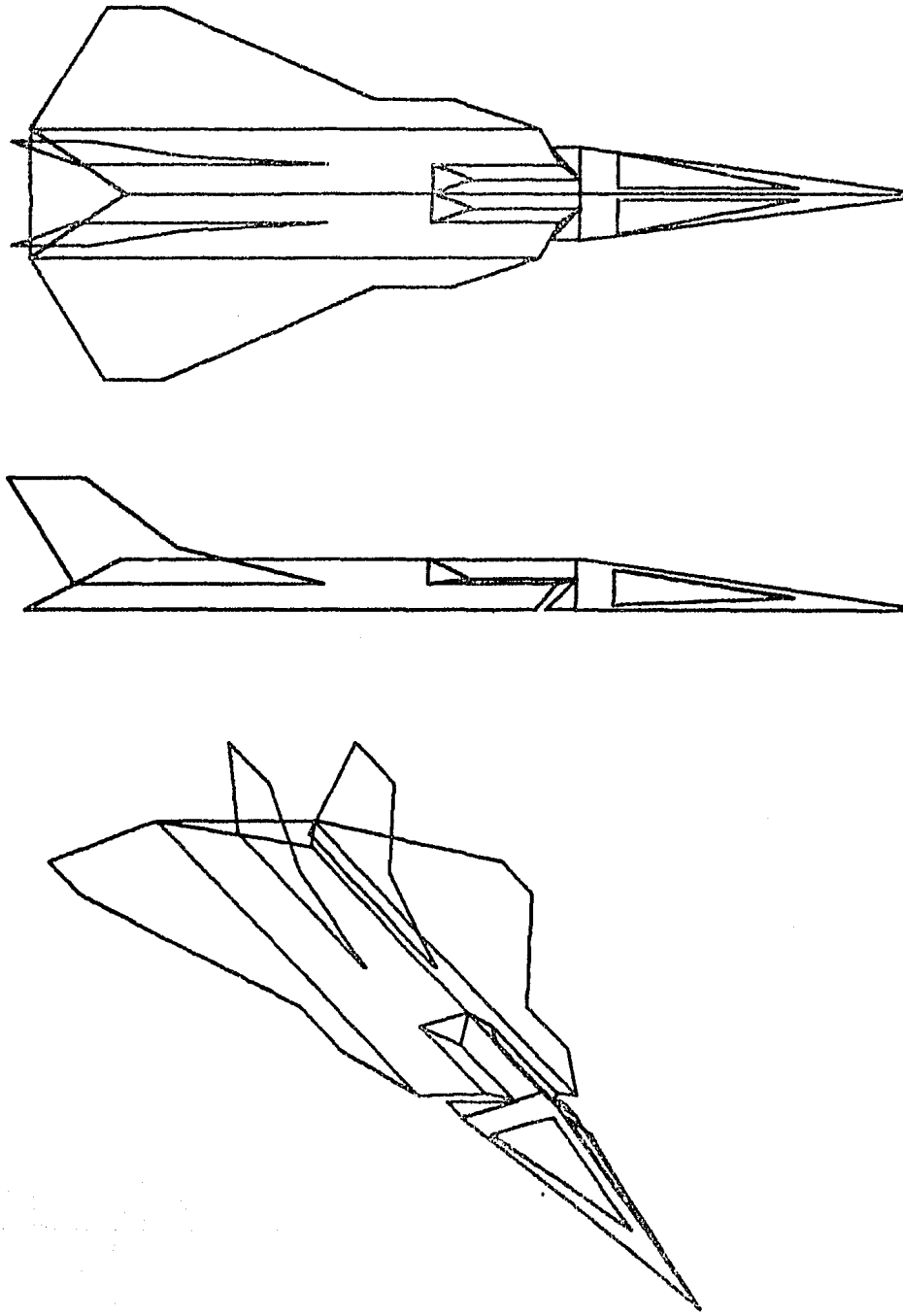
Al salir de este menú, comienza la generación de la gráfica. Lo primero que hace el programa es cargar todos los puntos y mapearlos a la ventana, las razones son por el consumo de memoria y por el tiempo de procesamiento, un punto es usualmente el extremo de más de un segmento de recta y al mapearlo, sólo se requieren dos coordenadas y no tres.

A continuación se comienza a leer los segmentos de recta que forman a la figura y simultáneamente se efectúan los trazos. Al terminar la gráfica aparece el último menú que tiene las opciones de continuar, borrar y salirse del programa.

CONTINUAR significa regresar de nuevo al primer menú con las matrices y definiciones que se han creado. **BORRAR** es borrar las definiciones (matrices), cerrar los dispositivos gráficos y regresar al primer menú. **SALIR** del programa no cierra el dispositivo gráfico.

Dos gráficas creadas con el programa aparecen en las figuras 5.3 y 5.4.

Figura 5.4



Capítulo 6.

Manual de Uso

6-1 Instalación de la Tarjeta

La tarjeta debe ser instalada en la ranura marcada con el número 3 dentro de la microcomputadora APPLE II, por restricciones de espacio, no puede haber alguna tarjeta en la ranura 2.

El cable plano multicolor tiene en su extremo un conector DB25 macho, éste se debe conectar a la entrada de señales remotas del plotter, que se localiza en la parte de atrás del osciloscopio. El otro cable, es gris y termina en un DB25 hembra que se debe conectar al adaptador para la entrada del plotter.

Los conectores RCA instalados en la placa que llega al exterior de la computadora, tienen las señales X y Y para el osciloscopio, en la salida superior e inferior respectivamente. Se usarán en caso que la señal mandada a través del cable plano no sea de una calidad aceptable.

Sólo el osciloscopio debe estar encendido antes de iniciar una sesión de generación de gráficas, pues hay procesos automáticos que corren al llamar a la biblioteca GRAFICAS.

6-2 Uso de las rutinas de enlace

Para emplear las rutinas básicas de enlace, se requiere del conocimiento previo del lenguaje Pascal y se debe emplear la siguiente declaración en la aplicación:

```
USES GRAFICAS;
```

que cargará las rutinas en el programa de aplicación.

Se definen los siguientes tipos de variables globales:

```
BYTE = 0..255;
COM_GRAF = PACKED RECORD
     COMANDO : BYTE;
     PARAM1, PARAM2 : INTEGER;
     END;
DISPOSITIVO = ( OSC, PLT );
ORIENTACION = ( ARR, DER, ABJ, IZQ );
PUNTO = RECORD
     X, Y : REAL;
     END;
```

Se definen las siguientes variables en forma global:

```
CM : PUNTO;
```

tiene el cursor virtual.

```
NOM_AR_ECO : STRING[20];
```

tiene el nombre del archivo de eco.

DEF_CAR : FILE

es el archivo de definición de caracteres.

ECO : PACKED FILE OF COM_GRAF

es el archivo del eco de comandos gráficos.

Se definen las siguientes rutinas de generación de elementos gráficos:

PROCEDURE **PLM_ABJ**;

indica que se desea bajar la pluma.

PROCEDURE **PLM_ARR**;

indica que se desea subir la pluma.

PROCEDURE **LINEA_HACIA**(X,Y:REAL);

mueve la pluma desde la posición actual a X,Y con la última posición de la pluma activa.

PROCEDURE **LINEA**(X1,Y1,X2,Y2:REAL);

mueve la pluma levantada hacia X1,Y1 y de ahí hacia X2,Y2 con la última posición de la pluma indicada.

PROCEDURE **MUEVE_HACIA**(X,Y:REAL);

mueve la pluma levantada desde la posición actual hacia X,Y.

PROCEDURE **ESC_STR**(STR:STRING;ESC:INTEGER;ORIENTA:ORIENTACION);

escribe una cadena de caracteres STR desde la posición del cursor con escala ESC y orientación ORIENTA.

PROCEDURE **ESC_CAR**(CAR:CHAR;ESC:INTEGER,ORIENTA:ORIENTACION);

escribe el carácter CAR en la posición del cursor, a la escala ESC, y orientación ORIENTA.

PROCEDURE **SEL_TIPO**(T:INTEGER);

selecciona el tipo de línea T.

Están definidas las siguientes rutinas de manejo de dispositivo:

PROCEDURE **INI_GRAF**(DISP:DISPOSITIVO,B_ECO:BOOLEAN);

selecciona e inicia el dispositivo DISP como salida y activa o inhibe el eco de comandos gráficos.

PROCEDURE **TERM_GRAF**(BORRA:BOOLEAN);

cierra si es posible, el archivo de eco y dependiendo del parámetro BORRA, hace un proceso de borrado sobre el dispositivo activo.

PROCEDURE **SEL_VENTANA**(X1,Y1,X2,Y2:REAL);

selecciona la ventana de representación con los márgenes X1,Y1,X2,Y2; como margen izquierdo, inferior, derecho y superior, respectivamente.

PROCEDURE **SEL_PUERTO**(X1,Y1,X2,Y2:REAL);

selecciona los márgenes límites de las coordenadas del dispositivo como X1.Y1,X2,Y2; de nuevo izquierda, abajo, derecha y arriba, respectivamente.

Por último están definidas las siguientes rutinas de manejo general del ambiente gráfico:

PROCEDURE **LEE_TAB_CAR**(AR_DEF_CAR:STRING);

lee la definición de los caracteres del archivo con nombre AR_DEF_CAR.

PROCEDURE **PROC_GRAF**;

hace la interpretación de los comandos de eco generados en el archivo con nombre NOM_AR_ECO.

FUNCTION **DAT_GRAF**(PREG:INTEGER):REAL;

regresa el valor del parámetro del ambiente gráfico con número PREG.

El paquete gráfico, automáticamente inicia con salida al osciloscopio, empleando toda la pantalla, con ventana igual al puerto, y con los caracteres de **#:SYSCAR**

6-3 Uso del Sistema de Representación Gráfica de Datos

Para usar el sistema de representación gráfica de datos, se debe tener conocimiento del uso de las rutinas de enlace o emplear los parámetros seleccionados automáticamente. Se debe incluir la siguiente declaración en el programa de aplicación:

```
USES GRAFICAS, TRANSCENDS, GR2;
```

Se definen cuatro variables de tipo STRING (cadena de caracteres) para las leyendas de la gráfica:

TPRIMARIO

título primario de la gráfica.

TSECUNDARIO

título secundario de la gráfica.

TITULOX

título del eje X.

TITULOY

título del eje Y.

Las rutinas que se ofrecen son las siguientes:

FUNCTION **DIREC**(VAR VARIABLE):INTEGER;

regresa la dirección en que se almacena la variable VARIABLE de cualquier tipo.

PROCEDURE **GRAF2**(XAPT,YAPT,N,TGRAF,TLIN,OPCION:INTEGER);

traza la gráfica tipo TGRAF de las series de N datos almacenados en vectores apuntados por XAPT y YAPT, con tipo de línea TLIN y las opciones indicadas en OPCION.

Para el manejo del dispositivo y detalles adicionales a la gráfica, se emplearán las rutinas de **GRAFICAS**.

Los valores para TGRAF de **GRAF2** son:

0 unir los puntos con líneas.

1 marcar sólo los puntos.

2 trazar impulsos del eje X al punto.

Los parámetros que maneja OPCION en **GRAF2** son:

bit0 0=X-Y, 1=o-Y.

bit1 0=nuevos parámetros, 1= mantener parámetros.

bit2 0=trazar malla, 1=no trazar la malla.

6-4 Uso del Programa de Representación gráfica en tres dimensiones

Este programa está en el archivo **GR3.CODE** y se ejecuta desde el sistema, se maneja a través de menús. A continuación se presentan los menús y una breve descripción.

-Menú de descripción del objeto

S)alvar R)ecuperar M)antener

Sirve para declarar el archivo donde reside la definición del objeto. La opción S activa un pequeño sistema de captura de definiciones.

-Menú de descripción de salida

A> Osciloscopio sin eco

B> Osciloscopio con eco

C> Plotter sin eco

D> Plotter con eco

E> Mantener

Declara el dispositivo que se desea como salida. Básicamente llama a **INI_GRAF** de **GRAFICAS.**

-Menú de transformaciones del objeto

R)ot E)sc L)oc I)nit O)tro S)alida

Se define la matriz homogénea de transformaciones al objeto, se permiten rotaciones, escalamientos, traslaciones y matrices especiales.

-Submenú de rotaciones

X Y Z V)ec S)alida

Permite indicar el tipo de rotación, sobre el eje X, Y, Z o cualquier vector, dadas sus componentes, a un ángulo cualquiera en grados sexagesimales y en sentido derecho.

-Menú de transformaciones al observador

F)oco R)ot L)oc I)nit V)ent P)uerto S)alida

Indica las características de vista del usuario, se permiten definir perspectivas, rotaciones, traslaciones, la ventana y el puerto de visión.

-Menú de control general

B)orra C)ontinúa S)alida

Borra las matrices definidas y la ventana, las mantiene o permite salir del programa.

Capitulo 7

Conclusiones

Se logró desarrollar una tarjeta de interface para comunicar a la microcomputadora APPLE II con un osciloscopio Tektronix 611 y con un plotter Houston Instruments DP-1. Ambos dispositivos generan gráficas de alta calidad y su uso representa una inversión muy baja debido a que la adquisición de los instrumentos se realizó hace más de 15 años.

La tarjeta de interface se elaboró a un costo muy bajo y en un lapso de tiempo corto debido principalmente a que no se emplearon microprocesadores, lo cual hubiera elevado innecesariamete los costos y muy probablemente, debido a que los desarrollos con microprocesadores requieren una depuración más completa y extensa, el tiempo de puesta a punto se hubiera prolongado.

Las rutinas más básicas fueron programadas en lenguaje ensamblador por razones de velocidad de ejecución y por el uso de las interrupciones. Esto no se podria realizar en Pascal porque por un lado, no se tiene acceso al sistema de interrupciones, y por otro lado, el compilador no genera

código nativo, sino que genera un código que se debe interpretar, esto obliga a codificar las rutinas críticas en el tiempo, en ensamblador. Las rutinas de trazado de líneas en el graficador incremental traslapan los cálculos con el movimiento efectivo de los motores de paso, por otro lado, una línea trazada en el osciloscopio de esquina a esquina, tarda menos de 0.3 segundos.

Para poder emplear estas rutinas básicas se hizo un paquete de biblioteca con las rutinas de enlace que añaden los conceptos de ventanas, puertos de visión, recorte, escritura de caracteres y archivos de eco de comandos gráficos. Emplea un sistema de coordenadas virtuales derecho, que evitan al usuario la necesidad de escalar y trasladar puntos para adaptar los programas a los dispositivos.

Para poder cubrir un cuadro de aplicaciones generales, se elaboró un paquete de biblioteca con rutinas para trazar gráficas de datos en dos dimensiones, y un programa para manipular objetos tridimensionales.

Para poder hacer flexible el paquete para trazar gráficas de datos, era necesario poder pasar arreglos de dimensión variable, cosa ilegal en Pascal estandar, pero que se podía implementar con rutinas en ensamblador en el UCSD Pascal.

El programa de manipulación de objetos tridimensionales emplea matrices homogéneas para lograr efectuar rotaciones, traslaciones, escalamientos y efectos de perspectiva. Los objetos que maneja, emplean definición de tipo armazón o *wireframe* y se almacena la estructura en archivos de disco, de este modo, un programa externo puede generar el objeto sobre un archivo y posteriormente con ayuda de este programa, se puede manipular y visualizar el objeto, e incluso generar copias en el plotter o graficador incremental.

Entre las aplicaciones para este programa se encuentran, visualización de funciones o datos de tres variables y modelado de objetos.

En resumen, se logró integrar un sistema gráfico alrededor de una microcomputadora, con muy bajos costos, con resultados de alta calidad y que permiten mayor flexibilidad a nivel de programación con respecto a lo que se contaba anteriormente. **GRAFICAS** y **GR2** son conjuntos de rutinas y como tales, sus aplicaciones gráficas no están limitadas, ma 's que por las habilidades de programación de los usuarios. GR3 es un programa que puede recibir entrada por programas externos lo que confirma el enfoque del sistema, como herramienta para la programación.

Apéndice A

Glosario

Bit Unidad de información binaria.

Buffer Dispositivo electrónico que amplifica corriente.

Byte Grupo de 8 bits.

Coordenadas Homogéneas Es una representación de un espacio n-dimensional en un espacion (n+1)-dimensional tal que las coordenadas del espacio n-dimensional son multiplicadas por un factor igual a la coordenada (n+1) que se llama "escala".

DB25 Conector en forma dela letra D que alberga 25 líneas en paralelo, usado en la norma RS232C.

Interrupción Evento externo a un procesador, asíncrono que provoca que dicho procesador comience la ejecución de una rutina predefinida llamada "rutina de servicio de interrupción".

Latch Dispositivo Electrónico capaz de almacenar un bit de información y es comandado por el nivel de una señal.

LSI *Large-Scale Integration* o Integración a gran escala. Circuitos digitales monolíticos con más de 1,000 y menos de 10,000 transistores.

Matriz Homogénea Matriz que representa una transformación dentro del espacio de las coordenadas homogéneas.

MSI *Medium-Scale Integration* o Integracion a escala media. Circuitos digitales monolíticos con más de 100 y menos de 1,000 transistores.

Osciloscopio de memoria Dispositivo de salida gráfica en el cual un haz de electrones es dirigido por campos eléctricos hacia una placa fosforescente de caracteristicas capacitivas en el que se puede almacenar y mostrar la traza que el haz de electrones deja a su paso.

PIA Es un circuito con puertos paralelos para las familias de microprocesadores 6800 y 6500. Acrónimo de *Peripherial Interface Adapter* o Adaptador de Interfaz para Perifricos.

Pixel Unidad mínima de información gráfica discreta.

Plotter (Graficador Incremental X-Y) Es un dispositivo de salida gráfica mecánico que emplea motores para posicionar una pluma y efectuar trazos en papel.

Puerto de visión (*Viewport*) Porción del área de visualización del dispositivo gráfico a la cual se restringen o dirigen los comandos gráficos.

Raster Matriz rectangular en la que se registra la intensidad de cada punto de la imagen en la pantalla.

SSI *Small-Scale Integration* o Integración a escala pequeña. Circuitos digitales monolíticos con menos de 100 transistores.

TLA Acrónimo de *The Last Assembler* o El Ultimo Ensamblador.

UCSD p-System Sistema operativo desarrollado en la Universidad de California, San Diego para la máquina virtual P.

Ventana Porción del mundo virtual que el usuario desea representar a través del puerto de visión definido en el dispositivo gráfico.

Wireframe Modelo de objetos tridimensionales en la que aparecen delineados por algo similar a un armazón de alambre.

Apéndice B

Listados

```
 1   1:D   1 (*$L PRINTER:*)
 1   1:D   1 (*$S+*)
 1   1:D   1
 1   1:D   1 (*****************************************************************)
 1   1:D   1 (*                                                               *)
 1   1:D   1 (*      Este programa integra las rutinas para el paquete de     *)
 1   1:D   1 (*  biblioteca GRAFICAS que emplea los segmentos 23 y 24.        *)
 1   1:D   1 (*      El paquete de biblioteca se presenta como una unidad     *)
 1   1:D   1 (*  y sirve para el manejo del Osciloscopio TEKTRONIX  611 y     *)
 1   1:D   1 (*  del Graficador Incremental X-Y Houston Instruments  DP-1     *)
 1   1:D   1 (*  a  traves  de  la  tarjeta  de  acoplamiento  construida      *)
 1   1:D   1 (*  exprofeso.                                                   *)
 1   1:D   1 (*                                                               *)
 1   1:D   1 (*                         Realizo:       Miguel Chin Auyon      *)
 1   1:D   1 (*                                                               *)
 1   1:D   1 (*                         Fecha de creacion     :  5-mar-85     *)
 1   1:D   1 (*                         Fecha de actualizacion: 11-mar-85     *)
 1   1:D   1 (*                                                               *)
 1   1:D   1 (*****************************************************************)
 1   1:D   1
 1   1:D   1
 1   1:D   1 UNIT  GRAFICAS;
 1   1:D   1
 1   1:D   1 INTRINSIC CODE 23
23   1:D   1           DATA 24;
23   1:D   1
23   1:D   1
23   1:D   1 (*****************************************************************)
23   1:D   1 (*                                                               *)
23   1:D   1 (*      Bloque de declaraciones publicas del paquete             *)
23   1:D   1 (*                                                               *)
23   1:D   1 (*****************************************************************)
23   1:D   1
23   1:D   1
23   1:D   1 INTERFACE
23   1:D   1
23   1:D   1
23   1:D   1 (*****************************************************************)
23   1:D   1
23   1:D   1    TYPE
23   1:D   1
23   1:D   1       BYTE=0..255;
23   1:D   1
23   1:D   1       COM_GRAF=PACKED RECORD
23   1:D   1          COMANDO          (* Almacena el comando de la instruc-  *)
23   1:D   1             :BYTE;        (* cion del archivo grafico            *)
23   1:D   1          PARAM1,          (* Parametros de la instruccion del    *)
23   1:D   1          PARAM2           (* archivo grafico de eco              *)
23   1:D   1             :INTEGER;
23   1:D   1          END;
23   1:D   1
23   1:D   1       DISPOSITIVO         (* Dispositivos graficos disponibles   *)
23   1:D   1          = ( OSC,         (* Osciloscopio de Memoria             *)
23   1:D   1              PLT          (* Graficador Incremental X-Y          *)
23   1:D   1            );
```

```
 56 23   1:0     1 (*$P+*)
 57 23   1:0     1       ORIENTACION       (* Posibles orientaciones del texto   *)
 58 23   1:0     1          = ( ARR,       (* Hacia arriba                        *)
 59 23   1:0     1              DER,       (* Hacia la derecha                    *)
 60 23   1:0     1              ABJ,       (* Hacia abajo                         *)
 61 23   1:0     1              IZQ        (* Hacia la izquierda                  *)
 62 23   1:0     1            );
 63 23   1:0     1
 64 23   1:0     1       PUNTO=RECORD
 65 23   1:0     1          X,             (* Abscisa y ordenada de las coordena- *)
 66 23   1:0     1          Y              (* das del mundo usuario               *)
 67 23   1:0     1             :REAL;
 68 23   1:0     1          END;
 69 23   1:0     1
 70 23   1:0     1 (*************************************************************************)
 71 23   1:0     1
 72 23   1:0     1    VAR
 73 23   1:0     1
 74 23   1:0     1       CM                (* Cursor en coordenadas del usuario   *)
 75 23   1:0     1          :PUNTO;
 76 23   1:0     5
 77 23   1:0     5       NOM_AR_ECO        (* Nombre del archivo grafico del eco  *)
 78 23   1:0     5          :STRING[20];
 79 23   1:0    16
 80 23   1:0    16       DEF_CAR           (* Archivo que contiene la definicion  *)
 81 23   1:0    16          :FILE;         (* vectorial de los caracteres         *)
 82 23   1:0    56
 83 23   1:0    56       ECO               (* Archivo del eco de las instruccio-  *)
 84 23   1:0    56          :PACKED FILE OF COM_GRAF;      (*  nes graficas      *)
 85 23   1:0   359
 86 23   1:0   359 (*************************************************************************)
 87 23   1:0   359
 88 23   2:0     1    PROCEDURE PLM_ARR;           (* Sube la pluma               *)
 89 23   2:0     1
 90 23   3:0     1    PROCEDURE PLM_ABJ;           (* Baja la pluma               *)
 91 23   3:0     1
 92 23   1:0     1    PROCEDURE LINEA_HACIA        (* Traslada la pluma desde la  *)
 93 23   1:0     1                                 (* pos. actual, hasta [X,Y]    *)
 94 23   4:0     1       ( X,                      (* Abscisa del destino         *)
 95 23   4:0     1         Y                       (* Ordenada del destino        *)
 96 23   4:0     1           :REAL
 97 23   4:0     1       );
 98 23   4:0     5
 99 23   1:0     1    PROCEDURE LINEA              (* Mueve la pluma a [X1,Y1] y  *)
100 23   1:0     1                                 (* de ahi a [X2,Y2]            *)
101 23   5:0     1       ( X1,                     (* Abscisa del origen          *)
102 23   5:0     1         Y1,                     (* Ordenada del origen         *)
103 23   5:0     1         X2,                     (* Abscisa del destino         *)
104 23   5:0     1         Y2                      (* Ordenada del destino        *)
105 23   5:0     1           :REAL
106 23   5:0     1       );
107 23   5:0     9
108 23   1:0     1    PROCEDURE MUEVE_HACIA        (* Mueve la pluma hacia [X,Y]  *)
109 23   6:0     1       ( X,                      (* Abscisa del destino         *)
110 23   6:0     1         Y                       (* Ordenada del destino        *)
111 23   6:0     1           :REAL
112 23   6:0     1       );
```

```
113 23    6:D    5 (*$P*)
114 23    1:D    1   PROCEDURE ESC_STR            (* Escribe una cuerda de    *)
115 23    1:D    1                                (* caracteres                 *)
116 23    7:D    1       ( STR                    (* Texto a escribir           *)
117 23    7:D    1         :STRING;
118 23    7:D    2         ESC                    (* Escala de los caracteres   *)
119 23    7:D    2          :INTEGER;
120 23    7:D    3         ORIENTA                (* Orientacion del texto      *)
121 23    7:D    3          :ORIENTACION
122 23    7:D    3       );
123 23    7:D   45
124 23    1:D    1   PROCEDURE ESC_CAR            (* Escribe un solo caracter   *)
125 23    8:D    1      ( CAR                     (* Caracter a escribir        *)
126 23    8:D    1          :CHAR;
127 23    8:D    2         ESC                    (* Escala del caracter        *)
128 23    8:D    2          :INTEGER;
129 23    8:D    3         ORIENTA                (* Orientacion del caracter   *)
130 23    8:D    3          :ORIENTACION
131 23    8:D    3       );
132 23    8:D    4
133 23    1:D    1   PROCEDURE SEL_VENTANA        (* Selecciona los margenes de *)
134 23    1:D    1                                (* la ventana con su diagonal *)
135 23    9:D    1       ( X1,                    (* Margen izquierdo           *)
136 23    9:D    1         Y1,                    (* Margen inferior            *)
137 23    9:D    1         X2,                    (* Margen derecho             *)
138 23    9:D    1         Y2                     (* Margen superior            *)
139 23    9:D    1          :REAL
140 23    9:D    1       );
141 23    9:D    9
142 23    1:D    1   PROCEDURE SEL_PUERTO         (* Selecciona los margenes del *)
143 23    1:D    1                                (* puerto con su diagonal     *)
144 23   10:D    1       ( X1,                    (* Margen izquierdo           *)
145 23   10:D    1         Y1,                    (* Margen inferior            *)
146 23   10:D    1         X2,                    (* Margen derecho             *)
147 23   10:D    1         Y2                     (* Margen superior            *)
148 23   10:D    1          :REAL
149 23   10:D    1       );
150 23   10:D    9
151 23    1:D    1   PROCEDURE SEL_TIPO           (* Selecciona el tipo de linea *)
152 23   11:D    1      ( T                       (* Tipo seleccionado          *)
153 23   11:D    1          :INTEGER
154 23   11:D    1       );
155 23   11:D    2
156 23    1:D    1   PROCEDURE INI_GRAF           (* Inicia la transmision de   *)
157 23    1:D    1                                (* instrucciones al disposi-  *)
158 23    1:D    1                                (* tivo grafico               *)
159 23   12:D    1       ( DISPO                  (* Seleccion de dispositivo   *)
160 23   12:D    1         :DISPOSITIVO;
161 23   12:D    2         B_ECO                  (* Seleccion de eco a archivo *)
162 23   12:D    2          :BOOLEAN              (* grafico                    *)
163 23   12:D    2       );
164 23   12:D    3
165 23    1:D    1   PROCEDURE LEE_TAB_CAR       (* Lee la definicion de los   *)
166 23    1:D    1                                (* caracteres de un archivo   *)
167 23   13:D    1       ( AR_DEF_CAR             (* Nombre del archivo de defi- *)
168 23   13:D    1         :STRING                (* niciones                   *)
169 23   13:D    1       );
```

```
23   13:D    43 (*$P*)
23    1:D     1    PROCEDURE TERM_GRAF          (*  Si hubo eco, cierra el      *)
23    1:D     1                                 (* archivo.  Borra el OSC o     *)
23    1:D     1                                 (* cambia de hoja en PLT depen- *)
23    1:D     1                                 (* diendo cual esta abierto     *)
23   14:D     1       ( BORRA                   (*  TRUE si se desea borrar     *)
23   14:D     1         :BOOLEAN
23   14:D     1       );
23   14:D     2
23   15:D     1    PROCEDURE PROC_GRAF;         (*  Procesa un archivo de eco   *)
23   15:D     1
23    1:D     3    FUNCTION DAT_GRAF            (* Informa de datos del sistema *)
23   16:D     3       ( PREG                    (*  Parametro de inquisicion    *)
23   16:D     3         :INTEGER
23   16:D     3       ):REAL;
23   16:D     4
23   16:D     4
23   16:D     4 (*****************************************************************)
23   16:D     4
23   16:D     4
23    1:D     4 IMPLEMENTATION
23    1:D   359
23    1:D   359
23    1:D   359 (*****************************************************************)
23    1:D   359 (*                                                               *)
23    1:D   359 (*       Bloque de declaraciones privadas del paquete            *)
23    1:D   359 (*       y cuerpo de las rutinas publicas del mismo.             *)
23    1:D   359 (*                                                               *)
23    1:D   359 (*****************************************************************)
23    1:D   359
23    1:D   359 (*$I GRLIB01 *)
```

```
23   1:D   359 (*$P*)
23   1:D   359 (*********************************************************************)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*      Archivo GRLIB01  se incluye en GRLIB00                       *)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*                                   Realizo:                        *)
23   1:D   359 (*                                           Miguel Chin Auyon       *)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*                          Fecha de creacion     :  5-mar-85        *)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*                          Fecha de actualizacion:  13-mar-85       *)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*********************************************************************)
23   1:D   359
23   1:D   359 (*********************************************************************)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*      Bloque de declaraciones privadas                             *)
23   1:D   359 (*                                                                   *)
23   1:D   359 (*********************************************************************)
23   1:D   359
23   1:D   359 CONST
23   1:D   359
23   1:D   359    OSCXMAX=2047;          (* Maxima abscisa en el osciloscopio    *)
23   1:D   359
23   1:D   359    OSCYMAX=2047;          (* Maxima ordenada en el osciloscopio   *)
23   1:D   359
23   1:D   359    PLTXMAX=2199;          (* Maxima abscisa en el plotter         *)
23   1:D   359
23   1:D   359    PLTYMAX=2199;          (* Maxima ordenada en el plotter        *)
23   1:D   359
23   1:D   359 (*********************************************************************)
23   1:D   359
23   1:D   359 VAR
23   1:D   359
23   1:D   359    CD:RECORD              (* Coordenadas fisicas del dispositivo *)
23   1:D   359       X,
23   1:D   359       Y
23   1:D   359         :INTEGER;
23   1:D   359       END;
23   1:D   361
23   1:D   361    TIPO,                  (* Tipo de linea actual                 *)
23   1:D   361    PLM,                   (* Posicion actual de la pluma          *)
23   1:D   361    DISP,                  (* Dispositivo seleccionado             *)
23   1:D   361    OSC_MOD                (* Modo de operacion del osciloscopio   *)
23   1:D   361       :INTEGER;
23   1:D   365
23   1:D   365    VENTANA:RECORD
23   1:D   365       PI,                 (* Esquina inf. izq. de la ventana      *)
23   1:D   365       PF                  (* Esquina sup. der. de la ventana      *)
23   1:D   365         :PUNTO;
23   1:D   365       END;
23   1:D   373
```

```
252 23    1:D   373 (*$P*)
253 23    1:D   373
254 23    1:D   373    PUERTO:RECORD
255 23    1:D   373       PI,              (*  Esquina inf. izq. del puerto      *)
256 23    1:D   373       PF,              (*  Esquina sup. der. del puerto      *)
257 23    1:D   373       RL               (*  Relacion puerto/ventana           *)
258 23    1:D   373          :PUNTO;
259 23    1:D   373       END;
260 23    1:D   385
261 23    1:D   385    B_ERR_ES,           (*  Bandera que indica error en ES    *)
262 23    1:D   385    B_RANGO,            (*  Bandera que indica cursor fuera de *)
263 23    1:D   385                                (* rango de la ventana        *)
264 23    1:D   385    B_SALVA             (*  Bandera que indica si se efectua  *)
265 23    1:D   385                              (*  el eco al archivo grafico  *)
266 23    1:D   385       : BOOLEAN;
267 23    1:D   388
268 23    1:D   388    IND_CAR             (*  Indices de acceso a la definicion *)
269 23    1:D   388       : PACKED ARRAY [0..127] OF INTEGER;      (* de caracter *)
270 23    1:D   516
271 23    1:D   516    TAB_CAR             (*  Vectores de definiciones de carac- *)
272 23    1:D   516       : PACKED ARRAY [0..1023] OF BYTE;        (* teres      *)
273 23    1:D  1028
274 23    1:D  1028 (*************************************************************************)
275 23    1:D  1028
276 23   17:D     1 PROCEDURE INICIA_PIA;
277 23   17:D     1
278 23   17:D     1 (*************************************************************************)
279 23   17:D     1 (*                                                                       *)
280 23   17:D     1 (*      Esta rutina inicia la tarjeta de acoplamiento                    *)
281 23   17:D     1 (*      Esta en el archivo CONT.ASM                                      *)
282 23   17:D     1 (*                                                                       *)
283 23   17:D     1 (*************************************************************************)
284 23   17:D     1
285 23   17:D     1          EXTERNAL;
286 23   17:D     1
287 23   17:D     1
288 23   18:D     1 PROCEDURE BORRA_OSC;
289 23   18:D     1
290 23   18:D     1 (*************************************************************************)
291 23   18:D     1 (*                                                                       *)
292 23   18:D     1 (*      Esta rutina borra la pantalla del osciloscopio                   *)
293 23   18:D     1 (*      Esta en el archivo CONT.ASM                                      *)
294 23   18:D     1 (*                                                                       *)
295 23   18:D     1 (*************************************************************************)
296 23   18:D     1
297 23   18:D     1          EXTERNAL;
298 23   18:D     1
299 23   18:D     1
300 23   19:D     1 PROCEDURE INI_PLT;
301 23   19:D     1
302 23   19:D     1 (*************************************************************************)
303 23   19:D     1 (*                                                                       *)
304 23   19:D     1 (*      Esta rutina sube la montura y la baja en el eje Y                *)
305 23   19:D     1 (*      Esta en al archivo CONT.ASM                                      *)
306 23   19:D     1 (*                                                                       *)
307 23   19:D     1 (*************************************************************************)
308 23   19:D     1
309 23   19:D     1          EXTERNAL;
310 23   19:D     1
```

```
23   19:D    1 (*$P*)
23   19:D    1
23   20:D    1 PROCEDURE BDOA(X1,       (* Abscisa inicial en coord. de disp.  *)
23   20:D    1                Y1,       (* Ordenada inicial en coord. de disp. *)
23   20:D    1                X2,       (* Abscisa final en coord. de disp.    *)
23   20:D    1                Y2,       (* Ordenada final en coord. de disp.   *)
23   20:D    1                TIPO      (* Tipo de linea pedida                *)
23   20:D    1                  :INTEGER);
23   20:D    6
23   20:D    6 (*************************************************************)
23   20:D    6 (*                                                           *)
23   20:D    6 (*      Esta rutina genera una linea en el dispositivo DISP, *)
23   20:D    6 (*  con la pluma en pos. PLM, entre [X1,Y1] y [X2,Y2] con    *)
23   20:D    6 (*  tipo de linea TIPO.   Emplea el algoritmo Bresenham.     *)
23   20:D    6 (*      Esta en el archivo CONT.ASM.                         *)
23   20:D    6 (*                                                           *)
23   20:D    6 (*************************************************************)
23   20:D    6
23   20:D    6           EXTERNAL;
23   20:D    6
23   20:D    6
23   21:D    1 PROCEDURE CLIP(VAR PI,          (* Punto inicial en coord. us. *)
23   21:D    1                    PF           (* Punto final en coord. us.   *)
23   21:D    1                      : PUNTO;
23   21:D    3                VAR B_CLIP       (* Indicador de no-trazo       *)
23   21:D    3                      :BOOLEAN);
23   21:D    4
23   21:D    4 (*************************************************************)
23   21:D    4 (*                                                           *)
23   21:D    4 (*      Esta rutina altera PI y PF para que la linea solo tenga *)
23   21:D    4 (*  puntos dentro de la ventana.  B_CLIP indica que no es posi- *)
23   21:D    4 (*  ble trazar una linea.                                    *)
23   21:D    4 (*      Tiene la rutina local CODE.                          *)
23   21:D    4 (*      Usa la var. global VENTANA.                          *)
23   21:D    4 (*                                                           *)
23   21:D    4 (*************************************************************)
23   21:D    4
23   21:D    4 TYPE
23   21:D    4    BORDES=SET OF ORIENTACION;
23   21:D    4
23   21:D    4 VAR
23   21:D    4    C,           (* Codigo del punto de trabajo     *)
23   21:D    4    C1,          (* Codigo del punto PI             *)
23   21:D    4    C2           (* Codigo del punto PF             *)
23   21:D    4      :BORDES;
23   21:D    7    X,           (* Abscisa del punto de trabajo    *)
23   21:D    7    Y            (* Ordenada del punto de trabajo   *)
23   21:D    7      :REAL;
23   21:D   11
```

```
23   21:D    11 (*$P*)
23   22:D     1 PROCEDURE CODE(     P              (*  Punto a codificar              *)
23   22:D     1                         :PUNTO;
23   22:D     2              VAR C                 (*  Codificacion del punto P       *)
23   22:D     2                         :BORDES);
23   22:D     7
23   22:D     7 (*****************************************************************)
23   22:D     7 (*                                                               *)
23   22:D     7 (*       Esta rutina codifica la posicion de un punto con res-   *)
23   22:D     7 (*  pecto a la ventana.   Si esta dentro de la venta, responde   *)
23   22:D     7 (*  con null en el parametro C.                                  *)
23   22:D     7 (*       Es una rutina local a CLIP.                             *)
23   22:D     7 (*       Usa la var. global VENTANA.                             *)
23   22:D     7 (*                                                               *)
23   22:D     7 (*****************************************************************)
23   22:D     7
23   22:0     0 BEGIN   (* Procedure CODE *)
23   22:1     0    C:=[];               (*  Inicia a null la posicion          *)
23   22:1    10    IF P.X<VENTANA.PI.X THEN
23   22:2    24       C:=[IZQ]          (*  Si esta a la izquerda              *)
23   22:1    25    ELSE
23   22:2    32       IF P.X>VENTANA.PF.X THEN
23   22:3    46          C:=[DER];      (*  Si esta a la derecha de la ventana  *)
23   22:1    52    IF P.Y<VENTANA.PI.Y THEN
23   22:2    66       C:=C+[ABJ]        (*  Si ademas esta abajo de la ventana  *)
23   22:1    70    ELSE
23   22:2    78       IF P.Y>VENTANA.PF.Y THEN
23   22:3    92          C:=C+[ARR];  (*  o esta arriba de la ventana          *)
23   22:0   102 END;     (* Procedure CODE *)
23   22:0   114
```

```
390  23   22:0   114 ($$P$)
391  23   21:0     0 $EGIN   (* Procedure  CLIP *)
392  23   21:1     0    CODE(PI,C1);
393  23   21:1     5    CODE(PF,C2);
394  23   21:1    10    B_CLIP:=FALSE;
395  23   21:1    13    WHILE ((C1<>[]) OR (C2<>[])) AND ((C1*C2)=[]) DO
396  23   21:2    35    BEGIN
397  23   21:3    35       C:=C1;
398  23   21:3    41       IF C=[] THEN C:=C2;
399  23   21:3    54       IF IZQ IN C THEN
400  23   21:4    60       BEGIN
401  23   21:5    60          Y:=PI.Y+(PF.Y-PI.Y)*(VENTANA.PI.X-PI.X)/(PF.X-PI.X);
402  23   21:5   100          X:=VENTANA.PI.X;
403  23   21:4   110       END
404  23   21:3   110       ELSE
405  23   21:4   112          IF DER IN C THEN
406  23   21:5   118          BEGIN
407  23   21:6   118             Y:=PI.Y+(PF.Y-PI.Y)*(VENTANA.PF.X-PI.X)/(PF.X-PI.X);
408  23   21:6   158             X:=VENTANA.PF.X;
409  23   21:5   168          END
410  23   21:4   168          ELSE
411  23   21:5   170             IF ABJ IN C THEN
412  23   21:6   176             BEGIN
413  23   21:7   176                X:=PI.X+(PF.X-PI.X)*(VENTANA.PI.Y-PI.Y)/(PF.Y-PI.Y);
414  23   21:7   216                Y:=VENTANA.PI.Y;
415  23   21:6   226             END
416  23   21:5   226             ELSE
417  23   21:6   228                IF ARR IN C THEN
418  23   21:7   234                BEGIN
419  23   21:8   234                   X:=PI.X+(PF.X-PI.X)*(VENTANA.PF.Y-PI.Y)/(PF.Y-PI.Y);
420  23   21:8   274                   Y:=VENTANA.PF.Y;
421  23   21:7   284                END;
422  23   21:3   284       IF C=C1 THEN
423  23   21:4   292       BEGIN
424  23   21:5   292          PI.X:=X;
425  23   21:5   301          PI.Y:=Y;
426  23   21:5   308          CODE(PI,C1);
427  23   21:4   313       END
428  23   21:3   313       ELSE
429  23   21:4   315       BEGIN
430  23   21:5   315          PF.X:=X;
431  23   21:5   324          PF.Y:=Y;
432  23   21:5   331          CODE(PF,C2);
433  23   21:4   336       END;($ENDIF$)
434  23   21:2   336    END;($ENDWHILE$)
435  23   21:1   338    IF (C1*C2)<>[] THEN
436  23   21:2   348       B_CLIP:=TRUE;
437  23   21:0   351 END;    (* Procedure CLIP *)
438  23   21:0   370
439  23   21:0   370 ($$I GRLIB01 $)
439  23   21:0   370 ($$I GRLIB02 $)
```

```
23   21:0   370 (*$P*)
23   21:0   370 (*************************************************************************)
23   21:0   370 (*                                                                       *)
23   21:0   370 (*      Archivo GRLIB02  se incluye en GRLIB00                           *)
23   21:0   370 (*                                                                       *)
23   21:0   370 (*                                    Realizo:                           *)
23   21:0   370 (*                                              Miguel Chin Auyon        *)
23   21:0   370 (*                                                                       *)
23   21:0   370 (*                         Fecha de creacion      :  5-mar-85            *)
23   21:0   370 (*                                                                       *)
23   21:0   370 (*                         Fecha de actualizacion:  18-mar-85            *)
23   21:0   370 (*                                                                       *)
23   21:0   370 (*************************************************************************)
23   21:0   370
23   21:0   370
23   23:D     1 PROCEDURE T_VEN_PR(    P        (*  Punto en coord. usuario      *)
23   23:D     1                              :PUNTO;
23   23:D     2                        VAR X,       (*  Abscisa en coord. de disp.  *)
23   23:D     2                            Y        (*  Ordenada en coord. de disp. *)
23   23:D     2                              :INTEGER);
23   23:D     8
23   23:D     8 (*************************************************************************)
23   23:D     8 (*                                                                       *)
23   23:D     8 (*      Transforma coordenadas del mundo usuario a las coord.           *)
23   23:D     8 (*  de dispositivo de acuerdo a ventana y puerto definidas.             *)
23   23:D     8 (*      Usa las var. globales VENTANA y PUERTO.                          *)
23   23:D     8 (*                                                                       *)
23   23:D     8 (*************************************************************************)
23   23:D     8
23   23:0     0 BEGIN   (* Procedure T_VEN_PR *)
23   23:1     0     X:=TRUNC((P.X-VENTANA.PI.X)*PUERTO.RL.X+PUERTO.PI.X);
23   23:1    34     Y:=TRUNC((P.Y-VENTANA.PI.Y)*PUERTO.RL.Y+PUERTO.PI.Y);
23   23:0    63 END;     (* Procedure T_VEN_PR *)
23   23:0    76
23   23:0    76
23   24:D     1 PROCEDURE T_PR_VEN(VAR P        (*  Punto en coord. usuario      *)
23   24:D     1                              :PUNTO;
23   24:D     2                            X,       (*  Abscisa en coord. de disp.  *)
23   24:D     2                            Y        (*  Ordenada en coord. de disp. *)
23   24:D     2                              :INTEGER);
23   24:D     4
23   24:D     4 (*************************************************************************)
23   24:D     4 (*                                                                       *)
23   24:D     4 (*      Transforma de coordenadas de dispositivo a coordenadas          *)
23   24:D     4 (*  del mundo del usuario.                                               *)
23   24:D     4 (*      Usa las var. globales VENTANA y PUERTO.                          *)
23   24:D     4 (*                                                                       *)
23   24:D     4 (*************************************************************************)
23   24:D     4
23   24:0     0 BEGIN   (* Procedure T_PR_VEN *)
23   24:1     0     P.X:=(X-PUERTO.PI.X)/PUERTO.RL.X+VENTANA.PI.X;
23   24:1    28     P.Y:=(Y-PUERTO.PI.Y)/PUERTO.RL.Y+VENTANA.PI.Y;
23   24:0    54 END;     (* Procedure T_P_RVEN *)
```

```
23   24:0   66 (*$P*)
23   25:D    1 PROCEDURE SALVA(COM                (*  Comando a salvar             *)
23   25:D    1                    :BYTE;
23   25:D    2                PAR1,              (*  Primer parametro             *)
23   25:D    2                PAR2               (*  Segundo parametro            *)
23   25:D    2                   :INTEGER);
23   25:D    4
23   25:D    4 (***************************************************************)
23   25:D    4 (*                                                              *)
23   25:D    4 (*       Esta rutina salva la instruccion grafica con el comando *)
23   25:D    4 (*  COM y los parametros PAR1 y PAR2, si la bandera B_SALVA      *)
23   25:D    4 (*  esta prendida.                                               *)
23   25:D    4 (*       Usa las var. globales B_SALVA y ECO.                    *)
23   25:D    4 (*                                                              *)
23   25:D    4 (***************************************************************)
23   25:D    4
23   25:0    0 BEGIN   (* Procedure SALVA *)
23   25:1    0    IF B_SALVA THEN
23   25:2    6    BEGIN
23   25:3    6       WITH ECO^ DO
23   25:4   11       BEGIN
23   25:5   11          COMANDO:=COM;
23   25:5   19          PARAM1:=PAR1;
23   25:5   24          PARAM2:=PAR2;
23   25:4   29       END;
23   25:3   29       PUT(ECO);
23   25:2   37    END;
23   25:0   37 END;    (* Procedure SALVA *)
23   25:0   50
23   25:0   50
23   26:D    1 PROCEDURE MUEVE_ABS( X,         (*  Abscisa en coord. de disp.  *)
23   26:D    1                      Y          (*  Ordenada en coord. de disp. *)
23   26:D    1                          :INTEGER);
23   26:D    3
23   26:D    3 (***************************************************************)
23   26:D    3 (*                                                              *)
23   26:D    3 (*       Esta rutina privada, mueve en forma absoluta, la pluma  *)
23   26:D    3 (*  a las coordenadas (X,Y) del dispositivo seleccionado.        *)
23   26:D    3 (*       Su comando grafico es el 9.                             *)
23   26:D    3 (*       Usa la rutina BDDA y SALVA.                             *)
23   26:D    3 (*       Recibe los parametros X,Y                               *)
23   26:D    3 (*                                                              *)
23   26:D    3 (***************************************************************)
23   26:D    3
23   26:D    3 VAR
23   26:D    3    PLM_TEMP             (*  Guarda la pos. de la pluma          *)
23   26:D    3       :INTEGER;
23   26:D    4
23   26:0    0 BEGIN   (* Procedure MUEVE_ABS *)
23   26:1    0    PLM_TEMP:=PLM;        (*  Salva la pos. de la pluma          *)
23   26:1    6    PLM:=2;               (*  Marca la pluma levantada           *)
23   26:1   11    BDDA(CD.X,CD.Y,X,Y,0);     (*  Genera el trazo               *)
23   26:1   24    SALVA(9,X,Y);         (*  Salva la instruccion grafica       *)
23   26:1   34    PLM:=PLM_TEMP;        (*  Regresa la pos. de la pluma        *)
23   26:1   39    CD.X:=X;              (*  Actualiza el valor de CD           *)
23   26:1   44    CD.Y:=Y;
23   26:0   49 END;    (* Procedure MUEVE_ABS *)
```

```
 23   26:0   62 (**P*)
 23   26:0   62
 23    2:D    1 PROCEDURE PLM_ARR;
 23    2:D    1
 23    2:D    1 (******************************************************************)
 23    2:D    1 (*                                                                *)
 23    2:D    1 (*      Esta rutina publica, provoca que la pluma del disp.       *)
 23    2:D    1 (*  suba.  El comando grafico es 1.                               *)
 23    2:D    1 (*      Llama a la rutina SALVA.                                  *)
 23    2:D    1 (*      Usa la var. global PLM.                                   *)
 23    2:D    1 (*                                                                *)
 23    2:D    1 (******************************************************************)
 23    2:D    1
 23    2:0    0 BEGIN   (* Procedure PLM_ARR *)
 23    2:1    0    IF PLM<>2 THEN
 23    2:2    8    BEGIN
 23    2:3    8       PLM:=2;
 23    2:3   13       SALVA(1,0,0);
 23    2:2   23    END;(*ENDIF*)
 23    2:0   23 END;     (* Procedure PLM_ARR *)
 23    2:0   36
 23    2:0   36
 23    3:D    1 PROCEDURE PLM_ABJ;
 23    3:D    1
 23    3:D    1 (******************************************************************)
 23    3:D    1 (*                                                                *)
 23    3:D    1 (*      Esta rutina publica, provoca que la pluma del disp.       *)
 23    3:D    1 (*  baje.  El comando grafico es 2.                               *)
 23    3:D    1 (*      Llama a la rutina SALVA.                                  *)
 23    3:D    1 (*      Usa la var. global PLM.                                   *)
 23    3:D    1 (*                                                                *)
 23    3:D    1 (******************************************************************)
 23    3:D    1
 23    3:0    0 BEGIN   (* Procedure PLM_ABJ *)
 23    3:1    0    IF PLM<>1 THEN
 23    3:2    8    BEGIN
 23    3:3    8       PLM:=1;
 23    3:3   13       SALVA(2,0,0);
 23    3:2   23    END;(*ENDIF*)
 23    3:0   23 END;     (* Procedure PLM_ABJ *)
 23    3:0   36
```

```
23   3:0    36 (**P*)
23   3:0    36
23   4:D     1 PROCEDURE LINEA_HACIA;
23   4:D     5
23   4:D     5 (*****************************************************************)
23   4:D     5 (*                                                               *)
23   4:D     5 (*        Esta rutina publica, traza una linea desde la posicion  *)
23   4:D     5 (*  actual del cursor del usuario CM, hasta el punto [X,Y]       *)
23   4:D     5 (*  destino, efectuando la correcion por la rutina CLIP.         *)
23   4:D     5 (*        Su comando grafico es el 3.                             *)
23   4:D     5 (*        Llama a las rutinas BDDA, SALVA, CLIP y T_VEN_PR.       *)
23   4:D     5 (*        Usa las var. globales CD, CM y B_RANGO.                 *)
23   4:D     5 (*        Recibe los parametros X y Y.                            *)
23   4:D     5 (*                                                               *)
23   4:D     5 (*****************************************************************)
23   4:D     5
23   4:D     5 VAR
23   4:D     5    PTEMP1,              (* Almacena punto inicial              *)
23   4:D     5    PTEMP2               (* Almacena punto final                *)
23   4:D     5       :PUNTO;
23   4:D    13    PLM_TEMP,            (* Almacena la pos. de la pluma        *)
23   4:D    13    XD,                  (* Abscisa en coor. de dispositivo     *)
23   4:D    13    YD                   (* Ordenada en coord. de dispositivo   *)
23   4:D    13       :INTEGER;
23   4:D    16    B_CLIP               (* Deteccion de no-trazo               *)
23   4:D    16       :BOOLEAN;
23   4:D    17
23   4:0     0 BEGIN (* Procedure LINEA_HACIA *)
23   4:1     0    PTEMP1:=CM;          (* Copia el punto inicial CM           *)
23   4:1     7    PTEMP2.X:=X;         (* Copia el punto final [X,Y]          *)
23   4:1    15    PTEMP2.Y:=Y;
23   4:1    23    CLIP(PTEMP1,PTEMP2,B_CLIP);  (*  Haz el recorte en las copias*)
23   4:1    31    IF NOT(B_CLIP) THEN  (* Si hay trazo, entonces             *)
23   4:2    35    BEGIN
23   4:3    35       IF PTEMP1<>CM THEN        (*  Si entra en la ventana     *)
23   4:4    45       BEGIN
23   4:5    45          B_RANGO:=FALSE;        (*  Ya no esta fuera de rango  *)
23   4:5    50          T_VEN_PR(PTEMP1,XD,YD);(*  Haz la transf. de PTEMP1   *)
23   4:5    58          MUEVE_ABS(XD,YD);      (*  Mueve en absoluto a PTEMP1 *)
23   4:4    62       END;
23   4:3    62       T_VEN_PR(PTEMP2,XD,YD);   (*  Haz la transf. de PTEMP2   *)
23   4:3    70       BDDA(CD.X,CD.Y,XD,YD,TIPO);      (*  Traza linea         *)
23   4:3    86       SALVA(3,XD,YD);           (*  Salva la inst. grafica     *)
23   4:3    96       CD.X:=XD;                 (*  Actualiza CD               *)
23   4:3   101       CD.Y:=YD;
23   4:3   106       IF (PTEMP2.X<>X) OR (PTEMP2.Y<>Y) THEN    (* Si sale     *)
23   4:4   129       BEGIN
23   4:5   129          B_RANGO:=TRUE;         (*  Esta fuera del rango       *)
23   4:5   134          MUEVE_ABS(XD,YD);      (*  Sube la pluma sin perder su *)
23   4:5   138                                        (*  estado actual       *)
23   4:4   138       END;(*ENDIF*)
23   4:2   138    END;(*ENDIF*)
23   4:1   138    CM.X:=X;                     (*  Actualiza CM               *)
23   4:1   147    CM.Y:=Y;
23   4:0   156 END;    (* Procedure LINEA_HACIA *)
```

```
23    4:0  168 (*$P*)
23    4:0  168
23    5:0    1 PROCEDURE LINEA;
23    5:0    9
23    5:0    9 (********************************************************************)
23    5:0    9 (*                                                                  *)
23    5:0    9 (*       Traza una linea entre los puntos [X1,Y1] y [X2,Y2]        *)
23    5:0    9 (*  con el TIPO de linea existente y la posicion de pluma          *)
23    5:0    9 (*  existente.                                                      *)
23    5:0    9 (*       Llama a las rutinas MUEVE_HACIA y LINEA_HACIA.             *)
23    5:0    9 (*       Recibe los parametros X1,Y1,X2,Y2                          *)
23    5:0    9 (*                                                                  *)
23    5:0    9 (********************************************************************)
23    5:0    9
23    5:0    0 BEGIN   (* Procedure LINEA *)
23    5:1    0    IF (CM.X<>X1) OR (CM.Y<>Y1) THEN
23    5:2   25       MUEVE_HACIA(X1,Y1);
23    5:2   35    (*endif*)
23    5:1   35    LINEA_HACIA(X2,Y2);
23    5:0   45 END;     (* Procedure LINEA *)
23    5:0   58
23    5:0   58 (*$I GRLIB02 *)
23    5:0   58 (*$I GRLIB03 *)
```

```
668  23    5:0   58 (*$P*)
669  23    5:0   58 (*************************************************************************)
670  23    5:0   58 (*                                                                       *)
671  23    5:0   58 (*      Archivo GRLIB03 con la tercera parte de las                      *)
672  23    5:0   58 (*      declaraciones privadas.                                          *)
673  23    5:0   58 (*                                                                       *)
674  23    5:0   58 (*                                  Realizo:                             *)
675  23    5:0   58 (*                                          Miguel Chin Auyon            *)
676  23    5:0   58 (*                                                                       *)
677  23    5:0   58 (*                        Fecha de creacion     :  5-mar-85             *)
678  23    5:0   58 (*                        Fecha de actualizacion:  13-mar-85            *)
679  23    5:0   58 (*                                                                       *)
680  23    5:0   58 (*************************************************************************)
681  23    5:0   58
682  23    5:0   58
683  23    6:D    1 PROCEDURE MUEVE_HACIA;
684  23    6:D    5
685  23    6:D    5 (*************************************************************************)
686  23    6:D    5 (*                                                                       *)
687  23    6:D    5 (*      Mueve la pluma a la posicion [X,Y] asegurando que la             *)
688  23    6:D    5 (*  pluma este levantada y regresa la pluma a su estado                  *)
689  23    6:D    5 (*  original.  No genera comando grafico propio.                         *)
690  23    6:D    5 (*      Llama a PLM_ABJ, PLM_ARR y LINEA_HACIA.                          *)
691  23    6:D    5 (*      Recibe los parametros X,Y.                                       *)
692  23    6:D    5 (*                                                                       *)
693  23    6:D    5 (*************************************************************************)
694  23    6:D    5
695  23    6:D    5 VAR
696  23    6:D    5   PLM_TEMP              (*  Alamacena la posicion de la pluma        *)
697  23    6:D    5       :INTEGER;
698  23    6:D    6
699  23    6:0    0 BEGIN   (* Procedure MUEVE_HACIA *)
700  23    6:1    0   PLM_TEMP:=PLM;       (*  Salva el estado de la pluma              *)
701  23    6:1    6   IF PLM=1 THEN        (*  Si esta abajo, levantala                 *)
702  23    6:2   14     PLM_ARR;
703  23    6:1   16   LINEA_HACIA(X,Y);    (*  Genera el movimiento                     *)
704  23    6:1   26   IF PLM_TEMP<>PLM THEN(*  Si cambiaste el estado de la pluma,      *)
705  23    6:2   34     PLM_ABJ;           (*  bajala de nuevo                          *)
706  23    6:0   36 END;    (* Procedure MUEVE_HACIA *)
707  23    6:0   48
708  23    6:0   48
709  23    7:D    1 PROCEDURE ESC_STR;
710  23    7:D   45
711  23    7:D   45 (*************************************************************************)
712  23    7:D   45 (*                                                                       *)
713  23    7:D   45 (*      Escribe una cadena de caracteres con escala ESC y                *)
714  23    7:D   45 (*  orientacion ORIENTA.                                                 *)
715  23    7:D   45 (*      Llama a la rutina ESC_CAR.                                       *)
716  23    7:D   45 (*      Recibe los parametros STR, ESC y ORIENTA                         *)
717  23    7:D   45 (*                                                                       *)
718  23    7:D   45 (*************************************************************************)
719  23    7:D   45
720  23    7:D   45 VAR
721  23    7:D   45   I           (*  Contador de caracteres de la cadena              *)
722  23    7:D   45       :INTEGER;
723  23    7:D   46
724  23    7:0    0 BEGIN   (* Procedure ESC_STR *)
725  23    7:1    0   FOR I:=1 TO LENGTH(STR) DO          (*  Manda cada caracter *)
726  23    7:2   21     ESC_CAR(STR[I],ESC,ORIENTA);      (*  a ESC_CAR           *)
727  23    7:0   39 END;    (* Procedure ESC_STR *)
```

```
23   7:0   54
23   8:D    1 PROCEDURE ESC_CAR;
23   8:D    4
23   8:D    4 (*****************************************************************)
23   8:D    4 (*                                                               *)
23   8:D    4 (*      Escribe un solo caracter con escala ESC y orientacion    *)
23   8:D    4 (*  ORIENTA, verificando los margenes del puerto.                *)
23   8:D    4 (*      Su comando grafico es de 5 a 8.                          *)
23   8:D    4 (*      Llama a las rutinas BDDA, T_PR_VEN y SALVA               *)
23   8:D    4 (*      Usa las globales CD, CM, PLM, B_ERR_ES, IND_CAR          *)
23   8:D    4 (*  y TAB_CAR.                                                   *)
23   8:D    4 (*      Tiene la funcion local B_CLIP y el procedimiento local   *)
23   8:D    4 (*  DESEMPACA.                                                   *)
23   8:D    4 (*      Recibe los parametros CAR, ESC y ORIENTA.                *)
23   8:D    4 (*                                                               *)
23   8:D    4 (*****************************************************************)
23   8:D    4
23   8:D    4 VAR
23   8:D    4    ASCII,      (* Valor ASCII en 7 bits correspondiente a CAR *)
23   8:D    4    DX,         (* Incremento en X en coord. de disp.          *)
23   8:D    4    DY,         (* Incremento en Y en coord. de disp.          *)
23   8:D    4    IND,        (* Indice a la tabla de caracteres             *)
23   8:D    4    OX,         (* Abscisa del origen relativo del caracter    *)
23   8:D    4    OY,         (* Ordenada del origen relativo del caracter   *)
23   8:D    4    PLM_TEMP    (* Almacena el valor de la pluma               *)
23   8:D    4      :INTEGER;
23   8:D   11    EOC         (* Determina el fin de la def. del caracter    *)
23   8:D   11      :BOOLEAN;
23   8:D   12
23  27:D    1 PROCEDURE DESEMPACA(VAR IND     (* Apuntador a TAB_CAR        *)
23  27:D    1                              :INTEGER);
23  27:D    2
23  27:D    2 (*****************************************************************)
23  27:D    2 (*                                                               *)
23  27:D    2 (*      Este procedimiento es local a ESC_CAR.                   *)
23  27:D    2 (*      Desglosa cada trazo del caracter apuntado por IND y      *)
23  27:D    2 (*  genera los incrementos en cada eje, considerando la escala   *)
23  27:D    2 (*  y orientacion pedida.  Detecta fin de caracter EOC y valor   *)
23  27:D    2 (*  de pluma del trazo.  Incrementa el valor de IND.             *)
23  27:D    2 (*                                                               *)
23  27:D    2 (*****************************************************************)
```

```
769 23  27:0    2 (*$P*)
770 23  27:0    2
771 23  27:0    2 VAR
772 23  27:0    2    MV          (* Vector comprimido del trazo              *)
773 23  27:0    2       :BYTE;
774 23  27:0    3    TEMP        (* Variable temporal para la orientacion     *)
775 23  27:0    3       :INTEGER;
776 23  27:0    4
777 23  27:0    0 BEGIN   (* Procedure DESEMPACA *)
778 23  27:1    0    MV:=TAB_CAR[IND];    (* Obten el vector de la tabla       *)
779 23  27:1   19    IND:=IND+1;          (* Incrementa el indice              *)
780 23  27:1   25    DX:=ESC*(MV MOD 8);  (* Captura el inc. en X escalado     *)
781 23  27:1   35    MV:=MV DIV 8;
782 23  27:1   45    DY:=ESC*(MV MOD 8);  (* Captura el inc. en Y escalado     *)
783 23  27:1   55    MV:=MV DIV 8;
784 23  27:1   65    PLM:=(MV  MOD 2)+1;  (* Obten la posicion de la pluma     *)
785 23  27:1   74    EOC:=MV>1;           (* Detecta fin de caracter           *)
786 23  27:1   80    CASE ORIENTA OF
787 23  27:1   85    DER:
788 23  27:2   85       BEGIN             (* Si es hacia la der.  rota -90     *)
789 23  27:3   85          TEMP:=DX;
790 23  27:3   90          DX:=DY;
791 23  27:3   96          DY:=-TEMP
792 23  27:2   96       END;
793 23  27:1  103    ABJ:
794 23  27:2  103       BEGIN             (* Si es hacia abajo, rota 180       *)
795 23  27:3  103          DX:=-DX;
796 23  27:3  110          DY:=-DY
797 23  27:2  110       END;
798 23  27:1  119    IZQ:
799 23  27:2  119       BEGIN             (* Si es hacia la izq., rota 90      *)
800 23  27:3  119          TEMP:=DY;
801 23  27:3  124          DY:=DX;
802 23  27:3  130          DX:=-TEMP
803 23  27:2  130       END
804 23  27:1  135    END;(*ENDCASE*)
805 23  27:1  150    DX:=OX+DX;
806 23  27:1  160    DY:=OY+DY
807 23  27:0  163 END;    (* Procedure DESEMPACA *)
```

```
23   27:0   182 (*$P*)
23   28:0     3 FUNCTION B_CLIP:BOOLEAN;
23   28:0     3
23   28:0     3 (*****************************************************************)
23   28:0     3 (*                                                               *)
23   28:0     3 (*      Esta funcion es local para ESC_CAR.                      *)
23   28:0     3 (*      Detecta si es posible que al trazar el caracter, se      *)
23   28:0     3 (*  sobrepasen los margenes del puerto definido.  Considera     *)
23   28:0     3 (*  que el caracter esta definido en una matriz de 8*8  y       *)
23   28:0     3 (*  toma en cuenta su escala y orientacion.                      *)
23   28:0     3 (*      Usa las var. globales CD, B_RANGO y PUERTO               *)
23   28:0     3 (*                                                               *)
23   28:0     3 (*****************************************************************)
23   28:0     3
23   28:0     0 BEGIN   (* Function B_CLIP *)
23   28:1     0    DX:=7;               (*  Maximo inc. en X sin escala          *)
23   28:1     4    DY:=7;               (*  Maximo inc. en Y sin escala          *)
23   28:1     8    CASE ORIENTA OF
23   28:1    13    DER:                 (*  Rotacion de -90 grados               *)
23   28:2    13       DY:=-7;
23   28:1    20    ABJ:                 (*  Rotacion de 180 grados               *)
23   28:2    20       BEGIN
23   28:3    20          DX:=-7;
23   28:3    25          DY:=-7;
23   28:2    30       END;
23   28:1    32    IZQ:                 (* Rotacion de 90 grados                 *)
23   28:2    32       DX:=-7;
23   28:1    39    END;(*ENDCASE*)
23   28:1    52    DX:=DX*ESC+CD.X;     (*  Maximo punto que alcanzaria en X     *)
23   28:1    67    DY:=DY*ESC+CD.Y;     (*  Maximo punto que alcanzaria en Y     *)
23   28:1    82    B_CLIP:= (DX<PUERTO.PI.X) OR (* Si rebasa el margen izq.      *)
23   28:1    94             (DX>PUERTO.PF.X) OR (* Si rebasa el margen der.      *)
23   28:1   107             (DY<PUERTO.PI.Y) OR (* Si rebasa el margen inf.      *)
23   28:1   120             (DY>PUERTO.PF.Y) OR (* Si rebasa el margen sup.      *)
23   28:1   133             B_RANGO;            (* Si esta fuera de rango        *)
23   28:0   140 END;    (* Function B_CLIP *)
23   28:0   152
23    8:0     0 BEGIN   (* Procedure ESC_CAR *)
23    8:1     0    B_ERR_ES:=TRUE;
23    8:1     5    IF NOT(B_CLIP) THEN  (*  Si es posible trazar el caracter,    *)
23    8:2    12    BEGIN
23    8:3    12       ASCII:=ORD(CAR) MOD 128;   (*  Obten el ASCII del car.       *)
23    8:3    19       PLM_TEMP:=PLM;             (*  Guarda la pos. de la pluma    *)
23    8:3    25       OX:=CD.X;                  (*  Almacena la absc. de CD       *)
23    8:3    31       OY:=CD.Y;                  (*  Almacena la ord. de CD        *)
23    8:3    37       IND:=IND_CAR(ASCII);       (*  Obten el ap. a TAB_CAR        *)
23    8:3    50       REPEAT
23    8:4    50          DESEMPACA(IND);         (*  Desempaca cada trazo          *)
23    8:4    54          (*  Haz el trazo hacia DX, DY escalados con linea cont. *)
23    8:4    54          BDDA(CD.X,CD.Y,DX,DY,0);
23    8:4    67          CD.X:=DX;     CD.Y:=DY;        (* Actualiza CD           *)
23    8:3    77       UNTIL EOC;            (*  Hasta detectar fin de caracter    *)
23    8:3    80       PLM:=PLM_TEMP;     (*  Regresa la pos. de la pluma          *)
23    8:3    85       T_PR_VEN(CM,CD.X,CD.Y);    (*  Mapea CD a CM                 *)
23    8:3    98       SALVA(5+ORD(ORIENTA),ASCII,ESC);   (*  Salva la inst. graf *)
23    8:3   110       B_ERR_ES:=FALSE
23    8:2   110    END(*endif*)
23    8:0   115 END;    (* Procedure ESC_CAR *)
```

```
866 23   8:0  130 (*$P8)
867 23   8:0  130
868 23   8:0  130 (*$I GRLIB03 *)
869 23   8:0  130 (*$I GRLIB04 *)
870 23   8:0  130 (*************************************************************************)
871 23   8:0  130 (*                                                                       *)
872 23   8:0  130 (*      Archivo GRLIB04 con la cuarta parte de las                       *)
873 23   8:0  130 (*      declaraciones privadas.                                          *)
874 23   8:0  130 (*                                                                       *)
875 23   8:0  130 (*                                  Realizo:                             *)
876 23   8:0  130 (*                                          Miguel Chin Auyon            *)
877 23   8:0  130 (*                                                                       *)
878 23   8:0  130 (*                        Fecha de creacion     :  5-mar-85             *)
879 23   8:0  130 (*                                                                       *)
880 23   8:0  130 (*                        Fecha de actualizacion:  18-mar-85            *)
881 23   8:0  130 (*                                                                       *)
882 23   8:0  130 (*************************************************************************)
883 23   8:0  130
884 23   9:0    1 PROCEDURE SEL_VENTANA;
885 23   9:0    9
886 23   9:0    9 (*************************************************************************)
887 23   9:0    9 (*                                                                       *)
888 23   9:0    9 (*      Esta rutina publica sirve para dar entrada a nuevos              *)
889 23   9:0    9 (*  parametros de la ventana.                                            *)
890 23   9:0    9 (*      El punto [X1,Y1] es la esquina inferior izquierda y              *)
891 23   9:0    9 (*  el punto [X2,Y2] es la esquina superior derecha.                     *)
892 23   9:0    9 (*      Revisa si estos son validos, actualiza la relacion con           *)
893 23   9:0    9 (*  el puerto, y genera el nuevo CM.                                     *)
894 23   9:0    9 (*      Llama a T_PR_VEN.                                                *)
895 23   9:0    9 (*      Recibe X1,Y1,X2,Y2                                               *)
896 23   9:0    9 (*      Usa VENTANA, PUERTO, B_ERR_ES, CD y CM.                          *)
897 23   9:0    9 (*                                                                       *)
898 23   9:0    9 (*************************************************************************)
899 23   9:0    9
900 23   9:0    0 BEGIN   (* Procedure SEL_VENTANA *)
901 23   9:1    0   IF (X1)=X2) OR (Y1)=Y2) THEN (*  Si la vent. es invalida,          *)
902 23   9:2   23     B_ERR_ES:=TRUE
903 23   9:1   23   ELSE
904 23   9:2   30   BEGIN
905 23   9:3   30     B_ERR_ES:=FALSE;
906 23   9:3   35     VENTANA.PI.X:=X1;          (*  Almacena las esquinas en la *)
907 23   9:3   45     VENTANA.PI.Y:=Y1;          (*  var. VENTANA                *)
908 23   9:3   55     VENTANA.PF.X:=X2;
909 23   9:3   65     VENTANA.PF.Y:=Y2;
910 23   9:3   75     (*  Recalcula las relaciones puerto/ventana                      *)
911 23   9:3   75     PUERTO.RL.X:=(PUERTO.PI.X-PUERTO.PF.X)/(X1-X2);
912 23   9:3  104     PUERTO.RL.Y:=(PUERTO.PI.Y-PUERTO.PF.Y)/(Y1-Y2);
913 23   9:3  133     T_PR_VEN(CM,CD.X,CD.Y);   (*  Recalcula CM de CD           *)
914 23   9:2  146   END;(*ENDIF*)
915 23   9:0  146 END;    (* Procedure SEL_VENTANA *)
```

```
23   9:0  158 (*$P*)
23   9:0  158
23  10:D    1 PROCEDURE SEL_PUERTO;
23  10:D    9
23  10:D    9 (**********************************************************************)
23  10:D    9 (*                                                                    *)
23  10:D    9 (*      Esta rutina publica admite los nuevos parametros para         *)
23  10:D    9 (*  el puerto de vision.                                              *)
23  10:D    9 (*      La esquina inferior izquierda es [X1,Y1] y la esquina         *)
23  10:D    9 (*  superior derecha esta dada por [X2,Y2].                           *)
23  10:D    9 (*      Actualiza el valor de PUERTO, CD y CM y recalcula la          *)
23  10:D    9 (*  relacion con la ventana.  Identifica parametros invalidos.        *)
23  10:D    9 (*      Llama a la rutina MUEVE_ABS.                                  *)
23  10:D    9 (*      Usa las variables globales PUERTO, VENTANA, B_ERR_ES,         *)
23  10:D    9 (*  B_RANGO, CM y CD.                                                 *)
23  10:D    9 (*      Recibe los parametros X1,Y1,X2,Y2                             *)
23  10:D    9 (*                                                                    *)
23  10:D    9 (**********************************************************************)
23  10:D    9
23  10:0    0 BEGIN   (* Procedure SEL_PUERTO *)
23  10:1    0    IF (X1)=X2) OR (Y1)=Y2) OR (X1<0) OR (Y1<0) OR
23  10:1   39       (X2>PLTXMAX) OR (Y2>PLTYMAX) OR
23  10:1   61       ((DISP=0) AND ((X2>OSCXMAX) OR (Y2>OSCYMAX)))
23  10:1   89                            THEN (*  Si el puerto es invalido,    *)
23  10:2   92       B_ERR_ES:=TRUE
23  10:1   92    ELSE
23  10:2   99    BEGIN
23  10:3   99       B_RANGO:=FALSE;
23  10:3  104       B_ERR_ES:=FALSE;
23  10:3  109       PUERTO.PI.X:=TRUNC(X1);    (*  Almacena los valores de      *)
23  10:3  122       PUERTO.PI.Y:=TRUNC(Y1);    (*  los margenes                 *)
23  10:3  135       PUERTO.PF.X:=TRUNC(X2);
23  10:3  148       PUERTO.PF.Y:=TRUNC(Y2);
23  10:3  161       (*  Recalcula la relacion puerto/ventana                      *)
23  10:3  161       PUERTO.RL.X:=(PUERTO.PI.X-PUERTO.PF.X)/
23  10:3  178                      (VENTANA.PI.X-VENTANA.PF.X);
23  10:3  194       PUERTO.RL.Y:=(PUERTO.PI.Y-PUERTO.PF.Y)/
23  10:3  211                      (VENTANA.PI.Y-VENTANA.PF.Y);
23  10:3  227       CM.X:=0;             (* Inicia CM a la esquina inf. izq.     *)
23  10:3  234       CM.Y:=0;
23  10:3  241       (*  Mueve fisicamente la pluma a la esquina inf. izq.         *)
23  10:3  241       MUEVE_ABS(TRUNC(PUERTO.PI.X),TRUNC(PUERTO.PI.Y));
23  10:2  259    END;(*ENDIF*)
23  10:0  259 END;    (* Procedure SEL_PUERTO *)
```

```
960  23   10:0   274 (*$P0)
961  23   11:0     1 PROCEDURE SEL_TIPO;
962  23   11:0     2
963  23   11:0     2 (*************************************************************************)
964  23   11:0     2 (*                                                                       *)
965  23   11:0     2 (*      Esta rutina publica altera el tipo de linea con el que           *)
966  23   11:0     2 (*  se esta trabajando.                                                  *)
967  23   11:0     2 (*      Este es un valor entre 0 y 127, con 0 = linea continua           *)
968  23   11:0     2 (*  y los valores subsecuentes son lineas discontinuas.                  *)
969  23   11:0     2 (*      Su comando grafico es el 4.                                      *)
970  23   11:0     2 (*      Llama a la rutina SALVA y recibe el parametro T                  *)
971  23   11:0     2 (*                                                                       *)
972  23   11:0     2 (*************************************************************************)
973  23   11:0     2
974  23   11:0     0 BEGIN   (* Procedure SEL_TIPO *)
975  23   11:1     0    IF TIPO<>(T MOD 128) THEN
976  23   11:2    12    BEGIN
977  23   11:3    12       TIPO:=T MOD 128;  (*  Alaacena el tipo de linea 0-127      *)
978  23   11:3    21       SALVA(4,TIPO,0);  (*  Salva la instruccion grafica         *)
979  23   11:2    34    END;(*ENDIF*)
980  23   11:0    34 END;    (* Procedure SEL_TIPO *)
981  23   11:0    46
982  23   11:0    46
983  23   12:0     1 PROCEDURE INI_GRAF;
984  23   12:0     3
985  23   12:0     3 (*************************************************************************)
986  23   12:0     3 (*                                                                       *)
987  23   12:0     3 (*      Esta rutina publica inicia el sistema grafico.                   *)
988  23   12:0     3 (*      Selecciona el dispositivo de salida, entre OSC y PLT             *)
989  23   12:0     3 (*  y si se realiza el eco en un archivo grafico.                        *)
990  23   12:0     3 (*      Llama a la rutina SEL_VENTANA.                                   *)
991  23   12:0     3 (*      Tiene el procedimiento local BORRA.                              *)
992  23   12:0     3 (*      Recibe de parametros DISPO y B_ECO.                              *)
993  23   12:0     3 (*      Usa las var.globales PUERTO, B_ERR_ES, B_RANGO, DISP,            *)
994  23   12:0     3 (*  y B_SALVA.                                                           *)
995  23   12:0     3 (*                                                                       *)
996  23   12:0     3 (*************************************************************************)
997  23   12:0     3
998  23   29:0     1 PROCEDURE BORRA;
999  23   29:0     1
1000 23   29:0     1 (*************************************************************************)
1001 23   29:0     1 (*                                                                       *)
1002 23   29:0     1 (*      Este procedimiento local a INI_GRAF, borra el disposi-           *)
1003 23   29:0     1 (*  tivo grafico seleccionado.                                           *)
1004 23   29:0     1 (*      Llama a BORRA_OSC o a INI_PLT                                    *)
1005 23   29:0     1 (*      Usa las var. globales DISP, PLM, CD y CM.                        *)
1006 23   29:0     1 (*                                                                       *)
1007 23   29:0     1 (*************************************************************************)
1008 23   29:0     1
1009 23   29:0     0 BEGIN   (* Procedure BORRA *)
1010 23   29:1     0    IF DISP=0 THEN       (*  Llama a la rutina de borrado        *)
1011 23   29:2     8       BORRA_OSC         (*  adecuada al dispositivo seleccionado *)
1012 23   29:1     8    ELSE
1013 23   29:2    12       INI_PLT;
1014 23   29:1    14    CD.X:=0;     CD.Y:=0;       (*  Inicia CD y CM al origen    *)
1015 23   29:1    24    CM.X:=0;     CM.Y:=0;
1016 23   29:1    38    PLM:=2;                     (*  Marca la pluma como levantada *)
1017 23   29:0    43 END;    (* Procedure BORRA *)
```

```
1018  23   29:0    56 ($$P$)
1019  23   12:0     0 BEGIN   (* Procedure INI_GRAF *)
1020  23   12:1     0    IF NOT(B_SALVA) THEN
1021  23   12:2     7    BEGIN
1022  23   12:3     7       DISP:=ORD(DISPO); (*  Decodifica el # de dispositivo      *)
1023  23   12:3    12       B_SALVA:=B_ECO;
1024  23   12:3    17       IF B_ECO THEN     (*  Si se pide eco grafico,            *)
1025  23   12:4    20       BEGIN             (*  abre el archivo                    *)
1026  23   12:5    20          REWRITE(ECO,NOM_AR_ECO);
1027  23   12:5    33          SALVA(0,DISP,0);       (* Indica el disp. seleccionado *)
1028  23   12:5    46                                 (*  con comando 0              *)
1029  23   12:4    46       END;(*ENDIF*)
1030  23   12:3    46       BORRA;            (*  Borra el disp seleccionado         *)
1031  23   12:3    48       PUERTO.PI.X:=0;   (*  Fija esq inf.izq. del puerto a 0,0  *)
1032  23   12:3    56       PUERTO.PI.Y:=0;
1033  23   12:3    64       IF DISPO=OSC THEN (*  Dependiendo del disp., fija la      *)
1034  23   12:4    69       BEGIN             (*  esquina superior derecha para usar *)
1035  23   12:5    69          PUERTO.PF.X:=OSCXMAX;  (* todo el dominio            *)
1036  23   12:5    79          PUERTO.PF.Y:=OSCYMAX;
1037  23   12:4    89       END
1038  23   12:3    89       ELSE
1039  23   12:4    91       BEGIN
1040  23   12:5    91          PUERTO.PF.X:=PLTXMAX;
1041  23   12:5   101          PUERTO.PF.Y:=PLTYMAX;
1042  23   12:4   111       END;
1043  23   12:4   111       (*  Selecciona ventana=puerto para calculo de la relacion *)
1044  23   12:3   111       SEL_VENTANA(0,0,PUERTO.PF.X,PUERTO.PF.Y);
1045  23   12:3   129       B_RANGO:=FALSE;
1046  23   12:3   134       B_ERR_ES:=FALSE
1047  23   12:2   134    END
1048  23   12:1   139    ELSE
1049  23   12:2   141       B_ERR_ES:=TRUE;
1050  23   12:0   146 END;    (* Procedure INI_GRAF *)
1051  23   12:0   160
1052  23   12:0   160
1053  23   13:0     1 PROCEDURE LEE_TAB_CAR;
1054  23   13:0    43
1055  23   13:0    43 (*************************************************************)
1056  23   13:0    43 (*                                                           *)
1057  23   13:0    43 (*      Esta rutina publica lee las tablas de definicion de  *)
1058  23   13:0    43 (*  caracteres y sus indices de acceso del archivo pedido.   *)
1059  23   13:0    43 (*      El nombre del archivo esta en AR_DEF_CAR y debe ser  *)
1060  23   13:0    43 (*  de exactamente 3 bloques de longitud.  Lee 128 indices de *)
1061  23   13:0    43 (*  acceso y 1024 trazos o vectores cortos.                  *)
1062  23   13:0    43 (*      Usa rutinas de lectura directa a disco y de movimiento *)
1063  23   13:0    43 (*  interno de bloques de memoria por rapidez.               *)
1064  23   13:0    43 (*      Recibe el parametro AR_DEF_CAR                       *)
1065  23   13:0    43 (*      Usa las var. globales IND_CAR, TAB_CAR, DEF_CAR      *)
1066  23   13:0    43 (*  y B_ERR_ES                                               *)
1067  23   13:0    43 (*                                                           *)
1068  23   13:0    43 (*************************************************************)
```

```
23   13:D    43 (**P*)
23   13:D    43 VAR
23   13:D    43    I              (*  Auxiliar para las lecturas de disco        *)
23   13:D    43       :INTEGER;
23   13:D    44
23   13:0     0 BEGIN   (* Procedure LEE_TAB_CAR *)
23   13:0     0    (*I-*)                       (* Deshabilita chequeo de IO     *)
23   13:1     0    RESET(DEF_CAR,AR_DEF_CAR);             (*  Abre para lectura    *)
23   13:1    17    (*I+*)                       (*  Habilita chequeo de IO       *)
23   13:1    17    B_ERR_ES:=TRUE;
23   13:1    22    IF IORESULT<>0 THEN          (*  Si no pudo abrir             *)
23   13:2    28       EXIT(LEE_TAB_CAR)
23   13:1    32    ELSE
23   13:2    34    BEGIN
23   13:3    34       B_ERR_ES:=FALSE;
23   13:3    39       I:=BLOCKREAD(DEF_CAR,TAB_CAR,1);  (* Lee el primer bloque *)
23   13:3    62       MOVELEFT(TAB_CAR,IND_CAR,256);    (*  Pasa 1/2 bloque a    *)
23   13:3    77                                         (* IND_CAR               *)
23   13:3    77       I:=BLOCKREAD(DEF_CAR,TAB_CAR,2,1);(* Lee los 2 siguientes *)
23   13:3    98                                         (* bloques en TAB_CAR    *)
23   13:3    98       CLOSE(DEF_CAR,LOCK)               (*  Cierra el archivo    *)
23   13:2   107    END (*ENDIF*)
23   13:0   107 END;    (* Procedure LEE_TAB_CAR *)
23   13:0   120
23   13:0   120
23   14:D     1 PROCEDURE TERM_GRAF;
23   14:D     2
23   14:D     2 (****************************************************************)
23   14:D     2 (*                                                              *)
23   14:D     2 (*      Esta rutina publica tiene el efecto de terminar las     *)
23   14:D     2 (*  transacciones graficas, cierra el archivo de eco, si existe *)
23   14:D     2 (*  y hace un cambio de hoja en el dispositivo empleado.        *)
23   14:D     2 (*      Usa las var. globales B_SALVA, ECO y DISP.              *)
23   14:D     2 (*      Usa las constantes PLTXMAX y PLTYMAX                    *)
23   14:D     2 (*      Llama a las rutinas BDDA y PLM_ARR.                     *)
23   14:D     2 (*      Recibe la var. BORRA                                    *)
23   14:D     2 (*                                                              *)
23   14:D     2 (****************************************************************)
23   14:D     2
23   14:0     0 BEGIN   (* Procedure TERM_GRAF *)
23   14:1     0    IF B_SALVA THEN      (*  Si hubo eco, cierra el archivo      *)
23   14:2     6    BEGIN
23   14:3     6       B_SALVA:=FALSE;
23   14:3    11       CLOSE(ECO,LOCK);
23   14:2    20    END;(*ENDIF*)
23   14:1    20    IF BORRA THEN
23   14:2    23       IF DISP=0 THEN     (* Para el osciloscopio, cambio de hoja *)
23   14:3    31          BORRA_OSC       (* es borrar la pantalla                *)
23   14:2    31       ELSE
23   14:3    35       BEGIN              (*  Para el plotter es mover la hoja    *)
23   14:4    35          MUEVE_ABS(PLTXMAX,PLTYMAX);
23   14:3    43       END;(*ENDIF*)
23   14:3    43    (*ENDIF*)
23   14:0    43 END;    (* Procedure TERM_GRAF *)
23   14:0    56
23   14:0    56 (*$I GRLIB04 *)
23   14:0    56 (*$I GRLIB05 *)
```

```
23   14:0   56 (*$P*)
23   14:0   56 (**********************************************************************)
23   14:0   56 (*                                                                    *)
23   14:0   56 (*      Este es el archivo GRLIB05 que se une al archivo             *)
23   14:0   56 (*  GRLIB00 en compilacion.                                          *)
23   14:0   56 (*                                                                    *)
23   14:0   56 (*                        Realizo:       Miguel Chin Auyon           *)
23   14:0   56 (*                                                                    *)
23   14:0   56 (*                        Fecha de creacion     :  23-mar-85         *)
23   14:0   56 (*                        Fecha de actualizacion:  25-mar-85         *)
23   14:0   56 (*                                                                    *)
23   14:0   56 (**********************************************************************)
23   14:0   56
23   15:D    1 PROCEDURE PROC_GRAF;
23   15:D    1
23   15:D    1 (**********************************************************************)
23   15:D    1 (*                                                                    *)
23   15:D    1 (*      Este procedimiento publico lee el archivo de comandos        *)
23   15:D    1 (*  graficos generados al optar por eco al iniciar la grafica       *)
23   15:D    1 (*  Tomara el puerto y dispositivo actualmente seleccionados y      *)
23   15:D    1 (*  la ventana se definira por la primera instruccion del           *)
23   15:D    1 (*  archivo de eco.                                                  *)
23   15:D    1 (*      Si se esta grabando en el momento de la llamada, un         *)
23   15:D    1 (*  archivo de eco, PROC_GRAF manda un mensaje de error.             *)
23   15:D    1 (*      Llama a las rutinas MUEVE_HACIA, LINEA_HACIA, PLM_ABJ,       *)
23   15:D    1 (*  PLM_ARR, SEL_VENTANA, SEL_TIPO y ESC_CAR.                        *)
23   15:D    1 (*      Usa las var. globales ECO, B_ERR_ES y NOM_AR_ECO.           *)
23   15:D    1 (*      Usa la funcion local REESC.                                  *)
23   15:D    1 (*                                                                    *)
23   15:D    1 (**********************************************************************)
23   15:D    1
23   30:D    3 FUNCTION REESC(ESCALA             (*  Escala leida del archivo     *)
23   30:D    3                     :INTEGER
23   30:D    3                 ):INTEGER;
23   30:D    4
23   30:D    4 (**********************************************************************)
23   30:D    4 (*                                                                    *)
23   30:D    4 (*      Esta funcion local a PROC_GRAF, reescala el valor de         *)
23   30:D    4 (*  escala pedido para los caracteres.                               *)
23   30:D    4 (*      Toma la funcion techo del escalamiento con respecto a        *)
23   30:D    4 (*  la menor distancia en el puerto.                                 *)
23   30:D    4 (*      Usa a var. global PUERTO.                                    *)
23   30:D    4 (*                                                                    *)
23   30:D    4 (**********************************************************************)
23   30:D    4
23   30:D    4 VAR
23   30:D    4    TEMP          (*  Almacena temporalmente el valor de REESC       *)
23   30:D    4       :REAL;
23   30:D    6
23   30:0    0 BEGIN   (* Function REESC *)
23   30:1    0    TEMP:=ESCALA*PUERTO.RL.X;
23   30:1   13    IF PUERTO.RL.X>PUERTO.RL.Y THEN
23   30:2   29       TEMP:=ESCALA*PUERTO.RL.Y;
23   30:2   42    (*ENDIF*)
23   30:1   42    IF TEMP=TRUNC(TEMP) THEN
23   30:2   57       REESC:=TRUNC(TEMP)
23   30:1   63    ELSE
23   30:2   67       REESC:=TRUNC(TEMP)+1;
23   30:2   77    (*ENDIF*)
23   30:0   77 END;    (* Function REESC *)
```

```
1185  23   30:0    90 ($$P$)
1186  23   30:0    90
1187  23   15:0     0 BEGIN   ($ Procedure PROC_GRAF $)
1188  23   15:1     0    B_ERR_ES:=FALSE;
1189  23   15:1     5    IF B_SALVA THEN      ($  Si se esta salvando actualmente     $)
1190  23   15:2    11    BEGIN
1191  23   15:3    11       B_ERR_ES:=TRUE;
1192  23   15:3    16       EXIT(PROC_GRAF)
1193  23   15:2    20    END;
1194  23   15:2    20    ($$I-$)              ($  Deshabilita chequeo de IO           $)
1195  23   15:1    20    RESET(ECO,NOM_AR_ECO);
1196  23   15:1    31    ($$I+$)              ($  Habilita chequeo de IO              $)
1197  23   15:1    31    IF IORESULT<>0 THEN  ($  Si hay error del archivo            $)
1198  23   15:2    37    BEGIN
1199  23   15:3    37       B_ERR_ES:=TRUE;
1200  23   15:3    42       EXIT(PROC_GRAF)
1201  23   15:2    46    END;
1202  23   15:1    46    IF ECO^.COMANDO<>0 THEN      ($ Si no es archivo grafico     $)
1203  23   15:2    54    BEGIN
1204  23   15:3    54       B_ERR_ES:=TRUE;
1205  23   15:3    59       CLOSE(ECO);
1206  23   15:3    68       EXIT(PROC_GRAF)
1207  23   15:2    72    END
1208  23   15:1    72    ELSE
1209  23   15:2    74    BEGIN
1210  23   15:3    74       IF ECO^.PARAM1=0 THEN
1211  23   15:4    82          SEL_VENTANA(0,0,OSCXMAX,OSCYMAX)
1212  23   15:3    94       ELSE
1213  23   15:4    98          SEL_VENTANA(0,0,PLTXMAX,PLTYMAX);
1214  23   15:3   112       MUEVE_HACIA(0,0);
1215  23   15:3   118       PLM_ARR;
1216  23   15:2   120    END;($ENDIF$)
1217  23   15:1   120    GET(ECO);
1218  23   15:1   128    WHILE NOT(EOF(ECO)) DO
1219  23   15:2   139    BEGIN
1220  23   15:3   139       WITH ECO^ DO
1221  23   15:4   144       BEGIN
1222  23   15:5   144          CASE COMANDO OF
1223  23   15:5   148          0:BEGIN
1224  23   15:7   148               B_ERR_ES:=TRUE;
1225  23   15:7   153               CLOSE(ECO);
1226  23   15:7   162               EXIT(PROC_GRAF)
1227  23   15:6   166             END;
1228  23   15:5   168          1:PLM_ARR;
1229  23   15:5   172          2:PLM_ABJ;
1230  23   15:5   176          3:LINEA_HACIA(PARAM1,PARAM2);
1231  23   15:5   186          4:SEL_TIPO(PARAM1);
1232  23   15:5   192          5:ESC_CAR(CHR(PARAM1),REESC(PARAM2),ARR);
1233  23   15:5   205          6:ESC_CAR(CHR(PARAM1),REESC(PARAM2),DER);
1234  23   15:5   218          7:ESC_CAR(CHR(PARAM1),REESC(PARAM2),ABJ);
1235  23   15:5   231          8:ESC_CAR(CHR(PARAM1),REESC(PARAM2),IZQ);
1236  23   15:5   244          9:MUEVE_HACIA(PARAM1,PARAM2);
1237  23   15:5   254          END;($ENDCASE$)
1238  23   15:4   282       END;($ENDWITH$)
1239  23   15:3   282       GET(ECO);
1240  23   15:2   290    END;($ENDWHILE$)
1241  23   15:1   292    CLOSE(ECO);
1242  23   15:0   301 END;    ($ Procedure PROC_GRAF $)
```

```
23   15:0   318 (*$P*)
23   15:0   318
23   16:D     3 FUNCTION DAT_GRAF;
23   16:D     4
23   16:D     4 (***************************************************************)
23   16:D     4 (*                                                             *)
23   16:D     4 (*       Esta funcion publica reporta los parametros del       *)
23   16:D     4 (*  paquete de generacion de graficos.  Hay 20 parametros que  *)
23   16:D     4 (*  indican el funcionamiento del sistema.                     *)
23   16:D     4 (*       Recibe la variable PREG y responde con un real.       *)
23   16:D     4 (*       Usa las var. globales PLM, DISP, TIPO, CM, CD, VENTANA, *)
23   16:D     4 (*  PUERTO, B_SALVA, B_RANGO y B_ERR_ES.                       *)
23   16:D     4 (*                                                             *)
23   16:D     4 (***************************************************************)
23   16:D     4
23   16:0     0 BEGIN   (* Function DAT_GRAF:REAL *)
23   16:1     0    CASE PREG OF
23   16:1     3     0: DAT_GRAF:=DISP;              (* Dispositivo OSC=0 PLT=1      *)
23   16:1    14     1: DAT_GRAF:=PLM DIV 2;         (* Pluma arriba=1 abajo=0       *)
23   16:1    27     2: DAT_GRAF:=TIPO;              (* Tipo de linea usado          *)
23   16:1    38     3: IF B_SALVA THEN              (* Salvando=1   sin eco=0       *)
23   16:3    44           DAT_GRAF:=1
23   16:2    46        ELSE
23   16:3    52           DAT_GRAF:=0;
23   16:1    60     4: DAT_GRAF:=OSCXMAX;           (* Maxima abscisa del OSC       *)
23   16:1    70     5: DAT_GRAF:=OSCYMAX;           (* Maxima ordenada del OSC      *)
23   16:1    80     6: DAT_GRAF:=PLTXMAX;           (* Maxima abscisa del PLT       *)
23   16:1    90     7: DAT_GRAF:=PLTYMAX;           (* Maxima ordenada del PLT      *)
23   16:1   100     8: DAT_GRAF:=CM.X;              (* Abscisa del cursor del usr   *)
23   16:1   111     9: DAT_GRAF:=CM.Y;              (* Ordenada del cursor del usr  *)
23   16:1   122    10: DAT_GRAF:=CD.X;              (* Abscisa del cursor de disp   *)
23   16:1   133    11: DAT_GRAF:=CD.Y;              (* Ordenada del cursor de disp  *)
23   16:1   144    12: DAT_GRAF:=VENTANA.PI.X;      (* Abscisa menor de la ventana  *)
23   16:1   156    13: DAT_GRAF:=VENTANA.PI.Y;      (* Ordenada menor de la vent    *)
23   16:1   168    14: DAT_GRAF:=VENTANA.PF.X;      (* Abscisa mayor de la ventana  *)
23   16:1   180    15: DAT_GRAF:=VENTANA.PF.Y;      (* Ordenada mayor de la vent    *)
23   16:1   192    16: DAT_GRAF:=PUERTO.PI.X;       (* Abscisa menor del puerto     *)
23   16:1   204    17: DAT_GRAF:=PUERTO.PI.Y;       (* Ordenada menor del puerto    *)
23   16:1   216    18: DAT_GRAF:=PUERTO.PF.X;       (* Abscisa mayor del puerto     *)
23   16:1   228    19: DAT_GRAF:=PUERTO.PF.Y;       (* Ordenada mayor del puerto    *)
23   16:1   240    20: DAT_GRAF:=PUERTO.RL.X;       (* Relacion abscisa prt/vent    *)
23   16:1   252    21: DAT_GRAF:=PUERTO.RL.Y;       (* Relacion ordenada prt/vent   *)
23   16:1   264    22: IF B_RANGO THEN
23   16:3   270           DAT_GRAF:=1               (* Fuera de la ventana=1        *)
23   16:2   272        ELSE
23   16:3   278           DAT_GRAF:=0;              (* Dentro de la ventana=0       *)
23   16:1   286    23: IF B_ERR_ES THEN
23   16:3   292           DAT_GRAF:=1               (* Error en la ultima ES=1      *)
23   16:2   294        ELSE
23   16:3   300           DAT_GRAF:=0;              (* Sin error en la ultima ES=0  *)
23   16:1   308    END;(*ENDCASE*)
23   16:0   364 END;   (* Function DAT_GRAF:REAL *)
23   16:0   380
23   16:0   380 (*$I GRLIB05 *)
```

```
1297  23   16:0   380 (*$P*)
1298  23   16:0   380
1299  23   16:0   380 (*##########################################################################*)
1300  23   16:0   380 (*                                                                        *)
1301  23   16:0   380 (*       Bloque de la rutina principal o de inicio global                 *)
1302  23   16:0   380 (*       del paquete de biblioteca.                                       *)
1303  23   16:0   380 (*                                                                        *)
1304  23   16:0   380 (*##########################################################################*)
1305  23   16:0   380
1306  23   16:0   380
1307  23    1:0     0 BEGIN
1308  23    1:0     0
1309  23    1:1     0    OSC_MOD:=0;                 (*  Selecciona el modo normal   *)
1310  23    1:1     5                                (*  de operacion del osc        *)
1311  23    1:1     5    TIPO:=0;                    (*  Selecciona tipo de linea    *)
1312  23    1:1    10                                (*  continua                    *)
1313  23    1:1    10    B_SALVA:=FALSE;             (*  No esta salvando            *)
1314  23    1:1    15    B_ERR_ES:=FALSE;            (*  No ha habido error en ES    *)
1315  23    1:1    20    B_RANGO:=FALSE;             (*  Cursor en ventana           *)
1316  23    1:1    25    INICIA_PIA;                 (*  Inicia la tarjeta           *)
1317  23    1:1    27    INI_GRAF(OSC,FALSE);        (*  Selecciona el osciloscopio  *)
1318  23    1:1    31                                (*  y deselecciona el eco       *)
1319  23    1:1    31    LEE_TAB_CAR('*:SYSCAR');    (*  Lee los carac. normales     *)
1320  23    1:1    44
1321   1    1:0    44 END.
```

```
PAGE -  0
Current memory available:   9128
0000:                           .NOMACROLIST
0000:
0000:                       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000:                       ;
0000:                       ;   RUTINAS PARA EL MANEJO DEL OSCILOSCOPIO
0000:                       ;                Y DEL PLOTTER.
0000:                       ;
0000:                       ;
0000:                       ;                       MIGUEL CHIN AUYON
0000:                       ;
0000:                       ;                       21-ENE-85
0000:                       ;
0000:                       ;                       ACTUALIZACION
0000:                       ;
0000:                       ;                       28-FEB-85
0000:                       ;
0000:                       ;
0000:                       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000:
0000:
0000:                       ;-----------------
0000:                       ;
0000:                       ;  CONSTANTES
0000:                       ;
0000:
0000: C0B0                  PORA1   .EQU  0C0B0     ;REG DE DATOS DEL PUERTO A1
0000: C0B1                  PORB1   .EQU  PORA1+1   ;REG DE DATOS DEL PUERTO B1
0000: C0B2                  PORA2   .EQU  PORA1+2   ;REG DE DATOS DEL PUERTO A2
0000: C0B3                  PORB2   .EQU  PORA1+3   ;REG DE DATOS DEL PUERTO B2
0000:
0000: C0B0                  PDRA1   .EQU  PORA1     ;REG DE SENTIDO DE A1
0000:
0000: C0B4                  PCRA1   .EQU  PORA1+4   ;REG DE CONTROL DE A1
0000: C0B5                  PCRB1   .EQU  PORA1+5   ;REG DE CONTROL DE B1
0000: C0B6                  PCRA2   .EQU  PORA1+6   ;REG DE CONTROL DE A2
0000: C0B7                  PCRB2   .EQU  PORA1+7   ;REG DE CONTROL DE B2
0000:
0000: FFFE                  IRQVEC  .EQU  0FFFE     ;VECTOR DE INTERRUPCION ENMASCARABLE
0000:
0000:                       ;====================
0000:                       ;
0000:                       ;  MACROS DE CONT.ASM
0000:                       ;
0000:
0000:                       .INCLUDE MAC1.ASM
0000:                       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000:                       ;
0000:                       ;       MACROS DE CONT.ASM
0000:                       ;
0000:                       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0000:
0000:
0000:
0000:                       ;-----------------

PAGE -  1           FILE:MAC1.ASM.TEXT
```

```
0000!                   ;
0000!                   ;  PON DIRECCION
0000!                   ;
0000!                   ;  SUBE LA PALABRA DE LA DIRECCION AL STACK
0000!                   ;
0000!
0000!                           .MACRO PON
0000!                           LDA Z1+1        ; SUBE Z1+1 PRIMERO AL STACK
0000!                           PHA
0000!                           LDA  Z1         ; DESPUES SUBE Z1 AL STACK
0000!                           PHA
0000!                           .ENDM
0000!
0000!                   ;-----------------
0000!                   ;
0000!                   ;  TOMA DIRECCION
0000!                   ;
0000!                   ;  BAJA UNA PALABRA DEL STACK A LA DIRECCION
0000!                   ;
0000!
0000!                           .MACRO TOMA
0000!                           PLA             ; BAJA EL TOPE DEL STACK A Z1
0000!                           STA  Z1
0000!                           PLA             ; BAJA EL SIG. BYTE A Z1+1
0000!                           STA  Z1+1
0000!                           .ENDM
0000!
0000!
0000!                   ;-----------------
0000!                   ;
0000!                   ;  ESPERA LOCALIDAD
0000!                   ;
0000!                   ;  ESPERA A QUE UNA LOCALIDAD CAMBIE A 0
0000!                   ;
0000!
0000!                           .MACRO ESPERA
0000!                           ; ESTE ESPACIO ES PARA PODER ETIQUETAR LA LLAMADA
0000!                   @01     LDA  Z1         ; REPITE HASTA QUE Z1
0000!                           BNE  @01        ; SEA 0.
0000!                           .ENDM
0000!
0000!
0000!
0000!
0000!
0000!                   ;-----------------
0000!                   ;
0000!                   ;  VARIABLES GLOBALES CON PASCAL
0000!                   ;
0000!                           .PUBLIC PLM,DISP,OSCMOD
0000!                           .PRIVATE POSPLM
0000!
0000!
2 blocks for procedure code  8400 words left

PAGE -  2  INICIAPI  FILE:GRLIB


0000!                           .PROC INICIAPIA
Current memory available:   8351
```

```
0000;                   ;=======================
0000;                   ;
0000;                   ;  PROCEDIMIENTO DE INICIACION DEL PIA Y
0000;                   ;  RUTINA DE SERVICIO DE INTERRUPCION.
0000;                   ;
0000;                   ;  PROCEDURE INICIAPIA;
0000;                   ;
0000;
0000;                   ;-------------------
0000;                   ;
0000;                   ;  LLENA DATO,ORIGEN,LONGITUD
0000;                   ;
0000;                   ;  LLENA UN AREA DE TAMA&O=LONGITUD MENOR O IGUAL A 256 Y
0000;                   ;  MAYOR A 0 DESDE ORIGEN CON DATO.
0000;                   ;  PARA LOGRAR UN AREA DE 256, LONGITUD=0
0000;                   ;
0000;
0000;                         .MACRO LLENA
0000;                         LDA  %1
0000;                         LDX  %3
0000;                   $01   STA  %2-1,X
0000;                         DEX
0000;                         BNE  $01
0000;                         .ENDM
0000;
0000;                   ;-------------------
0000;                   ;
0000;                   ;  SALVA
0000;                   ;
0000;                   ;  SALVA LOS REGISTROS AL STACK
0000;                   ;
0000;
0000;                         .MACRO SALVA
0000;                         PHA
0000;                         TXA
0000;                         PHA
0000;                         TYA
0000;                         PHA
0000;                         .ENDM
0000;
0000;                   ;-------------------
0000;                   ;
0000;                   ;  RECUPERA
0000;                   ;
0000;                   ;  RECUPERA LOS REGISTROS DEL STACK
0000;                   ;
0000;
0000;                         .MACRO RECUPERA
0000;                         PLA
0000;                         TAY
0000;                         PLA
0000;                         TAX
0000;                         PLA

PAGE -  3  INICIAPI  FILE:GRLIB


0000;                         .ENDM
0000;
0000;
```

```
0000:                           .DEF OSCRDY,PLTRDY
0000:                           .REF PXY,PLSPLT
0000:
0000:
0000:                           LLENA 0,PCRA1,4   ;INICIA REGS DE CONTROL
000A:                           LLENA OFF,PDRA1,4 ;FIJA EL SENTIDO COMO SALIDA
0014:                           LLENA 34,PCRA1,4  ;FIJA REGS DE CONTROL
001E:                           LLENA 0,PDRA1,4   ;INICIA REGS DE DATOS
0028:                           PON  APUNTA       ;TRANSFIERE EL VECTOR DE LA RUTINA DE
0030:                           TOMA IRQVEC       ;SERVICIO DE INTERRUPCION DE LA TARJETA
0038:                                             ;SOBRE EL VECTOR DEL SISTEMA
0038: A9 35                     LDA  #35
003A: 8D B4C0                   STA  PCRA1        ;HABILITA INTERRUPCIONES
003D: A9 37                     LDA  #37
003F: 8D B6C0                   STA  PCRA2        ;DE LA TARJETA
0042: 58                        CLI               ;HABILITA INTERRUPCIONES DEL CPU
0043: A9 00                     LDA  #00
0045: 8D ####                   STA  OSCRDY       ;BORRA OSCRDY
0048: 8D ####                   STA  PLTRDY       ; Y PLTRDY
004B: 8D 0000                   STA  OSCMOD       ;PON OSC EN NORMAL
004E: A9 02                     LDA  #02          ;INDICA QUE LA PLUMA ESTA
0050: 8D 0000                   STA  POSPLM       ;LEVANTADA
0053: 8D 0000                   STA  PXY          ;Y SUBELA
0056: 20 0000                   JSR  PLSPLT
0059: 60                        RTS               ;REGRESA A PASCAL
005A:
005A:                   ;-----------------
005A:                   ;
005A:                   ; AREA DE VARIABLES RELOCALIZABLES
005A:                   ;
005A:
002D# 5A00
002F# 5B00
005A: ####              APUNTA .WORD SIRQ          ;APTOR A LA RUTINA DE INTERRUPCIONES
0046# 5C00
005C: 00                OSCRDY .BYTE 0            ;STATUS DEL OSC
0049# 5D00
005D: 00                PLTRDY .BYTE 0            ;STATUS DEL PLT
005E:
005E:
005E:                   SIRQ   SALVA              ;SALVA REGS DE CPU
005A# 5E00
0063: 2C B4C0                  BIT  PCRA1         ;AVERIGUA QUIEN PROVOCA LA INTERRUPCION
0066: 30##                     BMI  OSCINT        ;SALTA SI FUE EL OSCILOSCOPIO
0068: 2C B6C0                  BIT  PCRA2
006B: 30##                     BMI  PLTINT        ;SALTA SI FUE EL PLOTTER
006D:                   REGRESA RECUPERA          ;RECUPER REGS DE CPU
0072: 40                        RTI               ;REGRESA AL PROGRAMA PRINCIPAL
0073:
0066# 0B
0073: AD B0C0           OSCINT LDA  PDRA1         ;BORRA INTERRUPCION
0076: A9 00                    LDA  #0

PAGE -  4  INICIAPI  FILE:GRLIB


0078: 8D 5C00                  STA  OSCRDY        ;AVISA QUE EL OSC ESTA LISTO
0078: F0F0                     BEQ  REGRESA
007D:
006B# 10
```

```
007D: AD B2C0              PLTINT  LDA  PDRA2       ;BORRA INTERRUPCION
0080: A9 00                        LDA  #0
0082: 8D 5D00                      STA  PLTRDY      ;AVISA QUE EL PLOTTER ESTA LISTO
0085: F0E6                         BEQ  REGRESA
0087:
0087:
```

PAGE - 5 INICIAPI FILE:GRLIB SYMBOLTABLE DUMP

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

```
APUNTA  LB 005A:  DISP    PB ----:  ESPERA  MC ----:  INICIAPI PR ----:  IROVEC  AB FFFE:
LLENA   MC ----:  OSCINT  LB 0073
OSCRDY  PB ----:  OSCRDY  DF 005C:  PCRA1   AB C0B4:  PCRA2    AB C0B6:  PCRB1   AB C0B5:
PCRB2   AB C0B7:  PDRA1   AB C0B0
PDRA2   AB C0B2:  PDRB1   AB C0B1:  PDRB2   AB C0B3:  PLM      PB ----:  PLSPLT  RF ----:
PLTINT  LB 007D:  PLTRDY  DF 005D
PUN     MC ----:  PORA1   AB C0B0:  POSPLM  PV ----:  PXY      RF ----:  RECUPERA MC ----:
REGRESA LB 006D:  SALVA   MC ----
SIRB    LB 005E:  TOMA    MC ----:
```

PAGE - 6 INICIAPI FILE:GRLIB

Current minimum space is 8148 words

PAGE - 7 BORRAOSC FILE:GRLIB

```
0000:                          .PROC BORRAOSC
Current memory available:    8351
0000:                    ;=======================
0000:                    ;
0000:                    ;  PROCEDIMIENTO QUE BORRA LA PANTALLA DEL OSCILOSCOPIO
0000:                    ;
0000:                    ;  PROCEDURE BORRAOSC;
0000:                    ;
0000:
0000:                          .REF  OSCRDY
0000:
0000: 58                       CLI              ;HABILITA INTERRUPCIONES
0001:                          ESPERA OSCRDY    ;ESPERA A QUE ESTE LIBRE EL OSC
0006: A9 3D                    LDA  #3D         ;MANDA LA SEÑAL AL OSCILIOSCOPIO
0008: 8D B4C0                  STA  PCRA1       ;PARA QUE COMIENCE A BORRARSE
000B: A9 35                    LDA  #35
000D: 8D B4C0                  STA  PCRA1
0010: A9 01                    LDA  #1          ;MARCA EL USO DEL OSC
0012: 8D 0000                  STA  OSCRDY
0015:                          ESPERA OSCRDY    ;ESPERA A QUE ESTE LIBRE EL OSC
001A: 60                       RTS              ;RETORNA A PASCAL
001B:
```

PAGE - 8 BORRAOSC FILE:GRLIB SYMBOLTABLE DUMP

AB - Absolute LB - Label UD - Undefined MC - Macro

```
RF - Ref        DF - Def      PR - Proc        FC - Func
PB - Public     PV - Private  CS - Consts

BORRAOSC PR ----;  DISP    PB ----;  ESPERA   MC ----;  IRQVEC   AB FFFE;  OSCMOD   PB ----;
OSCRDY   RF ----;  PCRA1   AB C0B4
PCRA2    AB C0B6;  PCRB1   AB C0B5;  PCRB2    AB C0B7;  PDRA1    AB C0B0;  PDRA2    AB C0B2;
PDRB1    AB C0B1;  PDRB2   AB C0B3
PLM      PB ----;  PON     MC ----;  PORA1    AB C0B0;  POSPLM   PV ----;  TOMA     MC ----;

PAGE -   9  BORRAOSC  FILE:GRLIB


Current minimum space is 8148 words

PAGE -  10  INIPLT    FILE:GRLIB


0000:                           .PROC INIPLT
Current memory available:    8351
0000:                          ;=======================
0000:                          ;
0000:                          ;  PROCEDIMIENTO QUE INICIA EL PLOTTER AL ORIGEN
0000:                          ;
0000:                          ;  PROCEDURE INIPLT;
0000:                          ;
0000:
0000:                               .REF PLTRDY,PXY,PLSPLT
0000:
0000: 58                            CLI             ;HABILITA INTERRUPCIONES
0001: A9 02                         LDA  #2         ;SUBE LA PLUMILLA
0003: 8D 0000                       STA  PXY
0006: 20 0000                       JSR  PLSPLT
0009: A9 04                         LDA  #4
000B: 8D 0000                       STA  PXY
000E: A2 0B                         LDX  #11.       ;11 PLGS/HOJA
0010: A0 C8                   PULG  LDY  #200.      ;200 PTS/PULG
0012: 8A                      PUNTO TXA
0013: 48                            PHA
0014: 20 0000                       JSR  PLSPLT
0017: 68                            PLA
0018: AA                            TAX
0019: 88                            DEY
001A: D0F6                          BNE  PUNTO      ;SI NO HA TERMINADO LA PULG. REGRESA
001C: CA                            DEX
001D: D0F1                          BNE  PULG       ;SI NO HA TERMINADO LA HOJA, REGRESA
001F: 60                            RTS             ;REGRESA A PASCAL
0020:
0020:
0020:                          .INCLUDE MAC2.ASM
0020:                          ;=======================
0020:                          ;
0020:                          ;  MAC2.ASM
0020:                          ;
0020:                          ;  ENCABEZADO DE BODA Y MACRO RUTINAS
0020:                          ;
0020:
0020:

PAGE -  11  INIPLT    FILE:MAC2.ASM.TEXT  SYMBOLTABLE DUMP
```

```
AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF - Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts


DISP    PB ----;  ESPERA  MC ----;  INIPLT  PR ----;  IRQVEC  AB FFFE;  OSCMOD  PB ----;
PCRA1   AB C0D4;  PCRA2   AB C0B6
PCRB1   AB C0B5;  PCRB2   AB C0B7;  PDRA1   AB C0B0;  PDRA2   AB C0B2;  PDRB1   AB C0B1;
PDRB2   AB C0B3;  PLM     PB ----
PLSPLT  RF ----;  PLTRDY  RF ----;  PON     MC ----;  PORA1   AB C0B0;  POSPLM  PV ----;
PULG    LB 0010;  PUNTO   LB 0012
PXY     RF ----;  TOMA    MC ----;

PAGE -  12  INIPLT    FILE:MAC2.ASM.TEXT


Current minimum space is 8148 words

PAGE -  13  BDDA      FILE:MAC2.ASM.TEXT


0000;                          .PROC BDDA,5
Current memory available:   8351
0000;                       ;##################################
0000;                       ;
0000;                       ;  BDDA(X1,Y1,X2,Y2,TIPO:INTEGER);
0000;                       ;
0000;                       ;  ESTA RUTINA TRAZA UNA LINEA EN EL DISPOSITIVO INDICADO
0000;                       ;  POR LA VARIABLE DISP. EN EL OSCILOSCOPIO LA RESOLUCION
0000;                       ;  SERA DE 2048@2048 Y EN EL GRAFICADOR INCREMENTAL X-Y
0000;                       ;  SERA DE 2200@2200 maximo.
0000;                       ;
0000;
0000;
0000;                       ;----------------------------
0000;                       ;
0000;                       ;  ABS16 X,Y -> Y:=SGN(X), X:=ABS(X)
0000;                       ;
0000;                       ;  ESTE MACRO OBTIENE EL SIGNO Y VALOR ABSOLUTO DE UN NUMERO
0000;                       ;  ENTERO EN 16 BITS, 2'C
0000;                       ;
0000;                             .MACRO ABS16
0000;                             LDA  X1+1        ;VERIFICA EL MSB DE X1
0000;                             BMI  $01         ;SI ES <0, EL NUMERO ES <0
0000;                             BEQ  $02         ;SI ES 0,SALTA A VERIFICAR EL LSB
0000;                       $04   LDA  #20         ;ES >0, ALMACENA EN X2
0000;                             STA  X2          ;UN 1 EN 16 BITS, CORRIDO 5 BITS
0000;                             LDA  #00         ;PARA MANEJAR 11 BITS DE RESOLUCION
0000;                             STA  X2+1
0000;                             BEQ  $03
0000;                       $01   LDA  #0E0        ;ES <0, ALMACENA EN X2, -1
0000;                             STA  X2          ;EN 16 BITS, 2'C, CORRIDO EN 5 BITS
0000;                             LDA  #0FF        ;PARA MANEJAR 11 BITS DE RESOLUCION
0000;                             STA  X2+1
0000;                             BNE  $03
0000;                       $02   LDA  X1          ;SI EL LSB DE X1 ES <>0, X1 ES >0
0000;                             BNE  $04
0000;                             STA  X2          ;SI ES 0, SI SIGNO ES 0
```

```
0000!                   STA  X2+1
0000!             $03   LDA  X2          ;SI FUE X1<0,
0000!                   BPL  $05
0000!                   SEC            ;HAZ X1=-X1 EN 16 BITS
0000!                   LDA  #0
0000!                   SBC  X1
0000!                   STA  X1
0000!                   LDA  #0
0000!                   SBC  X1+1
0000!                   STA  X1+1
0000!             $05
0000!                   .ENDM
0000!
0000!             ;-----------------------------
0000!             ;
0000!             ;  ADAPTA X -> X1=X*32
0000!             ;
0000!             ;  ESTE MACRO MULTIPLICA X POR 32 EN 16 BITS

PAGE -  14  BDDA     FILE:MAC2.ASM.TEXT


0000!             ;
0000!
0000!                   .MACRO ADAPTA
0000!                   LDX  #05       ;HAZ UN CORRIMIENTO DE
0000!             $01   ASL  X1        ;5 BITS EN EL VALOR DE
0000!                   ROL  X1+1      ;16 BITS
0000!                   DEX
0000!                   BNE  $01
0000!                   .ENDM
0000!
0000!             ;-----------------------------
0000!             ;
0000!             ;  ADD16 X,Y -> X1=X+Y
0000!             ;
0000!             ;  ESTE MACRO SUMA DOS CANTIDADES EN 16 BITS
0000!             ;
0000!
0000!                  .MACRO ADD16
0000!                  CLC             ;CORRIGE EL ACARREO
0000!                  LDA  X1         ;SUMA LOS DOS LSB
0000!                  ADC  X2
0000!                  STA  X1         ;ALMACENA EL RESULTADO EN EL LSB DE X1
0000!                  LDA  X1+1       ;SUMA LOS MSB
0000!                  ADC  X2+1
0000!                  STA  X1+1       ;ALMACENA EL RESULTADO EN EL MSB DE X1
0000!                  .ENDM
0000!
0000!             ;-----------------------------
0000!             ;
0000!             ;  COMP16 X,Y -> CARRY:=X)=Y
0000!             ;
0000!             ;  ESTE MACRO HACE UNA COMPARACION DE >= ENTRE DOS CANTIDADES DE 16
 BITS
0000!             ;  EL RESULTADO APARECE EN EL ACARREO
0000!             ;
0000!
0000!                  .MACRO COMP16
0000!                  LDA  X1+1       ;COMPARA LOS MSB
```

```
0000!                           CMP  X2+1
0000!                           BNE  $01
0000!                           LDA  X1             ;SI LOS MSB SON IGUALES,
0000!                           CMP  X2             ;COMPARA LOS LSB
0000!                   $01
0000!                           .ENDM
0000!
0000!                   ;-----------------------------
0000!                   ;
0000!                   ;  DEC16 X -> X1=X-1
0000!                   ;
0000!                   ;  ESTE MACRO DECREMENTA UNA CANTIDAD DE 16 BITS
0000!                   ;
0000!
0000!                           .MACRO DEC16
0000!                           SEC                 ;CORRIGE EL ACARREO
0000!                           LDA  X1             ;RESTA 1 EN EL LSB
0000!                           SBC  01

PAGE -  15  BDOS     FILE:MAC2.ASM.TEXT


0000!                           STA  X1
0000!                           LDA  X1+1           ;RESTA EL ACARREO EN EL MSB
0000!                           SBC  00
0000!                           STA  X1+1
0000!                           .ENDM
0000!
0000!                   ;-----------------------------
0000!                   ;
0000!                   ;  M216 X,Y -> X1=2*Y
0000!                   ;
0000!                   ;  ESTE MACRO MULTIPLICA UNA CANTIDAD POR 2
0000!                   ;
0000!
0000!                           .MACRO M216
0000!                           LDA  X2             ;HAZ UN CORRIMIENTO EN EL LSB
0000!                           ASL  A
0000!                           STA  X1
0000!                           LDA  X2+1           ;HAZ UNA ROTACION EN EL MSB
0000!                           ROL  A
0000!                           STA  X1+1
0000!                           .ENDM
0000!
0000!                   ;-----------------------------
0000!                   ;
0000!                   ;  SUB16 X,Y -> X1=X-Y
0000!                   ;
0000!
0000!                           .MACRO SUB16
0000!                           SEC                 ;CORRIGE EL ACARREO
0000!                           LDA  X1             ;RESTA LOS LSB
0000!                           SBC  X2
0000!                           STA  X1             ;ALMACENA EN X1
0000!                           LDA  X1+1           ;RESTA LOS MSB
0000!                           SBC  X2+1
0000!                           STA  X1+1           ;ALMACENA EN X1+1
0000!                           .ENDM
0000!
0000!
```

```
0000!
0000!
0000!                    .INCLUDE CONT1.ASM
0000!                   ;
0000!                   ;       ARCHIVO MCA2:CONT1.ASM
0000!                   ;
0000!                   ;       INCLUIDO EN MCA2:CONT.ASM
0000!                   ;
0000!                   ;               MIGUEL CHIN AUYON
0000!                   ;
0000!                   ;               CREACION: 6-FEB-85
0000!                   ;
0000!                   ;               ACTUALIZADO:4-MAR-85
0000!                   ;
0000!
0000!
0000!                           .REF  PLTRDY,OSCRDY

PAGE -  16  BODA      FILE:CONT1.ASM.TEXT


0000!                           .DEF  PXY,PLSPLT
0000!
0000!                   ;-------------------------
0000!                   ;
0000!                   ;  VARIABLES
0000!                   ;
0000!
0000! 0000              DX      .EQU 00
0000! 0002              DY      .EQU DX+2
0000! 0004              DX2     .EQU DY+2
0000! 0006              DY2     .EQU DX2+2
0000! 0008              R       .EQU DY2+2
0000! 000A              CONT    .EQU R+2
0000! 000C              PX      .EQU CONT+2
0000! 000E              PY      .EQU PX+2
0000! 0010              RETORNO .EQU PY+2
0000! 0012              SX      .EQU RETORNO+2
0000! 0014              SY      .EQU SX+2
0000! 0016              TIPO    .EQU SY+2
0000! 0018              CTINT   .EQU TIPO+2
0000! 0002              PLMINT  .EQU PLMINT+2
0000!
0000! 58                       CLI             ;ACTIVA INTERRUPCIONES
0001!                          TOMA RETORNO    ;SALVA DIR DE REGRESO
0007!                          TOMA TIPO       ;RECUPERA TIPO
000D!                          TOMA DY         ;RECUPERA Y2
0013!                          TOMA DX         ;RECUPERA X2
0019!                          TOMA PY         ;RECUPERA Y1
001F!                          TOMA PX         ;RECUPERA X1
0025! A5 16                    LDA  TIPO       ;INICIA EL CONTADOR
0027! 85 18                    STA  CTINT      ;DE TIPO DE LINEA
0029! A9 01                    LDA  #1         ;INTERNAMENTE LA PLUMA INICIA
002B! 8D 0200                  STA  PLMINT     ;BAJADA PARA TIPO<>0 Y PLM=1
002E!                          SUB16 DY,PY     ;DY:=Y2-Y1
0038!                          SUB16 DX,PX     ;DX:=X2-X1
0048!                          ABS16 DX,SX     ;SX:=SGN(DX)
004A! 0C
004C! 14
0060! 08
```

```
0056: 12
006C: 0B
007B:                                   ;DX:=ABS(DX)
007B:                   ABS16 DY,SY     ;SY:=SGN(DY)
007D: 0C
007F: 14
0093: 0B
0089: 12
007F: 0B
00AE:                                   ;DY:=ABS(DY)
00AE:                   M216 DX2,DX     ;DX2:=2*DX
00B8:                   M216 DY2,DY     ;DY2:=2*DY
00C2:                   COMP16 DX,DY    ;ACARREO:=DX>=DY
00C6: 04
00CC: 9018              BCC  FUERTE
00CE:                   PON  DX

PAGE -  17  BODA      FILE:CONT1.ASM.TEXT


00D4:                   TOMA R          ;R:=DX
00DA:                   PON  DX
00E0:                   TOMA CONT       ;CONT:=DX
00E6: AD 0000           LDA  DISP
00E9: F018              BEQ  R1         ;LINEA DE PENDIENTE SUAVE AL OSC
00EB: 4C 8888           JMP  R2         ;LINEA DE PENDIENTE SUAVE AL PLT
00EE:                 FUERTE PON  DY
00CC: 20
00F4:                   TOMA R          ;R:=DY
00FA:                   PON  DY
0100:                   TOMA CONT       ;CONT:=DY
0106: AD 0000           LDA  DISP
0109: D018              BNE  $001
010B: 4C 8888           JMP  R3         ;LINEA DE PENDIENTE FUERTE AL OSC
0109: 03
010E: 4C 8888         $001   JMP  R4    ;LINEA DE PENDIENTE FUERTE AL PLT
0111:
0111:                 ;=============================
0111:                 ;
0111:                 ;  VARIABLE GLOBAL A EXTERNAS
0111:                 ;
0111: 00              PXY     .BYTE  00
0112:
0112:
0112:                 ;=============================
0112:                 ;
0112:                 ;  RUTINA PARA TRAZAR LINEAS DE PENDIENTE
0112:                 ;  SUAVE EN EL OSCILOSCOPIO.
0112:                 ;
0112:
00E9: 27
0112: AD 0000         R1      LDA  PLM        ;VERIFICA LA POS DE LA PLUMA
0115: C9 02                   CMP  #02        ;SI ESTA LEVANTADA,NO HAY TRAZO
0117: F018                    BEQ  R1FIN
0119:                         ESPERA OSCRDY
011E:                         ADAPTA PX       ;ADECUA DATOS DEL CANAL X
0127:                         ADAPTA PY       ;ADECUA DATOS DEL CANAL Y
0130: 20 8888         $002    JSR  OSCPT      ;TRANSMITE UN PUNTO AL OSC
0133:                         ADD16 PX,SX     ;ACTUALIZA EL VALOR DE X
0140:                         ADD16 R,DY2     ;GENERA PASO VIRTUAL EN CANAL Y
```

```
0140:                           CMP16 R,DX2     ;SI HAY SOBREFLUJO, ACTUALIZA EN
0151: 04
0157: 90##                      BCC  $001       ;CANAL Y
0159:                           ADD16 PY,SY
0166:                           SUB16 R,DX2     ;ACTUALIZA EL DETECTOR DE SOBREFLUJO
0173:                   $001    DEC16 CONT      ;DECREMENTA CONTADOR HASTA
0179: 1A
0180: A5 0B                     LDA  CONT+1     ;CONT<0
0182: 10AC                      BPL  $002
0178: 6B
0184: 4C ####           RIFIN   JMP  RPAS
0187:
0187:                   ;=============================
0187:                   ;
0187:                   ; RUTINA PARA TRAZAR LINEAS DE PENDIENTE

PAGE -  18  BDDA      FILE:CONT1.ASM.TEXT


0187:                   ; FUERTE EN EL OSCILOSCOPIO.
0187:                   ;
0187:
010C# 8701
0187: AD 0000           RJ      LDA  PLM        ;VERIFICA LA POS DE LA PLUMA
018A: C9 02                     CMP  #02        ;SI ESTA LEVANTADA,NO HAY TRAZO
018C: F0##                      BEQ  RJFIN
018E:                           ESPERA OSCRDY
0193:                           ADAPTA PX       ;ADECUA DATOS DEL CANAL X
019C:                           ADAPTA PY       ;ADECUA DATOS DEL CANAL Y
01A3: 20 ####           $002    JSR  OSCPT      ;TRANSMITE UN PUNTO AL OSC
01A8:                           ADD16 PY,SY     ;ACTUALIZA EL VALOR DE Y
01B5:                           ADD16 R,DX2     ;GENERA PASO VIRTUAL EN CANAL X
01C2:                           CMP16 R,DY2     ;SI HAY SOBREFLUJO, ACTUALIZA EN
01C8: 04
01CC: 90##                      BCC  $001       ;CANAL X
01CE:                           ADD16 PX,SX
01DB:                           SUB16 R,DY2     ;ACTUALIZA EL DETECTOR DE SOBREFLUJO
01E8:                   $001    DEC16 CONT      ;DECREMENTA CONTADOR HASTA
01CC# 1A
01F5: A5 0B                     LDA  CONT+1     ;CONT<0
01F7: 10AC                      BPL  $002
018C# 6B
01F9: 4C ####           RJFIN   JMP  RPAS
01FC:
01FC:                   ;=============================
01FC:                   ;
01FC:                   ; RUTINA PARA PONER UN PUNTO
01FC:                   ; EN EL OSCILOSCOPIO.
01FC:                   ;
01FC:
01A6# FC01
0131# FC01
01FC: A5 16             OSCPT   LDA  TIPO       ;SI EL TIPO ES 0,
01FE: F0##                      BEQ  $003       ;ES TRAZO CONTINUO
0200: C6 1B                     DEC  CTINT      ;SI NO, DECREMENTA EL CONTADOR
0202: 10##                      BPL  $003       ;INTERMO HASTA QUE CTINT<0
0204: 85 1B                     STA  CTINT      ;Y REESTABLECE CTINT=TIPO
0206: AD 0200                   LDA  PLMINT     ;POR CADA CICLO DE CTINT, CAMBIA
0209: 49 01                     EOR  #1         ;LA PLUMA ENTRE 0 Y 1
020B: 8D 0200                   STA  PLMINT
```

```
0202# 0A
01FE# 0E
020E: A5 0D             $003    LDA  PX+1        ;MSB DEL CANAL X
0210: 8D B0C0                   STA  PORA1
0213: A5 0F                     LDA  PY+1        ;MSB DEL CANAL Y
0215: 8D B2C0                   STA  PORA2
0218: A5 0E                     LDA  PY          ;LSB DEL CANAL Y
021A: 4A                        LSR  A
021B: 4A                        LSR  A
021C: 4A                        LSR  A
021D: 4A                        LSR  A
021E: 05 0C                     ORA  PX          ;LSB DEL CANAL X
0220: 8D B1C0                   STA  PORB1
0223: AD 0200                   LDA  PLMINT      ;SI LA PLUMA VA LEVANTADA,

PAGE -  19  BDDA     FILE:CONT1.ASM.TEXT


0226: F0##                      BEQ  $001        ;NO HAY TRAZO (NO DISPARA Z)
0228: A9 3F                     LDA  #3F
022A: 8D B6C0                   STA  PCRA2       ;ACTIVA EL EJE Z DE OSC
022D: A9 37                     LDA  #37
022F: 8D B6C0                   STA  PCRA2       ;DESACTIVA EL EJE Z DE OSC
0232# 0A
0232: 60                $001    RTS
0233:
0233:                   ;=============================
0233:                   ;
0233:                   ; RUTINA PARA TRAZAR LINEAS DE PENDIENTE
0233:                   ; SUAVE EN EL GRAFICADOR INCREMENTAL X-Y
0233:                   ;
0233:
00EC# 3302
0233: AD 0000           R2      LDA  PLM         ;SI HAY QUE MOVER
0236: CD 0000                   CMP  POSPLM      ;LA PLUMA, SOLO SE HACE
0239: F0##                      BEQ  $001        ;UNA VEZ POR CAMBIO
023B: 8D 1101                   STA  PXY
023E: 20 ####                   JSR  PLSPLT
0239# 06
0241: 20 ####           $001    JSR  GENMASK     ;TRANSFORMA SX Y SY EN CODIGOS
0244:                                            ;PARA EL GRAFICADOR
0244:                           DEC16 CONT
0251: AD 1101                   LDA  PXY         ;SI NO SE PIDE MOVIMIENTO,
0254: F0##                      BEQ  $003        ;REGRESA A PASCAL
0256: A5 0C             $004    LDA  PX          ;SIEMPRE MUEVETE EN X
0258: 8D 1101                   STA  PXY
025B:                           ADD16 R,DY2      ;GENERA UN PASO VIRTUAL EN Y
0268:                           COMP16 R,DX2
026C# 04
0272: 90##                      BCC  $002        ;HASTA DETECTAR SOBREFLUJO
0274: AD 1101                   LDA  PXY         ;PARA MOVER FISICAMENTE SOBRE Y
0277: 05 0E                     ORA  PY          ;SIMULTANEAMENTE CON X
0279: 8D 1101                   STA  PXY
027C:                           SUB16 R,DX2      ;ACTUALIZA EL DETECTOR DE SOBREFLUJO
0272# 15
0289: 20 ####           $002    JSR  PLTPT       ;EFECTUA EL MOVIMIENTO
028C:                           DEC16 CONT       ;DECREMENTA CONT HASTA
0299: A5 0B                     LDA  CONT+1      ;CONT<0
029B: 1089                      BPL  $004
029D: AD 0000                   LDA  PLM         ;SI LA PLUMA FUE LEVANTADA, NO
```

```
02A0: C9 02                     CMP  #2         ;NECCSITA CHECAR EL TIPO.
02A2: F0##                      BEQ  $003
02A4: A5 16                     LDA  TIPO       ;SI EL TRAZO FUE CONTINUO, NO SE
02A6: F0##                      BEQ  $003       ;VERIFICA EL ESTADO ACTUAL DE PLMINT
02A8: AD 0200                   LDA  PLMINT     ;SI LA PLUMA FUE LEVANTADA, TERMINA
02AB: D0##                      BNE  $003       ;CON LA PLUMA BAJADA.
02AD: A9 01                     LDA  #1
02AF: 8D 1101                   STA  PXY
02B2: 20 ####                   JSR  PLSPLT
02AB# 08
02A6# 0B
02A2# 11
0254# 5F

PAGE -  20  BODA      FILE:CONT1.ASM.TEXT


02B5: 4C ####           $003    JMP  RPAS
02B#:
02B#:
02B#:                   ;============================
02B#:                   ;
02B#:                   ;  RUTINA PARA TRAZAR LINEAS DE PENDIENTE
02B#:                   ;  FUERTE EN EL GRAFICADOR INCREMENTAL X-Y
02B#:                   ;
02B#:
010F# B802
02B#: AD 0000           R4      LDA  PLM        ;SI HAY QUE MOVER
02BB: CD 0000                   CMP  POSPLM     ;LA PLUMA, SOLO SE HACE
02BE: F0##                      BEQ  $001       ;UNA VEZ POR CAMBIO
02C0: 8D 1101                   STA  PXY
02C3: 20 ####                   JSR  PLSPLT
02BE# 06
02C6: 20 ####           $001    JSR  GENMASK    ;TRANSFORMA SX Y SY EN CODIGOS
02C9:                           ;PARA EL GRAFICADOR
02C9:                   DEC16 CONT
02D6: AD 1101                   LDA  PXY        ;SI NO SE PIDE MOVIMIENTO,
02D9: F0##                      BEQ  $003       ;REGRESA A PASCAL
02DB: A5 0E             $004    LDA  PY         ;SIEMPRE MUEVETE EN Y
02DD: 8D 1101                   STA  PXY
02E0:                           ADD16 R.DX2     ;GENERA UN PASO VIRTUAL EN X
02ED:                           COMP16 R.DY2
02F1# 04
02F7: 90##                      BCC  $002       ;HASTA DETECTAR SOBREFLUJO
02F9: AD 1101                   LDA  PXY        ;PARA MOVER FISICAMENTE SOBRE Y
02FC: 05 0C                     ORA  PX         ;SIMULTANEAMENTE CON X
02FE: 8D 1101                   STA  PXY
0301:                           SUB16 R.DY2     ;ACTUALIZA EL DETECTOR DE SOBREFLUJO
02F7# 15
030E: 20 ####           $002    JSR  PLTPT      ;EFECTUA EL MOVIMIENTO
0311:                   DEC16 CONT              ;DECREMENTA CONT HASTA
031E: A5 0B                     LDA  CONT+1     ;CONT<0
0320: 1089                      BPL  $004
0322: AD 0000                   LDA  PLM        ;SI LA PLUMA FUE LEVANTADA, NO
0325: C9 02                     CMP  #2         ;NECESITA CHECAR EL TIPO.
0327: F0##                      BEQ  $003
0329: A5 16                     LDA  TIPO       ;SI EL TRAZO FUE CONTINUO, NO SE
032B: F0##                      BEQ  $003       ;VERIFICA EL ESTADO ACTUAL DE PLMINT
032D: AD 0200                   LDA  PLMINT     ;SI LA PLUMA FUE LEVANTADA, TERMINA
0330: D0##                      BNE  $003       ;CON LA PLUMA BAJADA.
```

```
0332: A9 01                     LDA  #1
0334: 8D 1101                   STA  PXY
0337: 20 ####                   JSR  PLSPLT
0330# 08
032B# 0B
0327# 11
02D9# 5F
033A: 4C ####           $003    JMP  RPAS
033D:
033D:
033D:                   ;===============================
033D:                   ;
```

PAGE - 21 BDDA FILE:CONT1.ASM.TEXT

```
033D:                   ; RUTINA PARA GENERAR LAS MASCARAS
033D:                   ; DE CODIGO PARA EL GRAFICADOR X-Y
033D:                   ;
033D:
02C7# 3D03
0242# 3D03
033D: A5 12             GENMASK LDA  SX         ;PARA X, DETECTA SI
033F: F0##                      BEQ  $01        ;EL MOVIMIENTO ES NULO,
0341: 10##                      BPL  $02        ;ES POSITIVO,
0343: A9 10                     LDA  #10        ;O ES NEGATIVO
0345: D0##                      BNE  $01
0341# 06
0347: A9 20             $02     LDA  #20
0345# 02
033F# 08
0349: 85 0C             $01     STA  PX         ;GUARDA LA MASCARA EN PX
034B: A5 14                     LDA  SY         ;PARA Y, DETECTA SI
034D: F0##                      BEQ  $03        ;EL MOVIMIENTO ES NULO,
034F: 10##                      BPL  $04        ;ES POSITIVO,
0351: A9 04                     LDA  #04        ;O ES NEGATIVO
0353: D0##                      BNE  $03
034F# 04
0355: A9 08             $04     LDA  #08
0353# 02
034D# 08
0357: 85 0E             $03     STA  PY         ;GUARDA LA MASCARA EN PY
0359: 05 0C                     ORA  PX         ;DETECTA SI EL MOVIMIENTO ES
035B: 8D 1101                   STA  PXY        ;NULO EN AMBAS DIRECCIONES
035E: 60                        RTS
035F:
035F:
035F:                   ;===============================
035F:                   ;
035F:                   ; RUTINA PARA PRODUCIR MOVIMIENTO
035F:                   ; EN EL GRAFICADOR X-Y
035F:                   ;
035F:
030F# 5F03
028A# 5F03
035F: AD 0000           PLTPT   LDA  PLM        ;SI EL TRAZO ES CON PLUMA LEVENTADA,
0362: C9 02                     CMP  02         ;NO VERIFICA EL TIPO
0364: F0##                      BEQ  PLSPLT
0366: A5 16                     LDA  TIPO       ;SI EL TRAZO ES CONTINUO NO VERIFICA CTIN
T
```

```
0368: F088                      BEQ  PLSPLT
036A: C6 18                     DEC  CTINT        ;DECREMENTA CTINT
036C: 1088                      BPL  PLSPLT       ;HASTA CTINT<0
036E: 85 18                     STA  CTINT        ;REESTABLECE CTINT=TIPO
0370: AD 0200                   LDA  PLMINT       ;ALTERNA PLMINT ENTRE 0 Y 1
0373: 49 01                     EOR  #1
0375: 8D 0200                   STA  PLMINT
0378: F088                      BEQ  $001
037A: AC 1101                   LDY  PIY          ;SI SE TIENE QUE BAJAR, SALVA PIY,
037D: 8D 1101                   STA  PIY          ;MANDA EL COMANDO EN PIY,
0380: 20 8888                   JSR  PLSPLT       ;Y MANDA LOS PULSOS
0383: 8C 1101                   STY  PIY

PAGE -  22  BDDA      FILE:CONT1.ASM.TEXT


0386: 4C 8888                   JMP  PLSPLT
0388: 0F
0389: AC 1101           $001    LDY  PIY          ;SI SE TIENE QUE SUBIR, SALVA PIY,
038C: A9 02                     LDA  #2           ;MANDA EL COMANDO EN PIY
038E: 8D 1101                   STA  PIY
0391: 20 8888                   JSR  PLSPLT       ;Y MANDA LOS PULSOS ADECUADOS
0394: 8C 1101                   STY  PIY
0397:
0397:                   ;=============================
0397:                   ;
0397:                   ;  ESTA RUTINA MANDA LOS PULSOS A PLT
0397:                   ;
0397:
0397:                   PLSPLT  ESPERA PLTRDY     ;HASTA QUE PLT ESTE LISTO
0392: 9703
0387: 9703
0381: 9703
036C: 29
0368: 2D
0364: 31
0338: 9703
02C4: 9703
02B3: 9703
023F: 9703
039C: AD 1101                   LDA  PIY          ;SI NO HAY COMANDO,NO SE
039F: F088                      BEQ  $001         ;MANDAN PULSOS
03A1: 0D 0000                   ORA  OSCMOD       ;RESPETA LAS CONDICIONES DE OSC
03A4: 8D B3C0                   STA  PORB2        ;TRANSMITE LAS SEÑALES
03A7: A9 01                     LDA  #1           ;MARCA LA TRANSMISION
03A9: 8D 0000                   STA  PLTRDY       ;SOBRE PLTRDY
03AC: A2 20                     LDX  #20          ;ESPERA 178 MICROSEG
03AE: CA                $002    DEX               ;PARA CERRAR EL PULSO
03AF: D0FD                      BNE  $002
03B1: A9 00                     LDA  #0
03B3: 0D 0000                   ORA  OSCMOD
03B6: 8D B3C0                   STA  PORB2
039F: 18
03B9: 60                $001    RTS
03BA:
03BA:                   ;=============================
03BA:                   ;
03BA:                   ;  REGRESO A PASCAL
03BA:                   ;
03BA:
```

```
033B# BA03
02D6# BA03
01FA# BA03
0185# BA03
03BA: AD 0000              RPAS   LDA PLM          ;ALMACENA LA POSICION DE LA PLUMA
03B0: 8D 0000                     STA POSPLM       ;PARA DETECTAR FUTUROS CAMBIOS
03C0:                             PON RETORNO
03C6: 60                          RTS
03C7:
03C7:
03C7:
```



```
03C7:
03C7:
03C7:
03C7:                             .END             ;FIN DE LAS RUTINAS EN ENSAMBLADOR
```



```
AB - Absolute     LB - Label      UD - Undefined    MC - Macro
RF - Ref          DF - Def        PR - Proc         FC - Func
PB - Public       PV - Private    CS - Consts


ABS16   MC ----: ADAPTA  MC ----: ADD16   MC ----: BDDA    PR ----: COMP16  MC ----:
CONT    AB 000A: CTINT   AB 0018
DEC16   MC ----: DISP    PB ----: DX      AB 0000: DX2     AB 0004: DY      AB 0002:
DY2     AB 0006: ESPERA  MC ----
FUERTE  LB 00EE: GENMASK LB 033D: IRQVEC  AB FFFE: M216    MC ----: OSCMOD  PB ----:
OSCPT   LB 01FC: OSCRDY  RF ----
PCRA1   AB C0B4: PCRA2   AB C0B6: PCRB1   AB C0B5: PCRB2   AB C0B7: PDRA1   AB C0B0:
PDRA2   AB C0B2: PDRB1   AB C0B1
PDRB2   AB C0B3: PLM     PB ----: PLMINT  LB 0002: PLSPLT  DF 0397: PLTPT   LB 035F:
PLTRDY  RF ----: PON     MC ----
PORA1   AB C0B0: POSPLM  PV ----: PX      AB 000C: PXY     DF 0111: PY      AB 000E:
R       AB 0008: R1      LB 0112
R1FIN   LB 0184: R2      LB 0233: R3      LB 0187: R3FIN   LB 01F9: R4      LB 02B8:
RETORNO AB 0010: RPAS    LB 03BA
SUB16   MC ----: SX      AB 0012: SY      AB 0014: TIPO    AB C016: TOMA    MC ----:
```



```
Current minimum space is 7360 words

Assembly complete:      1133 lines
    0  Errors flagged on this Assembly
```

```
 1    1:D    1 (*$L MCA3:PAS*)
 1    1:D    1 (*$S+*)
 1    1:D    1 UNIT GR2;
 1    1:D    1
 1    1:D    1 INTRINSIC CODE 25
25    1:D    1           DATA 26;
25    1:D    1
25    1:D    1 (****************************************************************)
25    1:D    1 (*                                                              *)
25    1:D    1 (* Estas rutinas conforman el paquete                           *)
25    1:D    1 (* de graficacion de funciones en 2-D                           *)
25    1:D    1 (*                               REVISION 20-MAR-1985           *)
25    1:D    1 (*                                   MIGUEL CHIN AUYON          *)
25    1:D    1 (*                                                              *)
25    1:D    1 (*                                                              *)
25    1:D    1 (****************************************************************)
25    1:D    1
25    1:D    1
25    1:D    1 (****************************************************************)
25    1:D    1
25    1:D    1
25    1:D    1 INTERFACE
25    1:D    1
25    1:D    1
23    1:D    1
23    1:D    1    TYPE
23    1:D    1       BYTE=0..255;
23    1:D    1       COM_GRAF=PACKED RECORD COMANDO:BYTE;PARAM1,PARAM2:INTEGER;END;
23    1:D    1       DISPOSITIVO= (OSC,PLT);
23    1:D    1       ORIENTACION= (ARR,DER,ABJ,IZQ);
23    1:D    1       PUNTO=RECORD X,Y:REAL;END;
23    1:D    1    VAR
23    1:D    1       CH:PUNTO;
23    1:D    5       NOM_AR_ECO:STRING[20];
23    1:D   16       DEF_CAR:FILE;
23    1:D   56       ECO:PACKED FILE OF COM_GRAF;
23    2:D    1    PROCEDURE PLM_ARR;
23    3:D    1    PROCEDURE PLM_ABJ;
23    4:D    1    PROCEDURE LINEA_HACIA( X,Y:REAL);
23    5:D    1    PROCEDURE LINEA( X1,Y1,X2,Y2:REAL);
23    6:D    1    PROCEDURE MUEVE_HACIA( X,Y:REAL);
23    7:D    1    PROCEDURE ESC_STR( STR:STRING;ESC:INTEGER;ORIENTA:ORIENTACION);
23    8:D    1    PROCEDURE ESC_CAR( CAR:CHAR;ESC:INTEGER;ORIENTA:ORIENTACION);
23    9:D    1    PROCEDURE SEL_VENTANA( X1,Y1,X2,Y2:REAL);
23   10:D    1    PROCEDURE SEL_PUERTO( X1,Y1,X2,Y2:REAL);
23   11:D    1    PROCEDURE SEL_TIPO( T:INTEGER);
23   12:D    1    PROCEDURE INI_GRAF( DISPO:DISPOSITIVO;B_ECO:BOOLEAN);
23   13:D    1    PROCEDURE LEE_TAB_CAR( AR_DEF_CAR:STRING);
23   14:D    1    PROCEDURE TERM_GRAF(BORRA:BOOLEAN);
23   15:D    1    PROCEDURE PROC_GRAF;
23   16:D    3    FUNCTION DAT_GRAF(PREG:INTEGER):REAL;
23    1:D    4    USES GRAFICAS,        (* Paquete de primitivas graficas 2D   *)
29    1:D    1
29    2:D    3   FUNCTION SIN(X:REAL):REAL;
29    3:D    3   FUNCTION COS(X:REAL):REAL;
29    4:D    3   FUNCTION EXP(X:REAL):REAL;
29    5:D    3   FUNCTION ATAN(X:REAL):REAL;
29    6:D    3   FUNCTION LN(X:REAL):REAL;
29    7:D    3   FUNCTION LOG(X:REAL):REAL;
29    8:D    3   FUNCTION SQRT(X:REAL):REAL;
```

```
29   8:D    5
25   1:D    5        TRANSCENDS;     (* Paquete de funciones tracendentales *)
25   1:D    1
25   1:D    1
25   1:D    1   VAR
25   1:D    1      TPRIMARIO,        (* Titulo primario de la grafica       *)
25   1:D    1      TSECUNDARIO,      (* Titulo secundario de la grafica     *)
25   1:D    1      TITULOX,          (* Titulo del eje X                    *)
25   1:D    1      TITULOY           (* Titulo del eje Y                    *)
25   1:D    1         :STRING;
25   1:D  165
25   1:D  165
25   2:D    3   FUNCTION DIREC(VAR VARIABLE  (* Regresa la direccion de una  *)
25   2:D    3                 ):INTEGER; (* variable de cualquier tipo   *)
25   2:D    4
25   3:D    1   PROCEDURE GRAF2(XAPT,        (* Direccion del arreglo X     *)
25   3:D    1                   YAPT,        (* Direccion del arreglo Y     *)
25   3:D    1                   N,           (* Numero de elementos X o Y   *)
25   3:D    1                   TGRAF,       (* Tipo de grafica             *)
25   3:D    1                   TLIN,        (* Tipo de linea a usar        *)
25   3:D    1                   OPCION       (* Opciones de la grafica      *)
25   3:D    1                      :INTEGER);
25   3:D    7
25   3:D    7
25   3:D    7 (*****************************************************************)
25   3:D    7
```

```
25   3:D     7 (*$P*)
25   1:D     7 IMPLEMENTATION
25   1:D   165
25   1:D   165
25   1:D   165 CONST
25   1:D   165
25   1:D   165    ASUP=100;     (*  Area de titulos superior de 100 int. de m.  *)
25   1:D   165    AINF=70;      (*  Area de titulos inferior de 70 int. de m.   *)
25   1:D   165    AIZQ=280;     (*  Area de titulos a la izquierda de 280 i.m.  *)
25   1:D   165    ADER=0;       (*  Area de titulos a la derecha de 0 int. m.   *)
25   1:D   165
25   1:D   165
25   1:D   165 (*$R-*)          (*  Deshabilita chequeo de rangos en arreglos   *)
25   1:D   165
25   1:D   165 (*$I GR201 *)
25   1:D   165
25   1:D   165 (****************************************************************)
25   1:D   165 (*                                                              *)
25   1:D   165 (*       Este es el archivo GR201 que es llamado por GR200 para  *)
25   1:D   165 (*  poder compilar.                                             *)
25   1:D   165 (*                                      Miguel Chin Auyon       *)
25   1:D   165 (*                                                              *)
25   1:D   165 (****************************************************************)
25   1:D   165
25   1:D   165 TYPE
25   1:D   165
25   1:D   165    VEC=ARRAY[1..1] OF REAL;
25   1:D   165
25   1:D   165    APTVEC=^VEC;
25   1:D   165
25   1:D   165
25   2:D     3 FUNCTION DIREC;
25   2:D     3
25   2:D     3 (****************************************************************)
25   2:D     3 (*                                                              *)
25   2:D     3 (*       Esta funcion esta programada en ensamblador y su       *)
25   2:D     3 (*  objetivo es obtener la direccion de cualquier variable,     *)
25   2:D     3 (*  de cualquier tipo.                                          *)
25   2:D     3 (*       Recibe de parametro la variable.                       *)
25   2:D     3 (*       Responde con un entero.                                *)
25   2:D     3 (*                                                              *)
25   2:D     3 (****************************************************************)
25   2:D     3
25   2:D     3    EXTERNAL;
25   2:D     3
```

```
25   2:D    3 (*$P*)
25   4:D    3 FUNCTION PISO(R:REAL     (*  Argumento de la funcion                *)
25   4:D    3                  ):INTEGER;
25   4:D    3
25   4:D    3 (*******************************************************************)
25   4:D    3 (*                                                                *)
25   4:D    3 (*      Esta funcion regresa en un entero la funcion PISO del      *)
25   4:D    3 (*  real R pasado como parametro.                                 *)
25   4:D    3 (*      La funcion piso es el entero menor mas cercano o igual    *)
25   4:D    3 (*  al parametro de la funcion.                                   *)
25   4:D    3 (*                                                                *)
25   4:D    3 (*******************************************************************)
25   4:D    3
25   4:D    3 VAR
25   4:D    3    N:REAL;
25   4:0    0 BEGIN
25   4:1    0    N:=TRUNC(R);           (*  Trunca R                            *)
25   4:1   11    IF N>R THEN
25   4:2   23       N:=N-1;
25   4:1   34    PISO:=TRUNC(N);
25   4:0   42 END;(* Fin del PROCEDURE PISO *)
25   4:0   54
25   4:0   54
25   5:D    3 FUNCTION SGN(U:REAL      (*  Argumento de la funcion                *)
25   5:D    3                  ):INTEGER;
25   5:D    3
25   5:D    3 (*******************************************************************)
25   5:D    3 (*                                                                *)
25   5:D    3 (*  Esta rutina regresa el signo del argumento                    *)
25   5:D    3 (*                                                                *)
25   5:D    3 (*******************************************************************)
25   5:D    3
25   5:0    0 BEGIN (* SGN *)
25   5:1    0    IF U<0 THEN
25   5:2   10       SGN:=-1
25   5:1   10    ELSE
25   5:2   16       IF U=0 THEN
25   5:3   26          SGN:=0
25   5:2   26          ELSE
25   5:3   31             SGN:=1;
25   5:0   34 END; (* SGN *)
25   5:0   46
25   5:0   46
25   6:D    3 FUNCTION POTDIEZ(N:INTEGER):REAL;
25   6:D    4
25   6:D    4 (*******************************************************************)
25   6:D    4 (*                                                                *)
25   6:D    4 (*  Extiende el rango de la funcion PWROFTEN del sistema          *)
25   6:D    4 (*                                                                *)
25   6:D    4 (*******************************************************************)
25   6:D    4
25   6:0    0 BEGIN (* POTDIEZ *)
25   6:1    0    IF N<0 THEN
25   6:2    5       POTDIEZ:=1.0/PWROFTEN(-N)
25   6:1   16    ELSE
25   6:2   23       POTDIEZ:=PWROFTEN(N);
25   6:0   30 END; (* POTDIEZ *)
```

```
25   6:0    42 (*$P*)
25   6:0    42
25   7:D     1 PROCEDURE STRR(N:REAL; VAR S:STRING ;NUMDIGIT:INTEGER);
25   7:D     5
25   7:D     5 (*******************************************************************)
25   7:D     5 (*                                                                 *)
25   7:D     5 (*   Este procedimiento transforma el real N en una cadena de      *)
25   7:D     5 (* caracteres en S empleando NUMDIGIT digitos decimales.           *)
25   7:D     5 (*                                                                 *)
25   7:D     5 (*******************************************************************)
25   7:D     5
25   7:D     5 VAR
25   7:D     5    EXPONENTE,DIGITO,EXPREDONDEO,I:INTEGER;
25   7:D     9    MANTISA:REAL;
25   7:D    11    DIGST:STRING;
25   7:D    52
25   7:0     0 BEGIN (*STRR*)
25   7:1     0    IF N=0 THEN
25   7:2    10       S:='0.0'
25   7:1    11    ELSE
25   7:2    21    BEGIN
25   7:3    21       IF N<0 THEN
25   7:4    31          S:='-'
25   7:3    32       ELSE
25   7:4    37          S:=' ';
25   7:4    41       (*endif*)
25   7:3    41       N:=ABS(N);
25   7:3    50       EXPONENTE:=PISO(LOG(N));
25   7:3    65       EXPREDONDEO:=EXPONENTE-(NUMDIGIT-1);
25   7:3    72       MANTISA:=(N+0.5*POTDIEZ(EXPREDONDEO))/
25   7:3    91                   POTDIEZ(EXPONENTE);
25   7:3    99       IF MANTISA>=10.0 THEN
25   7:4   114       BEGIN
25   7:5   114          EXPONENTE:=EXPONENTE+1;
25   7:5   119          MANTISA:=MANTISA/10.0
25   7:4   125       END;(*endif*)
25   7:3   135       DIGITO:=TRUNC(MANTISA);
25   7:3   143       STR(DIGITO,DIGST);
25   7:3   155       S:=CONCAT(S,DIGST,'.');
25   7:3   192       MANTISA:=(MANTISA-DIGITO)*10;
25   7:3   206       FOR I:=2 TO NUMDIGIT DO
25   7:4   218       BEGIN
25   7:5   218          DIGITO:=TRUNC(MANTISA);
25   7:5   226          STR(DIGITO,DIGST);
25   7:5   238          S:=CONCAT(S,DIGST);
25   7:5   263          MANTISA:=(MANTISA-DIGITO)*10
25   7:4   272       END;(*endfor*)
25   7:3   284       IF EXPONENTE<>0 THEN
25   7:4   289       BEGIN
25   7:5   289          STR(EXPONENTE,DIGST);
25   7:5   301          S:=CONCAT(S,'E',DIGST)
25   7:4   334       END(*endif*)
25   7:2   336    END(*endif*)
25   7:0   336 END; (* STRR *)
```

```
242  25    7:0   352 (*$P*)
243  25    7:0   352
244  25    8:D     1 PROCEDURE STRF(N:REAL; VAR S:STRING);
245  25    8:D     4
246  25    8:D     4 (*************************************************************************)
247  25    8:D     4 (*                                                                       *)
248  25    8:D     4 (*  Rutina de transformacion de real a dato formateado                   *)
249  25    8:D     4 (*                                                                       *)
250  25    8:D     4 (*                                                                       *)
251  25    8:D     4 (*************************************************************************)
252  25    8:D     4
253  25    8:D     4 VAR
254  25    8:D     4    K:INTEGER;
255  25    8:D     5    CH:STRING[1];
256  25    8:D     6
257  25    8:0     0 BEGIN (*STRF*)
258  25    8:1     0    CH:=' ';
259  25    8:1     5    IF N<0 THEN CH:='-';
260  25    8:1    20    N:=ABS(N);
261  25    8:1    29    K:=1;
262  25    8:1    32    WHILE (N<>0) AND (N<10) AND (K)-1) DO
263  25    8:2    56    BEGIN
264  25    8:3    56       K:=K-1;
265  25    8:3    61       N:=N*10
266  25    8:2    67    END;(*endwhile*)
267  25    8:1    74    IF N>99 THEN N:=TRUNC(N);
268  25    8:1    95    IF N=0 THEN
269  25    8:2   105       S:=' 0.0'
270  25    8:1   106    ELSE
271  25    8:2   117    BEGIN
272  25    8:3   117       STR(ROUND(N),S);
273  25    8:3   133       IF LENGTH(S)=1 THEN S:=CONCAT('0',S);
274  25    8:3   165       INSERT(CH,S,1);
275  25    8:3   173       INSERT('.',S,LENGTH(S)+K)
276  25    8:2   187    END(*endif*)
277  25    8:0   187 END; (* STRF *)
```

```
25   8:0   202 (*$P*)
25   8:0   202
25   9:D     1 PROCEDURE REDONDEA(VAR M:REAL);
25   9:D     2
25   9:D     2 (************************************************************************)
25   9:D     2 (*                                                                      *)
25   9:D     2 (*  Esta rutina busca el punto mas cercano a 10,5 o 2 escalado          *)
25   9:D     2 (*                                                                      *)
25   9:D     2 (************************************************************************)
25   9:D     2
25   9:D     2 VAR
25   9:D     2    R:REAL;
25   9:D     4    K:INTEGER;
25   9:D     5
25   9:0     0 BEGIN (* REDONDEA *)
25   9:1     0    R:=LOG(M);
25   9:1    12    K:=PISO(R);
25   9:1    22    R:=R-K;
25   9:1    33    R:=EXP(R*LN(10));
25   9:1    54    IF R>5 THEN
25   9:2    64       R:=10
25   9:1    66    ELSE
25   9:2    72       IF R>2 THEN
25   9:3    82          R:=5
25   9:2    84       ELSE
25   9:3    90          R:=2;
25   9:1    96    M:=R*POTDIEZ(K)
25   9:0   102 END; (* REDONDEA *)
25   9:0   122
25   9:0   122
25   9:0   122 (*$I GR201 *)
25   9:0   122 (*$I GR202 *)
```

```
25     9:0   122 (**P*)
25     9:0   122 (****************************************************************)
25     9:0   122 (*                                                              *)
25     9:0   122 (*  Este es el archivo GR202 que se añade a GR200               *)
25     9:0   122 (*                                       Miquel Chin Auyon      *)
25     9:0   122 (*                                                              *)
25     9:0   122 (****************************************************************)
25     9:0   122
25     9:0   122
25    10:D     1 PROCEDURE ETIQUETA;
25    10:D     1
25    10:D     1 (****************************************************************)
25    10:D     1 (*                                                              *)
25    10:D     1 (*  Esta rutina etiqueta los ejes en dos dimensiones            *)
25    10:D     1 (*                                                              *)
25    10:D     1 (****************************************************************)
25    10:D     1
25    10:D     1 VAR
25    10:D     1    ST:STRING;
25    10:D    42    X11,X12,X21,X22,Y21,Y22,PTR,
25    10:D    42    E,OFF,INF,SUP,POS,N,M:REAL;
25    10:D    70
25    11:D     1 PROCEDURE ESCOFF(XE,YE,XO,YO:REAL);
25    11:D     9
25    11:D     9 (****************************************************************)
25    11:D     9 (*                                                              *)
25    11:D     9 (*      Esta rutina escribe la escala en las coord. XE,YE y     *)
25    11:D     9 (*  el desplazamiento u offset en XO,YO.                        *)
25    11:D     9 (*                                                              *)
25    11:D     9 (****************************************************************)
25    11:D     9
25    11:0     0 BEGIN (* ESCOFF *)
25    11:1     0    IF E<>0 THEN
25    11:2    11    BEGIN
25    11:3    11       MUEVE_HACIA(XE,YE);
25    11:3    22       ESC_STR('E',3,ARR);
25    11:3    28       STR(TRUNC(E),ST);
25    11:3    47       ESC_STR(ST,4,ARR)
25    11:2    52    END;(*endif*)
25    11:1    55    IF OFF<>0 THEN
25    11:2    66    BEGIN
25    11:3    66       MUEVE_HACIA(XO,YO);
25    11:3    77       ESC_CAR('A',4,ARR);
25    11:3    83       STRR(OFF,ST,4);
25    11:3    94       ESC_STR(ST,4,ARR)
25    11:2    99    END(*endif*)
25    11:0   102 END; (* ESCOFF *)
```

```
356  25  11:0   114 (*$P*)
357  25  12:D     1 PROCEDURE PARAMETROS;
358  25  12:D     1
359  25  12:D     1 (*************************************************************************)
360  25  12:D     1 (*                                                                       *)
361  25  12:D     1 (*      Esta rutina escoge la escala y el desplazamiento.                *)
362  25  12:D     1 (*                                                                       *)
363  25  12:D     1 (*************************************************************************)
364  25  12:D     1
365  25  12:0     0 BEGIN (* PARAMETROS *)
366  25  12:1     0    POS:=INF;
367  25  12:1    10    OFF:=0;
368  25  12:1    17    E:=0;
369  25  12:1    24    IF SUP<>INF THEN
370  25  12:2    38    BEGIN
371  25  12:3    38       IF ABS(INF)/(SUP-INF)>20 THEN OFF:=INF;
372  25  12:3    72       E:=PISO(LOG(SUP-INF))
373  25  12:2    91    END
374  25  12:1    98    ELSE
375  25  12:2   100       IF SUP<>0 THEN
376  25  12:3   111          E:=PISO(LOG(SUP));
377  25  12:3   131    (*endif*)
378  25  12:1   131    IF ABS(E)<3 THEN E:=0;
379  25  12:1   150    M:=1.0/POTDIEZ(TRUNC(E))
380  25  12:0   167 END; (* PARAMETROS *)
381  25  12:0   186
382  25  13:D     1 PROCEDURE ETIQY;
383  25  13:D     1
384  25  13:D     1 (*************************************************************************)
385  25  13:D     1 (*                                                                       *)
386  25  13:D     1 (*      Esta rutina etiqueta el eje  vertical.                           *)
387  25  13:D     1 (*                                                                       *)
388  25  13:D     1 (*************************************************************************)
389  25  13:D     1
390  25  13:D     1 VAR
391  25  13:D     1    DY:REAL;
392  25  13:0     0 BEGIN (* ETIQY *)
393  25  13:1     0    DY:=65/DAT_GRAF(21);          (*  Distancia minima 65 pixeles *)
394  25  13:1    13    REDONDEA(DY);
395  25  13:1    17    INF:=PISO(DAT_GRAF(13)/DY)*DY;
396  25  13:1    43    SUP:=-PISO(-DAT_GRAF(15)/DY)*DY;
397  25  13:1    71    PARAMETROS;
398  25  13:1    73    WHILE POS<=SUP DO
399  25  13:2    87    BEGIN
400  25  13:3    87       M:=M*(POS-OFF);
401  25  13:3   109       STRF(M,ST);
402  25  13:3   119       MUEVE_HACIA(4,POS-14/DAT_GRAF(21));       (*  Centrado   *)
403  25  13:3   139       ESC_STR(ST,4,ARR);
404  25  13:3   147       POS:=POS+DY
405  25  13:2   155    END;(*endif*)
406  25  13:1   164    SEL_PUERTO(X21-AIZQ,Y22,X22+ADER,Y22+ASUP);
407  25  13:1   198    SEL_VENTANA(0,-0.2,35,3);
408  25  13:1   214    ESCOFF(0,0,5,0)
409  25  13:0   222 END; (* ETIQY *)
```

```
25   13:0   238 (**P*)
25   14:D     1 PROCEDURE ETIQX;
25   14:D     1
25   14:D     1 (*******************************************************************)
25   14:D     1 (*                                                                 *)
25   14:D     1 (*      Esta rutina etiqueta el eje  horizontal.                   *)
25   14:D     1 (*                                                                 *)
25   14:D     1 (*******************************************************************)
25   14:D     1
25   14:D     1 VAR
25   14:D     1    DX:REAL;
25   14:0     0 BEGIN (* ETIQX *)
25   14:1     0    DX:=130/DAT_GRAF(20);          (* Distancia minima 130 pixeles *)
25   14:1    15    REDONDEA(DX);
25   14:1    19    INF:=PISO(DAT_GRAF(12)/DX)*DX;
25   14:1    45    SUP:=-PISO(-DAT_GRAF(14)/DX)*DX;
25   14:1    73    PARAMETROS;
25   14:1    75    WHILE POS<=SUP DO
25   14:2    89    BEGIN
25   14:3    89       N:=M*(POS-OFF);
25   14:3   111       STRF(N,ST);
25   14:3   121       MUEVE_HACIA(POS-56/DAT_GRAF(20),1);        (*  Centrado    *)
25   14:3   141       ESC_STR(ST,4,ARR);
25   14:3   149       POS:=POS+DX
25   14:2   157    END;(*endif*)
25   14:1   166    SEL_PUERTO(X21-AIZQ,Y21-AINF,X22+ADER,Y21);
25   14:1   200    SEL_VENTANA(0,0,35,2);
25   14:1   211    ESCOFF(30,0,25,0)
25   14:0   219 END; (* ETIQX *)
25   14:0   236
25   10:0     0 BEGIN (* ETIQUETA *)
25   10:1     0    X11:=DAT_GRAF(12);             X12:=DAT_GRAF(14);
25   10:1    20    X21:=DAT_GRAF(16);             X22:=DAT_GRAF(18);
25   10:1    40    Y21:=DAT_GRAF(17);             Y22:=DAT_GRAF(19);
25   10:1    60    SEL_PUERTO(X21-AIZQ,Y21,X21,Y22);
25   10:1    84    SEL_VENTANA(0,DAT_GRAF(13),10,DAT_GRAF(15));
25   10:1   103    ETIQY;
25   10:1   105    PTR:=(1-42*LENGTH(TPRIMARIO)/(X22-X21+AIZQ+ADER))*17.5;
25   10:1   145    IF PTR<0 THEN PTR:=0;
25   10:1   161    MUEVE_HACIA(PTR,1.2);
25   10:1   175    ESC_STR(TPRIMARIO,6,ARR);     (* Escribe titulo primario      *)
25   10:1   183    PTR:=(1-28*LENGTH(TSECUNDARIO)/(X22-X21+AIZQ+ADER))*17.5;
25   10:1   223    IF PTR<10 THEN PTR:=10;
25   10:1   239    MUEVE_HACIA(PTR,0);
25   10:1   248    ESC_STR(TSECUNDARIO,4,ARR);  (* Escribe titulo secundario    *)
25   10:1   256    SEL_PUERTO(X21,Y21-AINF,X22,Y21);
25   10:1   278    SEL_VENTANA(X11,0,X12,2);
25   10:1   293    ETIQX;
25   10:1   295    PTR:=(1-28*LENGTH(TITULOX)/(X22-X21+AIZQ+ADER))*17.5;
25   10:1   335    IF PTR<0 THEN PTR:=0;
25   10:1   351    MUEVE_HACIA(PTR,0);
25   10:1   360    ESC_STR(TITULOX,4,ARR);       (* Escribe el titulo horizontal *)
25   10:1   368    SEL_PUERTO(X21-AIZQ,Y21-AINF,X21,Y22);
25   10:1   395    SEL_VENTANA(0,0,10,35);
25   10:1   406    PTR:=(1-28*LENGTH(TITULOY)/(Y22-Y21+AINF))*17.5;
25   10:1   441    IF PTR<0 THEN PTR:=0;
25   10:1   457    MUEVE_HACIA(1,PTR);
25   10:1   466    ESC_STR(TITULOY,4,IZQ)        (* Escribe el titulo vertical   *)
25   10:0   471 END; (* ETIQUETA*)
```

```
25   10:0   486 (*$P*)
25   15:D     1 PROCEDURE MALLA;
25   15:D     1
25   15:D     1 (*******************************************************************)
25   15:D     1 (*                                                                 *)
25   15:D     1 (*  Esta rutina traza la malla ajustandola a los parametros de     *)
25   15:D     1 (*  la rutina etiqueta.    Se llama por la opcion(bit2=0).         *)
25   15:D     1 (*                                                                 *)
25   15:D     1 (*******************************************************************)
25   15:D     1
25   15:D     1 VAR
25   15:D     1    INF,SUP,POS:REAL;
25   15:D     7    DX,DY:REAL;
25   15:0     0 BEGIN (* MALLA *)
25   15:1     0    SEL_TIPO(20);
25   15:1     4    DX:=130/DAT_GRAF(20);
25   15:1    19    REDONDEA(DX);
25   15:1    23    INF:=PISO(DAT_GRAF(12)/DX)*DX;
25   15:1    48    SUP:=-PISO(-DAT_GRAF(14)/DX)*DX;
25   15:1    75    POS:=INF;
25   15:1    83    WHILE POS<=SUP DO
25   15:2    95    BEGIN
25   15:3    95       IF POS<>0 THEN
25   15:4   105       BEGIN
25   15:5   105          MUEVE_HACIA(POS,DAT_GRAF(15));
25   15:5   118          LINEA_HACIA(POS,DAT_GRAF(13))
25   15:4   128       END;(*endif*)
25   15:3   131       POS:=POS+DX
25   15:2   137    END;(*endwhile*)
25   15:1   146    DY:=65/DAT_GRAF(21);
25   15:1   159    REDONDEA(DY);
25   15:1   163    INF:=PISO(DAT_GRAF(13)/DY)*DY;
25   15:1   188    SUP:=-PISO(-DAT_GRAF(15)/DY)*DY;
25   15:1   215    POS:=INF;
25   15:1   223    WHILE POS<=SUP DO
25   15:2   235    BEGIN
25   15:3   235       IF POS<>0 THEN
25   15:4   245       BEGIN
25   15:5   245          MUEVE_HACIA(DAT_GRAF(12),POS);
25   15:5   258          LINEA_HACIA(DAT_GRAF(14),POS)
25   15:4   268       END;(*endif*)
25   15:3   271       POS:=POS+DY
25   15:2   277    END;(*endwhile*)
25   15:0   286 END; (* MALLA *)
```

```
513 25   15:0   302 (*$P*)
514 25   16:0     1 PROCEDURE MARCA;
515 25   16:0     1
516 25   16:0     1 (*************************************************************************)
517 25   16:0     1 (*                                                                       *)
518 25   16:0     1 (*  Pone las marcas orillas del puerto                                  *)
519 25   16:0     1 (*                                                                       *)
520 25   16:0     1 (*************************************************************************)
521 25   16:0     1
522 25   16:0     1 VAR
523 25   16:0     1    DX,DY,POS,INFX,INFY,
524 25   16:0     1    SUPX,SUPY,INX,INY    :REAL;
525 25   16:0    19
526 25   16:0     0 BEGIN (* MARCA *)
527 25   16:1     0    SEL_TIPO(0);
528 25   16:1     4    DX:=20/DAT_GRAF(20);          (* Minimo espacio entre marcas  *)
529 25   16:1    17    REDONDEA(DX);
530 25   16:1    21    INFX:=PISO(DAT_GRAF(12)/DX)*DX;
531 25   16:1    46    SUPX:=-PISO(-DAT_GRAF(14)/DX)*DX;
532 25   16:1    73    POS:=INFX;
533 25   16:1    81    DY:=20/DAT_GRAF(21);          (* Minimo espacio entre marcas  *)
534 25   16:1    94    REDONDEA(DY);
535 25   16:1    98    INFY:=PISO(DAT_GRAF(13)/DY)*DY;
536 25   16:1   123    SUPY:=-PISO(-DAT_GRAF(15)/DY)*DY;
537 25   16:1   150    INX:=5/DAT_GRAF(21);
538 25   16:1   163    INY:=5/DAT_GRAF(20);
539 25   16:1   176    WHILE POS<SUPX DO    (* Marcas verticales *)
540 25   16:2   188    BEGIN
541 25   16:3   188       MUEVE_HACIA(POS,DAT_GRAF(13)+INX);
542 25   16:3   206       LINEA_HACIA(POS,DAT_GRAF(13));
543 25   16:3   219       POS:=POS+DX
544 25   16:2   225    END;(*endwhile*)
545 25   16:1   234    POS:=INFY;
546 25   16:1   242    WHILE POS<SUPY DO    (* Marcas horizontales *)
547 25   16:2   254    BEGIN
548 25   16:3   254       MUEVE_HACIA(DAT_GRAF(14),POS);
549 25   16:3   267       LINEA_HACIA(DAT_GRAF(14)-INY,POS);
550 25   16:3   285       POS:=POS+DY
551 25   16:2   291    END;(*endwhile*)
552 25   16:1   300    POS:=SUPX;
553 25   16:1   308    WHILE POS>INFX DO    (* Marcas verticales *)
554 25   16:2   320    BEGIN
555 25   16:3   320       MUEVE_HACIA(POS,DAT_GRAF(15)-INX);
556 25   16:3   338       LINEA_HACIA(POS,DAT_GRAF(15));
557 25   16:3   351       POS:=POS-DX
558 25   16:2   357    END;(*endwhile*)
559 25   16:1   366    POS:=SUPY;
560 25   16:1   374    WHILE POS>INFY DO    (* Marcas horizontales *)
561 25   16:2   386    BEGIN
562 25   16:3   386       MUEVE_HACIA(DAT_GRAF(12),POS);
563 25   16:3   399       LINEA_HACIA(DAT_GRAF(12)+INY,POS);
564 25   16:3   417       POS:=POS-DY
565 25   16:2   423    END(*endwhile*)
566 25   16:0   430 END; (* MARCAS *)
567 25   16:0   452
568 25   16:0   452 (*$I GR202 *)
568 25   16:0   452 (*$I GR203 *)
```

```
25   16:0   452 (*$P*)
25   16:0   452 (*****************************************************************)
25   16:0   452 (*                                                               *)
25   16:0   452 (*       Este archivo es GR203 que es llamado al momento de      *)
25   16:0   452 (*   compilar GR200.                                             *)
25   16:0   452 (*                                                               *)
25   16:0   452 (*                                          Miguel Chin Auyon    *)
25   16:0   452 (*                                                               *)
25   16:0   452 (*****************************************************************)
25   16:0   452
25   17:D     1 PROCEDURE EJES;
25   17:D     1
25   17:D     1 (*****************************************************************)
25   17:D     1 (*                                                               *)
25   17:D     1 (*   Esta rutina traza ejes en el plano                          *)
25   17:D     1 (*       y el marco de la ventana                                *)
25   17:D     1 (*                                                               *)
25   17:D     1 (*****************************************************************)
25   17:D     1
25   17:0     0 BEGIN (* EJES *)
25   17:1     0    SEL_TIPO(0);
25   17:1     4    LINEA(DAT_GRAF(12),0,DAT_GRAF(14),0);
25   17:1    23    LINEA(0,DAT_GRAF(13),0,DAT_GRAF(15));
25   17:1    42    MUEVE_HACIA(DAT_GRAF(12),DAT_GRAF(13));
25   17:1    57    LINEA_HACIA(DAT_GRAF(12),DAT_GRAF(15));
25   17:1    72    LINEA_HACIA(DAT_GRAF(14),DAT_GRAF(15));
25   17:1    87    LINEA_HACIA(DAT_GRAF(14),DAT_GRAF(13));
25   17:1   102    LINEA_HACIA(DAT_GRAF(12),DAT_GRAF(13))
25   17:0   114 END; (* EJES *)
25   17:0   130
25   17:0   130
25   18:D     1 PROCEDURE MAXMIN( X:APTVEC;  N:INTEGER; VAR MAX,MIN:REAL);
25   18:D     5
25   18:D     5 (*****************************************************************)
25   18:D     5 (*                                                               *)
25   18:D     5 (*       Este procedimiento busca el valor mayor y menor del     *)
25   18:D     5 (*   vector de N elementos reales apuntado por X                 *)
25   18:D     5 (*                                                               *)
25   18:D     5 (*****************************************************************)
25   18:D     5
25   18:D     5 VAR
25   18:D     5    I:INTEGER;
25   18:0     0 BEGIN (*MAXMIN*)
25   18:1     0    MAX:=X^[1];
25   18:1    11    MIN:=MAX;
25   18:1    17    IF N>2 THEN
25   18:2    22       FOR I:=2 TO N DO
25   18:3    33       BEGIN
25   18:4    33          IF X^[I]>MAX THEN
25   18:5    48             MAX:=X^[I];
25   18:4    59          IF X^[I]<MIN THEN
25   18:5    74             MIN:=X^[I]
25   18:3    81       END(*endfor*)
25   18:0    85 END; (*MAXMIN*)
25   18:0   106
```

```
25    18:0   106 (*$P*)
25    19:D     3 FUNCTION ESTBIT(N,BIT:INTEGER):BOOLEAN;
25    19:D     5
25    19:D     5 (*****************************************************************)
25    19:D     5 (*                                                               *)
25    19:D     5 (*       Esta funcion prueba el bit BIT del entero N.            *)
25    19:D     5 (*       Si es 0, regresa falso, si es 1, regresa verdadero      *)
25    19:D     5 (*                                                               *)
25    19:D     5 (*****************************************************************)
25    19:D     5
25    19:D     5 VAR
25    19:D     5    I:INTEGER;
25    19:0     0 BEGIN (*ESTBIT*)
25    19:1     0    I:=BIT;
25    19:1     3    IF NOT((I<0) OR (I>15)) THEN
25    19:2    13    BEGIN
25    19:3    13       WHILE I>0 DO
25    19:4    18       BEGIN
25    19:5    18          N:=N DIV 2;
25    19:5    23          I:=I-1
25    19:4    24       END;(*endwhile*)
25    19:3    30       IF (N MOD 2)=1 THEN
25    19:4    37          ESTBIT:=TRUE
25    19:3    37       ELSE
25    19:4    42          ESTBIT:=FALSE
25    19:2    42    END
25    19:1    45    ELSE
25    19:2    47       ESTBIT:=FALSE
25    19:0    47 END; (*ESTBIT*)
25    19:0    64
25    19:0    64
25     3:D     1 PROCEDURE GRAF2(*(XAPT,YAPT,N,TGRAF,TLIN,OPCION:INTEGER)*);
25     3:D     7
25     3:D     7 (*****************************************************************)
25     3:D     7 (*                                                               *)
25     3:D     7 (*       Este rutina es el punto de entrada al sistema.          *)
25     3:D     7 (*       Traza la grafica de una serie de N datos almacenados    *)
25     3:D     7 (*  en los vectores apuntados por XAPT y YAPT.                   *)
25     3:D     7 (*       TGRAF es el tipo de grafica que se pide, (0) si es      *)
25     3:D     7 (*  continua, (1) puntual y (2) por impulsos.  TLIN es el tipo   *)
25     3:D     7 (*  de linea que empleara.   OPCION maneja tres parametros, el   *)
25     3:D     7 (*  bit0 indica si la grafica sera X-Y(0) o c-Y(1) donde c es    *)
25     3:D     7 (*  cardinalidad, bit1 indica si se recalcula la ventana(0) o    *)
25     3:D     7 (*  si se mantiene (1), bit2 indica si se desea trazar(0) o      *)
25     3:D     7 (*  no (1) la malla de referencia.                               *)
25     3:D     7 (*                                                               *)
25     3:D     7 (*****************************************************************)
25     3:D     7
25     3:D     7 VAR
25     3:D     7    XIZQ,XDER,YINF,YSUP,
25     3:D     7    TEMP,MAX,MIN,MAX1,MIN1:REAL;
25     3:D    25    I:INTEGER;
25     3:D    26    X,Y:APTVEC;
```

```
25   3:D   28 (*$P*)
25  20:D    1 PROCEDURE GRVEC1;
25  20:D    1
25  20:D    1 (**********************************************************************)
25  20:D    1 (*                                                                    *)
25  20:D    1 (*      Esta rutina efectua el trazado de la grafica si es            *)
25  20:D    1 (*  de tipo c-Y donde c es la cardinalidad                            *)
25  20:D    1 (*                                                                    *)
25  20:D    1 (**********************************************************************)
25  20:D    1
25  20:D    1 VAR
25  20:D    1    I:INTEGER;
25  20:0    0 BEGIN
25  20:1    0    CASE TGRAF OF
25  20:1    5    0:BEGIN(*CONTINUA*)
25  20:3    5          MUEVE_HACIA(1,Y^[1]);
25  20:3   20          FOR I:=2 TO N DO
25  20:4   33             LINEA_HACIA(I,Y^[I])
25  20:2   45       END;
25  20:1   57    1:BEGIN(*PUNTOS*)
25  20:3   57          FOR I:=1 TO N DO
25  20:4   70          BEGIN
25  20:5   70             MUEVE_HACIA(I,Y^[I]);
25  20:5   85             LINEA_HACIA(I,Y^[I])
25  20:4   97          END(*endfor*)
25  20:2  100       END;
25  20:1  109    2:BEGIN(*IMPULSOS*)
25  20:3  109          FOR I:=1 TO N DO
25  20:4  122          BEGIN
25  20:5  122             MUEVE_HACIA(I,0);
25  20:5  129             LINEA_HACIA(I,Y^[I])
25  20:4  141          END(*endfor*)
25  20:2  144       END
25  20:1  151    END(*END CASE*)
25  20:0  166 END;(*FIN DE GRVEC1*)
```

```
712  25   20:0   186 (*$P*)
713  25   21:0     1 PROCEDURE GRVEC2;
714  25   21:0     1
715  25   21:0     1 (*************************************************************************)
716  25   21:0     1 (*                                                                       *)
717  25   21:0     1 (*      Esta rutina efectua el trazado de la grafica si es               *)
718  25   21:0     1 (*  de tipo X-Y , con ambos vectores existentes.                         *)
719  25   21:0     1 (*                                                                       *)
720  25   21:0     1 (*************************************************************************)
721  25   21:0     1
722  25   21:0     1 VAR
723  25   21:0     1   I:INTEGER;
724  25   21:0     0 BEGIN
725  25   21:1     0   CASE TGRAF OF
726  25   21:1     5   0:BEGIN(*CONTINUA*)
727  25   21:3     5         MUEVE_HACIA(X^[1],Y^[1]);
728  25   21:3    28         FOR I:=2 TO N DO
729  25   21:4    41           LINEA_HACIA(X^[I],Y^[I])
730  25   21:2    61       END;
731  25   21:1    73   1:BEGIN(*PUNTOS*)
732  25   21:3    73         FOR I:=1 TO N DO
733  25   21:4    86         BEGIN
734  25   21:5    86           MUEVE_HACIA(X^[I],Y^[I]);
735  25   21:5   109           LINEA_HACIA(X^[I],Y^[I])
736  25   21:4   129         END(*endfor*)
737  25   21:2   132       END;
738  25   21:1   141   2:BEGIN(*IMPULSOS*)
739  25   21:3   141         FOR I:=1 TO N DO
740  25   21:4   154         BEGIN
741  25   21:5   154           MUEVE_HACIA(X^[I],0);
742  25   21:5   169           LINEA_HACIA(X^[I],Y^[I])
743  25   21:4   189         END(*endfor*)
744  25   21:2   192       END
745  25   21:1   199   END(*END CASE*)
746  25   21:0   214 END;(*FIN DE GRVEC2*)
```

```
747  25   21:0  236 ($SP8)
748  25    3:0    0 BEGIN ($GRAF2$)
749  25    3:1    0    MOVELEFT(XAPT,X,2);
750  25    3:1    9    MOVELEFT(YAPT,Y,2);
751  25    3:1   18    XIZQ:=DAT_GRAF(16);
752  25    3:1   28    XDER:=DAT_GRAF(18);
753  25    3:1   38    YINF:=DAT_GRAF(17);
754  25    3:1   48    YSUP:=DAT_GRAF(19);
755  25    3:1   58    SEL_PUERTO(XIZQ+AIZQ,YINF+AINF,XDER-ADER,YSUP-ASUP);
756  25    3:1   91    IF NOT(ESTBIT(OPCION,1)) THEN
757  25    3:2  100    BEGIN
758  25    3:3  100       IF ESTBIT(OPCION,0) THEN
759  25    3:4  108       BEGIN
760  25    3:5  108          MAX1:=1;
761  25    3:5  114          MIN1:=N
762  25    3:4  116       END
763  25    3:3  120       ELSE
764  25    3:4  122          MAXMIN(X,N,MAX1,MIN1);
765  25    3:3  131       MAXMIN(Y,N,MAX,MIN);
766  25    3:3  140       IF TGRAF=3 THEN
767  25    3:4  145       BEGIN
768  25    3:5  145          IF (MAX>0) AND (MIN>0) THEN
769  25    3:6  164             MIN:=0;
770  25    3:5  170          IF (MAX<0) AND (MIN<0) THEN
771  25    3:6  189             MAX:=0
772  25    3:4  191       END;
773  25    3:3  195       TEMP:=MAX1-MIN1;
774  25    3:3  208       MIN1:=MIN1-0.10$TEMP;
775  25    3:3  228       MAX1:=MAX1+0.05$TEMP;
776  25    3:3  248       TEMP:=MAX-MIN;
777  25    3:3  261       MIN:=MIN-0.10$TEMP;
778  25    3:3  282       MAX:=MAX+0.05$TEMP;
779  25    3:3  302       SEL_VENTANA(MIN1,MIN,MAX1,MAX);
780  25    3:3  321       EJES;
781  25    3:3  323       MARCA
782  25    3:2  323    END;
783  25    3:1  325    IF NOT(ESTBIT(OPCION,2)) THEN
784  25    3:2  334       MALLA;
785  25    3:1  336    SEL_TIPO(TLIN);
786  25    3:1  340    IF ESTBIT(OPCION,0) THEN
787  25    3:2  348       GRVEC1
788  25    3:1  348    ELSE
789  25    3:2  352       GRVEC2;
790  25    3:1  354    IF NOT(ESTBIT(OPCION,1)) THEN
791  25    3:2  363    BEGIN
792  25    3:3  363       ETIQUETA;
793  25    3:3  365       SEL_VENTANA(MIN1,MIN,MAX1,MAX);
794  25    3:2  384    END;
795  25    3:1  384    SEL_PUERTO(XIZQ,YINF,XDER,YSUP);
796  25    3:0  403 END; ($GRAF2$)
797  25    3:0  418
798  25    3:0  418 ($$I GR203.$)
799  25    3:0  418
800  25    3:0  418 ($$R+$)          ($  Habilita chequeo de rangos en arreglos     $)
```

```
25   3:0   418 (*$P*)
25   3:0   418
25   3:0   418 (*********************************************************************)
25   3:0   418 (*                                                                   *)
25   3:0   418 (*      Esta seccion se ejecuta al inicio de cada programa que      *)
25   3:0   418 (*  emplea este paquete.   Simplemente inicia las variables         *)
25   3:0   418 (*  para la transmision de titulos a vacio.                         *)
25   3:0   418 (*                                                                   *)
25   3:0   418 (*********************************************************************)
25   3:0   418
25   1:0     0 BEGIN
25   1:1     0    TPRIMARIO:='';
25   1:1     8    TSECUNDARIO:='';
25   1:1    16    TITULOX:='';
25   1:1    24    TITULOY:='';
 1   1:0    32 END.
```

```
PAGE -  0
Current memory available:   9128
0000:                       ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0000:                       ;
0000:                       ;  Esta rutina en ensamblador se añade a GR200
0000:                       ; para darle la facilidad de obtener la direccion
0000:                       ; de cualquier variable.
0000:                       ;
0000:                       ;                       Miguel Chin Anyon
0000:                       ;
0000:
0000:                       ;-------------------
0000:                       ;
0000:                       ;  PON DIRECCION
0000:                       ;
0000:                       ;  SUBE LA PALABRA DE LA DIRECCION AL STACK
0000:                       ;
0000:
0000:                             .MACRO PON
0000:                             LDA Z1+1          ; SUBE Z1+1 PRIMERO AL STACK
0000:                             PHA
0000:                             LDA  Z1           ; DESPUES SUBE Z1 AL STACK
0000:                             PHA
0000:                             .ENDM
0000:
0000:                       ;-------------------
0000:                       ;
0000:                       ;  TOMA DIRECCION
0000:                       ;
0000:                       ;  BAJA UNA PALABRA DEL STACK A LA DIRECCION
0000:                       ;
0000:
0000:                             .MACRO TOMA
0000:                             PLA               ; BAJA EL TOPE DEL STACK A Z1
0000:                             STA  Z1
0000:                             PLA               ; BAJA EL SIG. BYTE A Z1+1
0000:                             STA  Z1+1
0000:                             .ENDM
0000:
2 blocks for procedure code  8563 words left

PAGE -  1  DIREC     FILE:CONFIG.GEM


0000:                             .FUNC DIREC,1
Current memory available:   8514
0000:                       ;
0000:                       ;  FUNCTION DIREC(VAR VARIABLE):INTEGER
0000:                       ;
0000: 0000                  RETORNO .EQU  0
0000:                             TOMA  RETORNO     ; DIRECCION DE RETORNO
0000: 68                    @     PLA
0001: 85 00                 @     STA  RETORNO
0003: 68                    @     PLA
0004: 85 01                 @     STA  RETORNO+1
0006: 68                          PLA
0007: 68                          PLA
0008: 68                          PLA
0009: 68                          PLA               ; 2 PALABRAS DE CEROS
000A:                             PON   RETORNO     ; DIRECCION DE RETORNO
```

```
000A: A5 01                 0       LDA RETORNO+1
000C: 48                    0       PHA
000D: A5 00                 0       LDA  RETORNO
000F: 48                    0       PHA
0010: 60                            RTS
0011:                               .END

PAGE -   2  DIREC     FILE:CONFIG.6EM  SYMBOLTABLE DUMP


AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF - Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts


DIREC    FC ----:  PON      MC ----:  RETORNO  AB 0000:  TONA     MC ----:

PAGE -   3  DIREC     FILE:CONFIG.6EM


Current minimum space is 8506 words

Assembly complete:        58 lines
    0   Errors flagged on this Assembly
```

```
 1     1:D     1 (*$L MCA3:PAS*)
 1     1:D     1 (****************************************************************)
 1     1:D     1 (*                                                              *)
 1     1:D     1 (*       Este programa permite manipular objetos en tres        *)
 1     1:D     1 (* dimensiones definidos en 'wireframe' en archivos en disco.   *)
 1     1:D     1 (* Puede seleccionar caracteristicas del dispositivo de salida. *)
 1     1:D     1 (* Las transformaciones se manejan con matrices homogeneas      *)
 1     1:D     1 (* para rotar, trasladar, escalar y hacer perspectiva.          *)
 1     1:D     1 (*       Reside en el archivo GR300 y llama a GR301 y a GR302.  *)
 1     1:D     1 (*                                                              *)
 1     1:D     1 (*                                    Miguel Chin Auyon         *)
 1     1:D     1 (*                                                              *)
 1     1:D     1 (****************************************************************)
 1     1:D     1
 1     1:D     1 (*$S+*)
 1     1:D     1
 1     1:D     1 PROGRAM GR3;
 1     1:D     3
23     1:D     3
23     1:D     3    TYPE
23     1:D     1       BYTE=0..255;
23     1:D     1       COM_GRAF=PACKED RECORD COMANDO:BYTE;PARAM1,PARAM2:INTEGER;END;
23     1:D     1       DISPOSITIVO= (DSC,PLT);
23     1:D     1       ORIENTACION= (ARR,DER,ABJ,IZQ);
23     1:D     1       PUNTO=RECORD X,Y:REAL;END;
23     1:D     1    VAR
23     1:D     1       CN:PUNTO;
23     1:D     5       NOM_AR_ECO:STRING[20];
23     1:D    16       DEF_CAR:FILE;
23     1:D    56       ECO:PACKED FILE OF COM_GRAF;
23     2:D     1    PROCEDURE PLM_ARR;
23     3:D     1    PROCEDURE PLM_ABJ;
23     4:D     1    PROCEDURE LINEA_HACIA( X,Y:REAL);
23     5:D     1    PROCEDURE LINEA( X1,Y1,X2,Y2:REAL);
23     6:D     1    PROCEDURE MUEVE_HACIA( X,Y:REAL);
23     7:D     1    PROCEDURE ESC_STR( STR:STRING;ESC:INTEGER;ORIENTA:ORIENTACION);
23     8:D     1    PROCEDURE ESC_CAR( CAR:CHAR;ESC:INTEGER;ORIENTA:ORIENTACION);
23     9:D     1    PROCEDURE SEL_VENTANA( X1,Y1,X2,Y2:REAL);
23    10:D     1    PROCEDURE SEL_PUERTO( X1,Y1,X2,Y2:REAL);
23    11:D     1    PROCEDURE SEL_TIPO( T:INTEGER);
23    12:D     1    PROCEDURE INI_GRAF( DISPO:DISPOSITIVO;B_ECO:BOOLEAN);
23    13:D     1    PROCEDURE LEE_TAB_CAR( AR_DEF_CAR:STRING);
23    14:D     1    PROCEDURE TERM_GRAF(BORRA:BOOLEAN);
23    15:D     1    PROCEDURE PROC_GRAF;
23    16:D     3    FUNCTION DAT_GRAF(PREG:INTEGER):REAL;
29     1:D     4
29     2:D     3   FUNCTION SIN(X:REAL):REAL;
29     3:D     3   FUNCTION COS(X:REAL):REAL;
29     4:D     3   FUNCTION EXP(X:REAL):REAL;
29     5:D     3   FUNCTION ATAN(X:REAL):REAL;
29     6:D     3   FUNCTION LN(X:REAL):REAL;
29     7:D     3   FUNCTION LOG(X:REAL):REAL;
29     8:D     3   FUNCTION SQRT(X:REAL):REAL;
29     8:D     5
 1     1:D     5 USES GRAFICAS,TRANSCENDS;
 1     1:D     3
 1     1:D     3 CONST
 1     1:D     3    MAXPT=200;            (* numero de puntos del objeto       *)
```

```
 59  1   1:0    3 (*$P+*)
 60  1   1:0    3 TYPE
 61  1   1:0    3    MATRIZ=ARRAY [1..4,1..4] OF REAL;    (* Matriz homogenea    *)
 62  1   1:0    3    PTJD=RECORD
 63  1   1:0    3       X,Y,Z,W:REAL;
 64  1   1:0    3       END;
 65  1   1:0    3
 66  1   1:0    3 VAR
 67  1   1:0    3    EXFILE:FILE OF REAL;
 68  1   1:0  305    N,I:INTEGER;
 69  1   1:0  307    CH:CHAR;
 70  1   1:0  308    X,Y:ARRAY [1..MAXPT] OF REAL;
 71  1   1:0 1108    RAD,A,B,C,K:REAL;
 72  1   1:0 1118    TRANSFORMACION,MATIDENT,
 73  1   1:0 1118    MATWN,MATON,MATTRAN:MATRIZ;
 74  1   1:0 1278    NOMBRE:STRING;
 75  1   1:0 1319
 76  1   1:0 1319
 77  1   2:0    1 PROCEDURE MULTIPLICA(VAR Y,X:PTJD;VAR T:MATRIZ);
 78  1   2:0    4
 79  1   2:0    4 (*************************************************************************)
 80  1   2:0    4 (*                                                                       *)
 81  1   2:0    4 (*      Esta rutina multiplica un punto con una matriz de 4x4            *)
 82  1   2:0    4 (* para generar un nuevo punto Y=T*X.                                    *)
 83  1   2:0    4 (*                                                                       *)
 84  1   2:0    4 (*************************************************************************)
 85  1   2:0    4
 86  1   2:0    4 VAR
 87  1   2:0    4    I,J:INTEGER;
 88  1   2:0    6    A,B:ARRAY[1..4] OF REAL;
 89  1   2:0    0 BEGIN (*MULTIPLICA*)
 90  1   2:1    0    B[1]:=X.X;          B[2]:=X.Y;
 91  1   2:1   34    B[3]:=X.Z;          B[4]:=X.W;
 92  1   2:1   66    FOR I:=1 TO 4 DO
 93  1   2:2   78    BEGIN
 94  1   2:3   78       A[I]:=0.0;
 95  1   2:3   96       FOR J:=1 TO 4 DO
 96  1   2:4  108          A[I]:=A[I]+T[I,J]*B[J]
 97  1   2:2  159    END;(*enfor*)
 98  1   2:1  179    Y.X:=A[1];          Y.Y:=A[2];
 99  1   2:1  213    Y.Z:=A[3];          Y.W:=A[4];
100  1   2:0  245 END; (*MULTIPLICA*)
```

```
1    2:0   262 (*$P*)
1    3:D     1 PROCEDURE CONC(A,B:MATRIZ; VAR C:MATRIZ);
1    3:D    68
1    3:D    68 (*******************************************************************)
1    3:D    68 (*                                                                 *)
1    3:D    68 (*      Esta rutina multiplica un par de matrices de 4x4           *)
1    3:D    68 (*  en el orden A*B=C                                              *)
1    3:D    68 (*                                                                 *)
1    3:D    68 (*******************************************************************)
1    3:D    68
1    3:D    68 VAR
1    3:D    68    I,J,K:INTEGER;
1    3:0     0 BEGIN (*CONC*)
1    3:1     0    FOR I:=1 TO 4 DO
1    3:2    23       FOR J:=1 TO 4 DO
1    3:3    36       BEGIN
1    3:4    36          C[I,J]:=0;
1    3:4    59          FOR K:=1 TO 4 DO
1    3:5    72             C[I,J]:=C[I,J]+A[I,K]*B[K,J]
1    3:3   154       END
1    3:0   168 END; (*CONC*)
1    3:0   206
1    4:D     1 PROCEDURE DEF_PUNTO(VAR P:PT3D; X,Y,Z,W:REAL);
1    4:D    10
1    4:D    10 (*******************************************************************)
1    4:D    10 (*                                                                 *)
1    4:D    10 (*      Esta rutina carga datos a un registro de tipo punto        *)
1    4:D    10 (*  de tres dimensiones homogeneas PT3D.                           *)
1    4:D    10 (*                                                                 *)
1    4:D    10 (*******************************************************************)
1    4:D    10
1    4:0     0 BEGIN (*DEF_PUNTO*)
1    4:1     0    P.X:=X;
1    4:1     9    P.Y:=Y;
1    4:1    18    P.Z:=Z;
1    4:1    27    P.W:=W
1    4:0    28 END; (*DEF_PUNTO*)
1    4:0    46
1    5:D     1 PROCEDURE POSICION(VAR M:MATRIZ; A,B,C:REAL);
1    5:D     8
1    5:D     8 (*******************************************************************)
1    5:D     8 (*                                                                 *)
1    5:D     8 (*      Esta rutina sirve para definir una matriz homogenea        *)
1    5:D     8 (*  para hacer transformacion de traslacion a X=A, Y=B, Z=C        *)
1    5:D     8 (*                                                                 *)
1    5:D     8 (*******************************************************************)
1    5:D     8
1    5:0     0 BEGIN (*POSICION*)
1    5:1     0    M:=MATIDENT;
1    5:1     6    M[1,4]:=A;
1    5:1    29    M[2,4]:=B;
1    5:1    52    M[3,4]:=C;
1    5:0    75 END; (*POSICION*)
```

```
154  1    5:0    88 (*$P*)
155  1    6:0     1 PROCEDURE TRANS3(P:PT3D;VAR X,Y:REAL);
156  1    6:0    12
157  1    6:0    12 (*************************************************************************)
158  1    6:0    12 (*                                                                       *)
159  1    6:0    12 (*      Esta rutina transforaa de tres dimensiones homogeneas            *)
160  1    6:0    12 (*  a dos dimensiones.                                                   *)
161  1    6:0    12 (*                                                                       *)
162  1    6:0    12 (*************************************************************************)
163  1    6:0    12
164  1    6:0     0 BEGIN (*TRANS3*)
165  1    6:1     0    MULTIPLICA(P,P,MATTRAN);
166  1    6:1    14    IF P.W=0 THEN
167  1    6:2    24       WRITE(CHR(7))
168  1    6:1    34    ELSE
169  1    6:2    36    BEGIN
170  1    6:3    36       X:=P.X/P.W;
171  1    6:3    48       Y:=P.Y/P.W
172  1    6:2    55    END
173  1    6:0    60 END; (*TRANS3*)
174  1    6:0    72
175  1    7:0     1 PROCEDURE INVMAT1(T:MATRIZ; VAR IT:MATRIZ);
176  1    7:0    33
177  1    7:0    33 (*************************************************************************)
178  1    7:0    33 (*                                                                       *)
179  1    7:0    33 (*      Esta rutina efectua la inversion de una matriz de 4x4            *)
180  1    7:0    33 (*  con las caracteristicas de la matriz de rotacion.                    *)
181  1    7:0    33 (*                                                                       *)
182  1    7:0    33 (*************************************************************************)
183  1    7:0    33
184  1    7:0    33 VAR
185  1    7:0    33    I,J:INTEGER;
186  1    7:0    37
187  1    7:0     0 BEGIN (*INVMAT1*)
188  1    7:1     0    FOR I:=1 TO 3 DO
189  1    7:2    18    BEGIN
190  1    7:3    18       IT[I,4]:=0;
191  1    7:3    40       IT[4,I]:=0;
192  1    7:3    62       FOR J:=1 TO 3 DO
193  1    7:4    75       BEGIN
194  1    7:5    75          IT[I,J]:=T[J,I];
195  1    7:5   118          IT[I,4]:=IT[I,4]-T[J,I]*T[J,4]
196  1    7:4   197       END(*endfor*)
197  1    7:2   203    END;(*endfor*)
198  1    7:1   219    IT[4,4]:=1;
199  1    7:0   240 END; (*INVMAT1*)
200  1    7:0   260
```

```
1    7:0  260 (*$P*)
1    8:D    1 PROCEDURE ROTACION(VAR M:MATRIZ; KX,KY,KZ,ANG:REAL);
1    8:D   10
1    8:D   10 (*****************************************************************)
1    8:D   10 (*                                                               *)
1    8:D   10 (*       Esta rutina genera una matriz de transformacion para    *)
1    8:D   10 (*   rotar un angulo ANG alrededor del vector KX,KY,KZ.          *)
1    8:D   10 (*       ANG esta en grados sexagesimales.                       *)
1    8:D   10 (*                                                               *)
1    8:D   10 (*****************************************************************)
1    8:D   10
1    8:D   10 VAR
1    8:D   10    I:INTEGER;
1    8:D   11    C,S,V,L:REAL;
1    8:D   19
1    8:0    0 BEGIN (*ROTACION*)
1    8:1    0    L:=SQRT(KX*KX+KY*KY+KZ*KZ);  (* Modulo del vector    *)
1    8:1   38    C:=COS(ANG*RAD);             (* coseno de ANG        *)
1    8:1   57    S:=-SIN(ANG*RAD);            (* seno de ANG          *)
1    8:1   77    V:=1.0-C;
1    8:1   93    KX:=KX/L;                    (* Normalizacion        *)
1    8:1  106    KY:=KY/L;
1    8:1  119    KZ:=KZ/L;
1    8:1  132    M:=MATIDENT;
1    8:1  138    M[1,1]:=KX*KX*V+   C;
1    8:1  176    M[1,2]:=KX*KY*V+KZ*S;
1    8:1  219    M[1,3]:=KX*KZ*V-KY*S;
1    8:1  262    M[2,1]:=KY*KX*V-KZ*S;
1    8:1  305    M[2,2]:=KY*KY*V+   C;
1    8:1  343    M[2,3]:=KY*KZ*V+KX*S;
1    8:1  386    M[3,1]:=KZ*KX*V+KY*S;
1    8:1  429    M[3,2]:=KZ*KY*V-KX*S;
1    8:1  472    M[3,3]:=KZ*KZ*V+   C;
1    8:0  510 END; (*ROTACION*)
1    8:0  522
1    8:0  522
1    9:D    1 PROCEDURE INIT3DGRAPH;
1    9:D    1
1    9:D    1 (*****************************************************************)
1    9:D    1 (*                                                               *)
1    9:D    1 (*       Esta rutina inicia el ambiente de graficas en tres      *)
1    9:D    1 (*  dimensiones.                                                 *)
1    9:D    1 (*                                                               *)
1    9:D    1 (*****************************************************************)
1    9:D    1
1    9:D    1 VAR
1    9:D    1    I,J:INTEGER;
1    9:0    0 BEGIN (*INIT3DGRAPH*)
1    9:1    0    SEL_VENTANA(-1,-1,1,1);
1    9:1   13    FOR I:=1 TO 4 DO
1    9:2   24       FOR J:=1 TO 4 DO
1    9:3   35          MATIDENT[I,J]:=0;
1    9:1   72    FOR I:=1 TO 4 DO
1    9:2   83       MATIDENT[I,I]:=1;
1    9:1  113    MATON:=MATIDENT;
1    9:1  121    MATWN:=MATIDENT;
1    9:1  129    NOMBRE:='MCA7:GRAF';
1    9:0  146 END; (*INIT3DGRAPH*)
```

```
259  1    9:0   164 (*$P*)
260  1   10:D     1 PROCEDURE COMUNICACIONES;
261  1   10:D     1
262  1   10:D     1 (*************************************************************************)
263  1   10:D     1 (*                                                                       *)
264  1   10:D     1 (*      Esta rutina establece las comunicaciones con el usuario        *)
265  1   10:D     1 (* para definir el objeto, la salida y la transformacion.             *)
266  1   10:D     1 (*                                                                       *)
267  1   10:D     1 (*************************************************************************)
268  1   10:D     1
268  1   10:D     1 (*$I GR3010*)
269  1   10:D     1
270  1   10:D     1 (*************************************************************************)
271  1   10:D     1 (*                                                                       *)
272  1   10:D     1 (*      Este es el archivo GR310 que llama GR300 al momento            *)
273  1   10:D     1 (*  de su compilacion.                                                   *)
274  1   10:D     1 (*                                                                       *)
275  1   10:D     1 (*                                          Miguel Chin Auyon          *)
276  1   10:D     1 (*                                                                       *)
277  1   10:D     1 (*************************************************************************)
278  1   10:D     1
279  1   10:D     1
280  1   11:D     1 PROCEDURE DEF_MATTRAN(VAR M:MATRIZ; TIPO:INTEGER);
281  1   11:D     3
282  1   11:D     3 (*************************************************************************)
283  1   11:D     3 (*                                                                       *)
284  1   11:D     3 (*      Esta rutina hace la definicion de las matrices de              *)
285  1   11:D     3 (*  transformacion para el objeto y para el usuario.                   *)
286  1   11:D     3 (*                                                                       *)
287  1   11:D     3 (*************************************************************************)
288  1   11:D     3
289  1   11:D     3 VAR
290  1   11:D     3    MAT:MATRIZ;
291  1   11:D    35    CH,I:CHAR;
292  1   11:D    37    V:ARRAY ['X'..'Z'] OF REAL;
293  1   11:D    43
294  1   12:D     1 PROCEDURE DEF_POS(TIPO:INTEGER; VAR M:MATRIZ);
295  1   12:D     3
296  1   12:D     3 (*************************************************************************)
297  1   12:D     3 (*                                                                       *)
298  1   12:D     3 (*      Esta rutina hace la definicion de la matriz de                 *)
299  1   12:D     3 (*  traslacion en coordenadas homogeneas.                              *)
300  1   12:D     3 (*                                                                       *)
301  1   12:D     3 (*************************************************************************)
302  1   12:D     3
303  1   12:0     0 BEGIN (*DEF_POS*)
304  1   12:1     0    CASE TIPO OF
305  1   12:1     3    0:WRITELN('POSICION DEL ORIGEN RELATIVO DEL OBJETO');
306  1   12:1    64    1:WRITELN('POSICION DEL OBSERVADOR EN EL MUNDO');
307  1   12:1   121    END;
308  1   12:1   132    FOR I:='X' TO 'Z' DO
309  1   12:2   146    BEGIN
310  1   12:3   146       WRITE(I,' := ');
311  1   12:3   174       READLN(V[I])
312  1   12:2   203    END;
313  1   12:1   213    CASE TIPO OF
314  1   12:1   216    0:POSICION(M,V['X'],V['Y'],V['Z']);
315  1   12:1   260    1:POSICION(M,-V['X'],-V['Y'],-V['Z'])
316  1   12:1   303    END
317  1   12:0   318 END; (*DEF_POS*)
```

```
318  1   12:0   332 (*$P*)
319  1   13:D     1 PROCEDURE DEF_ROT(TIPO:INTEGER; VAR M:MATRIZ);
320  1   13:D     3
321  1   13:D     3 (*************************************************************************)
322  1   13:D     3 (*                                                                       *)
323  1   13:D     3 (*      Esta rutina define una matriz homogenea de rotacion              *)
324  1   13:D     3 (*  sobre cualquier eje, en grados sexagesimales.                        *)
325  1   13:D     3 (*                                                                       *)
326  1   13:D     3 (*************************************************************************)
327  1   13:D     3
328  1   13:D     3 VAR
329  1   13:D     3    CH:CHAR;
330  1   13:D     4    MAT:MATRIZ;
331  1   13:D    36    ANG:REAL;
332  1   13:D    38
333  1   13:0     0 BEGIN (*DEF_ROT*)
334  1   13:1     0    M:=MATIDENT;
335  1   13:1     6    REPEAT
336  1   13:2     6       MAT:=MATIDENT;
337  1   13:2    13       WRITELN('ROTACIONES: X Y Z V)ECT S)ALIDA');
338  1   13:2    64       READ(KEYBOARD,CH);
339  1   13:2    74       CASE CH OF
340  1   13:2    77       'V':BEGIN
341  1   13:4    77          WRITE('ANGULO DE GIRO := ');
342  1   13:4   107          READLN(ANG);
343  1   13:4   125          WRITELN('COMPONENTES DEL VECTOR EJE');
344  1   13:4   171          FOR CH:='X' TO 'Z' DO
345  1   13:5   183          BEGIN
346  1   13:6   183             WRITE(CH,' := ');
347  1   13:6   209             READLN(V[CH])
348  1   13:5   236          END; (*endfor*)
349  1   13:4   243          ROTACION(MAT,V['X'],V['Y'],V['Z'],ANG)
350  1   13:3   288       END;
351  1   13:2   292       'X':BEGIN
352  1   13:4   292          WRITE('ROTACION EN X := ');
353  1   13:4   321          READLN(ANG);
354  1   13:4   339          ROTACION(MAT,1,0,0,ANG)
355  1   13:3   351       END;
356  1   13:2   355       'Y':BEGIN
357  1   13:4   355          WRITE('ROTACION EN Y := ');
358  1   13:4   384          READLN(ANG);
359  1   13:4   402          ROTACION(MAT,0,1,0,ANG)
360  1   13:3   414       END;
361  1   13:2   418       'Z':BEGIN
362  1   13:4   418          WRITE('ROTACION EN Z := ');
363  1   13:4   447          READLN(ANG);
364  1   13:4   465          ROTACION(MAT,0,0,1,ANG)
365  1   13:3   477       END
366  1   13:2   479       END; (*endcase*)
367  1   13:2   498       IF NOT(CH IN ['V','S','X','Y','Z']) THEN
368  1   13:3   519          WRITE(CHR(7));
369  1   13:2   529       CASE TIPO OF
370  1   13:2   532       0:CONC(MAT,M,M);
371  1   13:2   540       1:CONC(M,MAT,M)
372  1   13:2   544       END;
373  1   13:1   560    UNTIL CH='S';
374  1   13:1   565    IF TIPO=1 THEN INVMAT1(M,M)
375  1   13:0   572 END; (*DEF_ROT*)
```

```
376  1  13:0  394 (*$P*)
377  1  14:0    1 PROCEDURE DEF_ESC(VAR M:MATRIZ);
378  1  14:0    2
379  1  14:0    2 (*************************************************************************)
380  1  14:0    2 (*                                                                       *)
381  1  14:0    2 (*      Esta rutina crea una matriz homogenea de escalamiento            *)
382  1  14:0    2 (*                                                                       *)
383  1  14:0    2 (*************************************************************************)
384  1  14:0    2
385  1  14:0    0 BEGIN (*DEF_ESC*)
386  1  14:1    0    WRITELN('ESCALAMIENTO');
387  1  14:1   32    FOR I:='X' TO 'Z' DO
388  1  14:2   46    BEGIN
389  1  14:3   46       WRITE(I,' := ');
390  1  14:3   74       READLN(V[I])
391  1  14:2  103    END;(*endfor*)
392  1  14:1  113    M:=MATIDENT;
393  1  14:1  119    IF V['X']=V['Y']=V['Z']=0 THEN
394  1  14:2  166       WRITE(CHR(7))
395  1  14:1  176    ELSE
396  1  14:2  178    BEGIN
397  1  14:3  178       M[1,1]:=V['X'];
398  1  14:3  210       M[2,2]:=V['Y'];
399  1  14:3  242       M[3,3]:=V['Z']
400  1  14:2  270    END(*endif*)
401  1  14:0  274 END; (*DEF_ESC*)
402  1  14:0  288
403  1  14:0  288
404  1  15:0    1 PROCEDURE DEF_MAT(VAR M:MATRIZ);
405  1  15:0    2
406  1  15:0    2 (*************************************************************************)
407  1  15:0    2 (*                                                                       *)
408  1  15:0    2 (*      Esta rutina permite definir una matriz de 4x4                    *)
409  1  15:0    2 (*                                                                       *)
410  1  15:0    2 (*************************************************************************)
411  1  15:0    2
412  1  15:0    2 VAR
413  1  15:0    2    I,J:INTEGER;
414  1  15:0    4
415  1  15:0    0 BEGIN (*DEF_MAT*)
416  1  15:1    0    WRITELN('DEF_INICION DE TRANSFORMACION ESPECIAL');
417  1  15:1   58    WRITELN;
418  1  15:1   66    FOR I:=1 TO 4 DO
419  1  15:2   77       FOR J:=1 TO 4 DO
420  1  15:3   88       BEGIN
421  1  15:4   88          WRITE('TRANS[',I,',',J,'] := ');
422  1  15:4  153          READLN(M[I,J])
423  1  15:3  186       END(*endfor*)
424  1  15:0  186 END; (*DEF_MAT*)
```

```
425  1  15:0  216 (*$P*)
426  1  16:D    1 PROCEDURE DEF_PRT;
427  1  16:D    1
428  1  16:D    1 (*************************************************************************)
429  1  16:D    1 (*                                                                       *)
430  1  16:D    1 (*      Esta rutina permite alterar los margenes del puerto              *)
431  1  16:D    1 (*                                                                       *)
432  1  16:D    1 (*************************************************************************)
433  1  16:D    1
434  1  16:D    1 VAR
435  1  16:D    1   L,B,R,T:REAL;
436  1  16:D    9
437  1  16:0    0 BEGIN (*DEF_PRT*)
438  1  16:1    0   WRITELN('MARCO DEL PUERTO');
439  1  16:1   36   WRITELN;
440  1  16:1   44   WRITELN('TOPE SUPERIOR := ',DAT_GRAF(19));
441  1  16:1   97   WRITELN('TOPE INFERIOR := ',DAT_GRAF(17));
442  1  16:1  150   WRITELN('TOPE IZQUIERDO:= ',DAT_GRAF(16));
443  1  16:1  203   WRITELN('TOPE DERECHO  := ',DAT_GRAF(18));
444  1  16:1  256   WRITELN;
445  1  16:1  264   WRITE('TOPE SUP := ');       READLN(T);
446  1  16:1  306   WRITE('TOPE INF := ');       READLN(B);
447  1  16:1  348   WRITE('TOPE IZQ := ');       READLN(L);
448  1  16:1  390   WRITE('TOPE DER := ');       READLN(R);
449  1  16:1  432   SEL_PUERTO(L,B,R,T)
450  1  16:0  448 END; (*DEF_PRT*)
451  1  16:0  464
452  1  17:D    1 PROCEDURE DEF_VEN;
453  1  17:D    1
454  1  17:D    1 (*************************************************************************)
455  1  17:D    1 (*                                                                       *)
456  1  17:D    1 (*      Esta rutina permite alterar los margenes de la ventna            *)
457  1  17:D    1 (*                                                                       *)
458  1  17:D    1 (*************************************************************************)
459  1  17:D    1
460  1  17:D    1 VAR
461  1  17:D    1   L,B,R,T:REAL;
462  1  17:D    9
463  1  17:0    0 BEGIN (*DEF_VEN*)
464  1  17:1    0   WRITELN('MARCO DE LA VENTANA');
465  1  17:1   39   WRITELN;
466  1  17:1   47   WRITELN('TOPE SUPERIOR := ',DAT_GRAF(15));
467  1  17:1  100   WRITELN('TOPE INFERIOR := ',DAT_GRAF(13));
468  1  17:1  153   WRITELN('TOPE IZQUIERDO:= ',DAT_GRAF(12));
469  1  17:1  206   WRITELN('TOPE DERECHO  := ',DAT_GRAF(14));
470  1  17:1  259   WRITELN;
471  1  17:1  267   WRITE('TOPE SUP := ');       READLN(T);
472  1  17:1  309   WRITE('TOPE INF := ');       READLN(B);
473  1  17:1  351   WRITE('TOPE IZQ := ');       READLN(L);
474  1  17:1  393   WRITE('TOPE DER := ');       READLN(R);
475  1  17:1  435   SEL_VENTANA(L,B,R,T)
476  1  17:0  451 END; (*DEF_VEN*)
```

```
477  1   17:0   466 (*$P*)
478  1   18:D     1 PROCEDURE DEF_PER(VAR M:MATRIZ);
479  1   18:D     2
480  1   18:D     2 (*************************************************************************)
481  1   18:D     2 (*                                                                       *)
482  1   18:D     2 (*      Esta rutina define una matriz homogenea de perspectiva           *)
483  1   18:D     2 (*                                                                       *)
484  1   18:D     2 (*************************************************************************)
485  1   18:D     2
486  1   18:0     0 BEGIN (*DEF_PER*)
487  1   18:1     0    WRITELN('DEF_INICION DEL FOCO DE PERSPECTIVA');
488  1   18:1    55    WRITELN('       DAME LAS COMPONENTES');
489  1   18:1   102    FOR I:='X' TO 'Z' DO
490  1   18:2   116    BEGIN
491  1   18:3   116       WRITE(I,' := ');
492  1   18:3   144       READLN(V[I])
493  1   18:2   173    END;
494  1   18:1   183    M:=MATIDENT;
495  1   18:1   189    M[4,1]:=-V['X'];
496  1   18:1   222    M[4,2]:=-V['Y'];
497  1   18:1   255    M[4,3]:=-V['Z']
498  1   18:0   283 END; (*DEF_PER*)
499  1   18:0   302
500  1   11:0     0 BEGIN (*DEF_MAT*)
501  1   11:1     0    REPEAT
502  1   11:2     9       WRITE(CHR(7),CHR(12));
503  1   11:2    20       MAT:=MATIDENT;
504  1   11:2    27       CASE TIPO OF
505  1   11:2    30       0:BEGIN
506  1   11:4    30          WRITELN('TRANSFORMACION DEL OBJETO ');
507  1   11:4    76          WRITELN;
508  1   11:4    84          WRITELN('R)OT E)SC L)OC I)NI O)TRO S)ALIDA');
509  1   11:4   137          WRITELN('==== ==== ==== ==== ===== =======')
510  1   11:3   190       END;
511  1   11:2   192       1:BEGIN
512  1   11:4   192          WRITELN('POSICION Y ORIENTACION DEL OBSERVADOR');
513  1   11:4   249          WRITELN;
514  1   11:4   257          WRITELN('F)OCO R)OT L)OC I)NI V)ENT P)RTO S)ALIDA');
515  1   11:4   317          WRITELN('===== ==== ==== ==== ===== ===== =======')
516  1   11:3   377       END
517  1   11:2   377       END;(*endcase*)
518  1   11:2   390       READ(KEYBOARD,CH);
519  1   11:2   400       CASE CH OF
520  1   11:2   404       'E':IF TIPO=0 THEN DEF_ESC(MAT);
521  1   11:2   415       'I':M:=MATIDENT;
522  1   11:2   423       'L':DEF_POS(TIPO,MAT);
523  1   11:2   430       'O':IF TIPO=0 THEN DEF_MAT(MAT);
524  1   11:2   441       'F':IF TIPO=1 THEN DEF_PER(MAT);
525  1   11:2   452       'R':DEF_ROT(TIPO,MAT);
526  1   11:2   459       'V':IF TIPO=1 THEN DEF_VEN;
527  1   11:2   468       'P':IF TIPO=1 THEN DEF_PRT
528  1   11:2   473       END;(*endcase*)
529  1   11:2   520       IF CH IN ['F','R','E','O','L'] THEN
530  1   11:3   540          CONC(MAT,M,M)
531  1   11:2   544       ELSE
532  1   11:3   548          WRITE(CHR(7));
533  1   11:3   558       (*endif*)
534  1   11:1   558    UNTIL CH='S'
535  1   11:0   560 END; (*DEF_MAT*)
```

```
536  1  11:0  582 (*$P0)
537  1  11:0  582 (*$I GR3018)
537  1  11:0  582 (*$I GR3028)
538  1  11:0  582 (*****************************************************************************)
539  1  11:0  582 (*                                                                           *)
540  1  11:0  582 (*      Este es el archivo GR302 que es llamado por GR3 al                   *)
541  1  11:0  582 (*  ser compilado este.                                                      *)
542  1  11:0  582 (*                                                                           *)
543  1  11:0  582 (*                                          Miguel Chin Auyon                *)
544  1  11:0  582 (*                                                                           *)
545  1  11:0  582 (*****************************************************************************)
546  1  11:0  582
547  1  19:D    1 PROCEDURE DEF_OBJETO;
548  1  19:D    1
549  1  19:D    1 (*****************************************************************************)
550  1  19:D    1 (*                                                                           *)
551  1  19:D    1 (*      Esta rutina permite definir objetos en wireframe,                    *)
552  1  19:D    1 (*  salvando en archivos en disco, o recuperarlos.                           *)
553  1  19:D    1 (*                                                                           *)
554  1  19:D    1 (*****************************************************************************)
555  1  19:D    1
556  1  19:D    1 VAR
557  1  19:D    1    I ,J :INTEGER;
558  1  19:D    3    A ,N :REAL;
559  1  19:D    7    L:PACKED ARRAY[1..3] OF STRING;
560  1  19:D  130    CH ,K :CHAR;
561  1  19:D  132    NOM:STRING;
562  1  19:D  173    EXISTE:BOOLEAN;
563  1  19:D  174
564  1  20:D    1 PROCEDURE MANTENER;
565  1  20:D    1
566  1  20:D    1 (*****************************************************************************)
567  1  20:D    1 (*                                                                           *)
568  1  20:D    1 (*      Revisa si el archivo activo esta aun vigente                         *)
569  1  20:D    1 (*                                                                           *)
570  1  20:D    1 (*****************************************************************************)
571  1  20:D    1
572  1  20:0    0 BEGIN (*MANTENER*)
573  1  20:1    0    REPEAT
574  1  20:2    0       EXISTE:=TRUE;
575  1  20:2    5       (*$I-*)        (* Apaga verificacion de error de I/O  *)
576  1  20:2    5       RESET(EXFILE,NOMBRE);
577  1  20:2   15       IF IORESULT<>0 THEN
578  1  20:3   21       BEGIN
579  1  20:4   21          EXISTE:=FALSE;
580  1  20:4   26          WRITELN('El archivo ',NOMBRE,' no existe');
581  1  20:4   83          WRITELN('DAME EL NOMBRE DEL ARCHIVO DEL OBJETO');
582  1  20:4  136          READLN(NOMBRE)
583  1  20:3  152       END;
584  1  20:3  152       (*endif*)
585  1  20:2  152       CLOSE(EXFILE);
586  1  20:2  158       (*$I+*)        (* Prende verificacion de error de I/O  *)
587  1  20:1  158    UNTIL EXISTE
588  1  20:0  158 END; (*MANTENER*)
```

```
589  1  20:0  180 (*$P*)
590  1  21:0    1 PROCEDURE RECUPERAR;
591  1  21:0    1
592  1  21:0    1 (*************************************************************************)
593  1  21:0    1 (*                                                                       *)
594  1  21:0    1 (*      Recupera un archivo con definicion de objetos                    *)
595  1  21:0    1 (*                                                                       *)
596  1  21:0    1 (*************************************************************************)
597  1  21:0    1
598  1  21:0    0 BEGIN (*RECUPERAR*)
599  1  21:1    0    REPEAT
600  1  21:2    0       WRITELN('DAME EL NOMBRE DEL ARCHIVO DEL OBJETO');
601  1  21:2   57       READLN(NOMBRE);
602  1  21:2   77       EXISTE:=TRUE;
603  1  21:2   82       (*$I-*)        (* Apaga verificacion de error de I/O   *)
604  1  21:2   82       RESET(EXFILE,NOMBRE);
605  1  21:2   92       IF IORESULT<>0 THEN
606  1  21:3   98       BEGIN
607  1  21:4   98          EXISTE:=FALSE;
608  1  21:4  103          WRITELN('El archivo ',NOMBRE,' no existe')
609  1  21:3  160       END;
610  1  21:3  160       (*endif*)
611  1  21:2  160       CLOSE(EXFILE);
612  1  21:2  166       (*$I+*)        (* Prende verificacion de error de I/O  *)
613  1  21:1  166    UNTIL EXISTE
614  1  21:0  166 END; (*RECUPERAR*)
615  1  21:0  186
616  1  22:0    1 PROCEDURE ESC_FILE(VAR DATO:REAL);
617  1  22:0    2
618  1  22:0    2 (*************************************************************************)
619  1  22:0    2 (*                                                                       *)
620  1  22:0    2 (*      Esta rutina lee de consola y escribe el dato leido al            *)
621  1  22:0    2 (* archivo EXFILE.                                                       *)
622  1  22:0    2 (*                                                                       *)
623  1  22:0    2 (*************************************************************************)
624  1  22:0    2
625  1  22:0    0 BEGIN (*ESC_FILE*)
626  1  22:1    0    READLN(DATO);
627  1  22:1   17    EXFILE^:=DATO;
628  1  22:1   23    PUT(EXFILE);
629  1  22:0   30 END; (*ESC_FILE*)
```

```
1   22:0    42 (*$P*)
1   23:D     1 PROCEDURE SALVAR;
1   23:D     1
1   23:D     1 (*****************************************************************)
1   23:D     1 (*                                                               *)
1   23:D     1 (*       Permite salvar un archivo con la definicion de algun    *)
1   23:D     1 (*  objeto en wireframe.                                         *)
1   23:D     1 (*                                                               *)
1   23:D     1 (*****************************************************************)
1   23:D     1
1   23:0     0 BEGIN (*SALVAR*)
1   23:1     0    WRITELN('DAME EL NOMBRE DEL ARCHIVO  DEL OBJETO');
1   23:1    58    READLN(NOMBRE);
1   23:1    78    REWRITE(EXFILE,NOMBRE);
1   23:1    90    WRITELN('DAME EL NUMERO DE PUNTOS');
1   23:1   134    ESCFILE(N);
1   23:1   139    FOR I:=1 TO TRUNC(N) DO
1   23:2   159    BEGIN
1   23:3   159       WRITELN('PUNTO 3D(',I,')');
1   23:3   210       FOR K:='X' TO 'Z' DO
1   23:4   226       BEGIN
1   23:5   226          WRITE(K,' := ');
1   23:5   255          ESCFILE(A)
1   23:4   258       END(*endfor*)
1   23:2   260    END;(*endfor*)
1   23:1   282    WRITELN('DAME EL NUMERO DE LINEAS');
1   23:1   326    ESCFILE(N);
1   23:1   331    FOR I:=1 TO TRUNC(N) DO
1   23:2   351    BEGIN
1   23:3   351       WRITELN('LINEA(',I,')');
1   23:3   399       FOR J:=1 TO 3 DO
1   23:4   413       BEGIN
1   23:5   413          WRITE(L[J],' := ');
1   23:5   451          ESCFILE(A)
1   23:4   454       END(*endfor*)
1   23:2   456    END;(*endfor*)
1   23:1   476    CLOSE(EXFILE,LOCK)
1   23:0   484 END; (*SALVAR*)
1   23:0   504
1   19:0     0 BEGIN (*DEF_OBJETO*)
1   19:1     0    L[1]:='PUNTO 1';
1   19:1    22    L[2]:='PUNTO 2';
1   19:1    44    L[3]:=' TIPO DE LINEA';
1   19:1    73    WRITELN(CHR(12),CHR(7),' S)ALVAR  R)ECUPERAR M)ANTENER');
1   19:1   143    REPEAT
1   19:2   143       READ(KEYBOARD,CH);
1   19:2   154       IF NOT(CH IN ['R','S','M']) THEN
1   19:3   177          WRITE(CHR(7));
1   19:1   187    UNTIL CH IN ['R','S','M'];
1   19:1   208    CASE CH OF
1   19:1   213    'M': MANTENER;
1   19:1   217    'R': RECUPERAR;
1   19:1   221    'S': SALVAR;
1   19:1   225    END(*endcase*)
1   19:0   246 END; (*DEF_OBJETO*)
```

```
 1   19:0   260 (*$P*)
 1   24:D     1 PROCEDURE DEF_SALIDA;
 1   24:D     1
 1   24:D     1 (*****************************************************************)
 1   24:D     1 (*                                                               *)
 1   24:D     1 (*      Esta rutina sirve para alterar las caracteristicas del  *)
 1   24:D     1 (*  dispositivo de salida seleccionado.                         *)
 1   24:D     1 (*                                                               *)
 1   24:D     1 (*****************************************************************)
 1   24:D     1
 1   24:D     1 VAR
 1   24:D     1    CH:CHAR;
 1   24:0     0 BEGIN (*DEF_SALIDA*)
 1   24:1     0    WRITE(CHR(7),CHR(12));
 1   24:1    20    WRITELN('A > OSCILOSCOPIO SIN ECO');
 1   24:1    64    WRITELN('B > OSCILOSCOPIO CON ECO');
 1   24:1   108    WRITELN('C > PLOTTER SIN ECO');
 1   24:1   147    WRITELN('D > PLOTTER CON ECO');
 1   24:1   186    WRITELN('E > MANTENER');
 1   24:1   218    REPEAT
 1   24:2   218       READ(KEYBOARD,CH);
 1   24:2   228       IF CH IN ['B','D'] THEN
 1   24:3   246       BEGIN
 1   24:4   246          WRITELN('DAME EL NOMBRE DEL ARCHIVO DE ECO');
 1   24:4   299          READLN(NOM_AR_ECO);
 1   24:4   319          TERM_GRAF(FALSE)
 1   24:3   320       END;
 1   24:2   323       CASE CH OF
 1   24:2   326       'A': INI_GRAF(OSC,FALSE);
 1   24:2   333       'B': INI_GRAF(OSC,TRUE);
 1   24:2   340       'C': INI_GRAF(PLT,FALSE);
 1   24:2   347       'D': INI_GRAF(PLT,TRUE)
 1   24:2   349       END;
 1   24:1   370    UNTIL CH IN ['A'..'E'];
 1   24:1   388    PLM_ABJ;
 1   24:0   391 END; (*DEF_SALIDA*)
```

```
1   24:0  406 (**P*)
1   25:D    1 PROCEDURE MAPEA;
1   25:D    1
1   25:D    1 (****************************************************************)
1   25:D    1 (*                                                              *)
1   25:D    1 (*      Esta rutina efectua el mapeo de los puntos en tres      *)
1   25:D    1 (* dimensiones a la ventana en dos dimensiones.                 *)
1   25:D    1 (*      Se efectua una proyeccion paralela al plano de vision.  *)
1   25:D    1 (*                                                              *)
1   25:D    1 (****************************************************************)
1   25:D    1
1   25:D    1 VAR
1   25:D    1    P1,P2,NUMP,NUML,I:INTEGER;
1   25:D    6    P:PT3D;
1   25:D   14
1   26:D    3 FUNCTION LEE_ARCH:REAL;
1   26:D    3
1   26:D    3 (****************************************************************)
1   26:D    3 (*                                                              *)
1   26:D    3 (*      Esta funcion reemplaza a una lectura a EXFILE.          *)
1   26:D    3 (*                                                              *)
1   26:D    3 (****************************************************************)
1   26:D    3
1   26:0    0 BEGIN (*LEE_ARCH*)
1   26:1    0    GET(EXFILE);
1   26:1    7    LEE_ARCH:=EXFILE^
1   26:0   10 END; (*LEE_ARCH*)
1   26:0   26
1   25:0    0 BEGIN (*MAPEA*)
1   25:1    0    RESET(EXFILE,NOMBRE);
1   25:1   12    NUMP:=TRUNC(EXFILE^);
1   25:1   19    FOR I:=1 TO NUMP DO
1   25:2   30    BEGIN
1   25:3   30       DEF_PUNTO(P,LEE_ARCH,LEE_ARCH,LEE_ARCH,1);
1   25:3   48       TRANS3(P,X[I],Y[I])
1   25:2   76    END;
1   25:1   85    NUML:=TRUNC(LEE_ARCH);
1   25:1   93    FOR I:=1 TO NUML DO
1   25:2  104    BEGIN
1   25:3  104       P1:=TRUNC(LEE_ARCH);
1   25:3  112       P2:=TRUNC(LEE_ARCH);
1   25:3  120       SEL_TIPO(TRUNC(LEE_ARCH));
1   25:3  129       LINEA(X[P1],Y[P1],X[P2],Y[P2])
1   25:2  189    END;(*endfor*)
1   25:1  199    CLOSE(EXFILE);
1   25:0  207 END; (*MAPEA*)
1   25:0  224
1   25:0  224 (*$I GR302*)
```

```
769  1   25:0   224 (*$P*)
770  1   10:0     0 BEGIN (*COMUNICACIONES*)
771  1   10:1     0    DEF_OBJETO;
772  1   10:1     2    DEF_SALIDA;
773  1   10:1     4    DEF_MATTRAN(MATOW,0);
774  1   10:1    10    DEF_MATTRAN(MATWM,1);
775  1   10:1    16    CONC(MATWM,MATOW,MATTRAN);
776  1   10:1    27    WRITELN;
777  1   10:1    35    WRITELN('Calculando y Dibujando');
778  1   10:1    77    MAPEA
779  1   10:0    77 END; (*COMUNICACIONES*)
780  1   10:0    92
781  1   10:0    92
782  1   10:0    92 (**************************************************************************)
783  1   10:0    92 (*                                                                        *)
784  1   10:0    92 (*      Este es el programa principal de GR3                              *)
785  1   10:0    92 (*                                                                        *)
786  1   10:0    92 (**************************************************************************)
787  1   10:0    92
788  1    1:0     0 BEGIN
789  1    1:1     0    RAD:=ATAN(1)/45;     (* Factor de conversion a radianes      *)
790  1    1:1    57    PLN_ABJ;
791  1    1:1    60    INIT3D6RAPH;
792  1    1:1    62    REPEAT
793  1    1:2    62       COMUNICACIONES;
794  1    1:2    64       WRITELN;
795  1    1:2    72       WRITELN('B)orra C)ontinua S)alida');
796  1    1:2   116       REPEAT READ(KEYBOARD,CH);
797  1    1:2   127       UNTIL CH IN ['B','C','S'];
798  1    1:2   148       IF CH='B' THEN INIT3D6RAPH;
799  1    1:1   157    UNTIL CH='S';
800  1    1:0   164 END.
```

```
1   1:D     1 (*$L MCA3:PAS*)
1   1:D     1 Program gen_tab_char;
1   1:D     3
1   1:D     3 (***************************************************************)
1   1:D     3 (*                                                             *)
1   1:D     3 (*        Este programa tiene por objetivo generar las tablas con *)
1   1:D     3 (* los vectores y los indices de acceso para la generacion de  *)
1   1:D     3 (* caracteres.                                                 *)
1   1:D     3 (*        El archivo de entrada es TABCAR.TEXT que es creado en *)
1   1:D     3 (* el editor del sistema.                                      *)
1   1:D     3 (*                                                             *)
1   1:D     3 (***************************************************************)
1   1:D     3
1   1:D     3 Type
1   1:D     3    byte= 0..255;
1   1:D     3
1   1:D     3 Var
1   1:D     3    PCAR,
1   1:D     3    UCAR,
1   1:D     3    i,
1   1:D     3    ind: Integer;
1   1:D     7    dx,
1   1:D     7    dy: 0..7;
1   1:D     9    eoc,
1   1:D     9    plm :0..1;
1   1:D    11    mv_car:byte;
1   1:D    12    ind_car: PACKED array [0..255] of Integer;
1   1:D   268    tab_car: PACKED array [0..1023] of byte;
1   1:D   780    outfile: File;
1   1:D   820    infile: Interactive;
1   1:D  1121    car:STRING;
1   1:D  1162
1   2:D     1 PROCEDURE INICIA;
1   2:D     1
1   2:D     1 (***************************************************************)
1   2:D     1 (*                                                             *)
1   2:D     1 (*        Esta rutina abre para lectura el achivo TAB_CAR con  *)
1   2:D     1 (*  la definicion de los vectores de los caracteres.           *)
1   2:D     1 (*        Lee el ASCII del primer caracter definido y del ultimo *)
1   2:D     1 (*                                                             *)
1   2:D     1 (***************************************************************)
1   2:D     1
1   2:D     1 VAR
1   2:D     1   I: INTEGER;
1   2:D     2
1   2:0     0 BEGIN (*INICIA*)
1   2:1     0    FOR I:=0 TO 255 DO
1   2:2    13       INDCAR[I]:=0;
1   2:2    32    (*endfor*)
1   2:1    32    FOR I:=0 TO 1023 DO
1   2:2    45       TABCAR[I]:=0;
1   2:2    68    (*endfor*)
1   2:1    68    Reset(infile,'TABCAR.TEXT');
1   2:1    92    ind:=0;                    (*  Indice de vectores a 0      *)
1   2:1    95    READLN(INFILE,PCAR);
1   2:1   113    READLN(INFILE,UCAR);
1   2:1   131    WRITELN('DEFINICION DESDE ASCII ',PCAR)
1   2:0   184 END; (*INICIA*)
```

```
59  1   2:0   200 (*$P*)
60  1   3:D     1 PROCEDURE LEE_DEF;
61  1   3:D     1
62  1   3:D     1 (*************************************************************************)
63  1   3:D     1 (*                                                                       *)
64  1   3:D     1 (*      Esta  rutina lee cada uno de los vectores dentro del             *)
65  1   3:D     1 (*  arreglo TAB_CAR apuntado por IND_CAR.  Estos arreglos                *)
66  1   3:D     1 (*  seran leidos por GRAFICAS.                                           *)
67  1   3:D     1 (*      El formato del archivo TABCAR es:                                *)
68  1   3:D     1 (*      Primer y Ultimo caracter definidos (enteros)                     *)
69  1   3:D     1 (*      Control del caracter definido.                                   *)
70  1   3:D     1 (*        Por cada vector:                                               *)
71  1   3:D     1 (*           Sig. punto [X,Y] ( ambos menores a 8),                      *)
72  1   3:D     1 (*           pluma(1=arr,0=abj), eoc(ultimo vect=1)                      *)
73  1   3:D     1 (*      El bloque se repite por cada caracter hasta que desde            *)
74  1   3:D     1 (*  el primero al ultimo queden definidos.                               *)
75  1   3:D     1 (*                                                                       *)
76  1   3:D     1 (*************************************************************************)
77  1   3:D     1
78  1   3:D     1 VAR
79  1   3:D     1    I,
80  1   3:D     1    J: INTEGER;
81  1   3:D     3
82  1   3:0     0 BEGIN (*LEE_DEF*)
83  1   3:1     0    FOR i:=PCAR TO UCAR DO
84  1   3:2    11    BEGIN
85  1   3:3    11       ind_car[i]:=ind;          (* Apunta al vector inicial   *)
86  1   3:3    23       Readln(infile,car);       (* Lee el caracter a definir  *)
87  1   3:3    43       Write(car,I:10);          (* para efectos de control    *)
88  1   3:3    63       WRITELN(IND:10);
89  1   3:3    83       Repeat            (* Por cada caracter a definir         *)
90  1   3:4    83          Readln(infile,dx,dy,pla,eoc); (* Lee def del vector  *)
91  1   3:4   131          av_car:=((eoc*2+pla)*8+dy)*8+dX;       (* Codifica   *)
92  1   3:4   151          tab_car[ind]:=av_car;          (* Almacena en TAB_CAR *)
93  1   3:4   167          ind:=ind+1;                    (* Inc. indice de vect *)
94  1   3:3   172       Until (eoc=1);    (* Hasta el ultimo vect del caracter   *)
95  1   3:2   177    END;(*endfor*)
96  1   3:2   184
97  1   3:1   184    Writeln;
98  1   3:1   192    Writeln(' Ya lei todos los caracteres');
99  1   3:1   240    Writeln;
100 1   3:1   248    WRITELN(' UN TOTAL DE ',IND-1,' VECTORES');
101 1   3:1   314    WRITELN(' Y UN TOTAL DE ',I-PCAR,' CARACTERES');
102 1   3:1   384
103 1   3:1   384    FOR J:=UCAR+1 TO 255 DO      (* Todos los caracteres no     *)
104 1   3:2   399       ind_car[J]:=ind_car[32]; (* definidos = espacio (32)    *)
105 1   3:2   428    (*endfor*)
106 1   3:1   428    FOR J:=0 TO PCAR-1 DO
107 1   3:2   441       ind_car[J]:=ind_car[32]
108 1   3:2   461    (*endfor*)
109 1   3:2   461
110 1   3:0   461 END; (*LEE_DEF*)
```

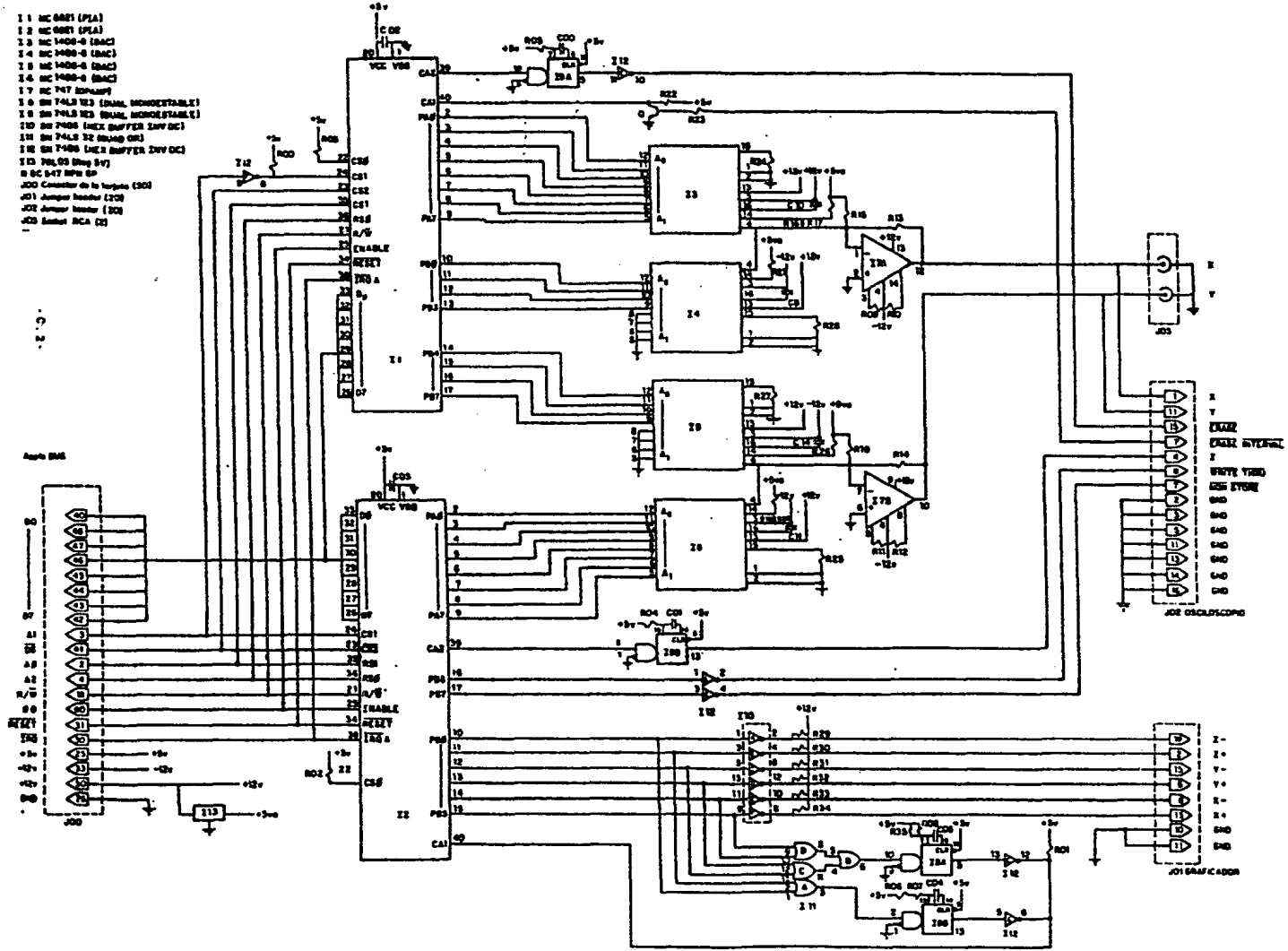
```
111  1    3:0   492 (*$P8)
112  1    4:0     1 PROCEDURE ESCTAB;
113  1    4:0     1
114  1    4:0     1 (*************************************************************************)
115  1    4:0     1 (*                                                                       *)
116  1    4:0     1 (*      Esta rutina escribe las tablas de definicion de caracteres al    *)
117  1    4:0     1 (* archivo SYSCAR2.  Ocupa 3 bloques exactamente.                        *)
118  1    4:0     1 (*                                                                       *)
119  1    4:0     1 (*************************************************************************)
120  1    4:0     1
121  1    4:0     1 VAR
122  1    4:0     1    I: INTEGER;
123  1    4:0     0 BEGIN (*ESC_TAB*)
124  1    4:1     0    Rewrite(outfile,'SYSCAR2');
125  1    4:1    20    I:=BLOCKWRITE(OUTFILE,INDCAR,1);     (* Salva los indices   *)
126  1    4:1    41    I:=BLOCKWRITE(OUTFILE,TABCAR,2,1);   (* Salva los vectores  *)
127  1    4:1    60    Close(outfile,Lock);
128  1    4:0    69 END; (*ESC_TAB*)
129  1    4:0    82
130  1    4:0    82 (*************************************************************************)
131  1    4:0    82 (*                                                                       *)
132  1    4:0    82 (*                        Programa principal                             *)
133  1    4:0    82 (*                                                                       *)
134  1    4:0    82 (*************************************************************************)
135  1    4:0    82
136  1    1:0     0 Begin (*SEN_TAB_CHAR*)
137  1    1:1     0    INICIA;
138  1    1:1    27    LEE_DEF;
139  1    1:1    29    ESC_TAB;
140  1    1:1    31
141  1    1:0    31 End.
```

Apéndice C

Diagramas

A continuación se muestran los digramas tanto físicos como eléctricos de la tarjeta de interfaz que fue construida para comunicar la microcomputadora Apple II con el osciloscopio de memoria Tektronix 611 y con el graficador incremental Houston Instrument DP-1.

La descripción del circuito y de su desarrollo está en el capítulo 3. Se incluyen las tablas de costos de materiales que se emplearon en la construcción de la tarjeta y una lista detallada de cada elemento.



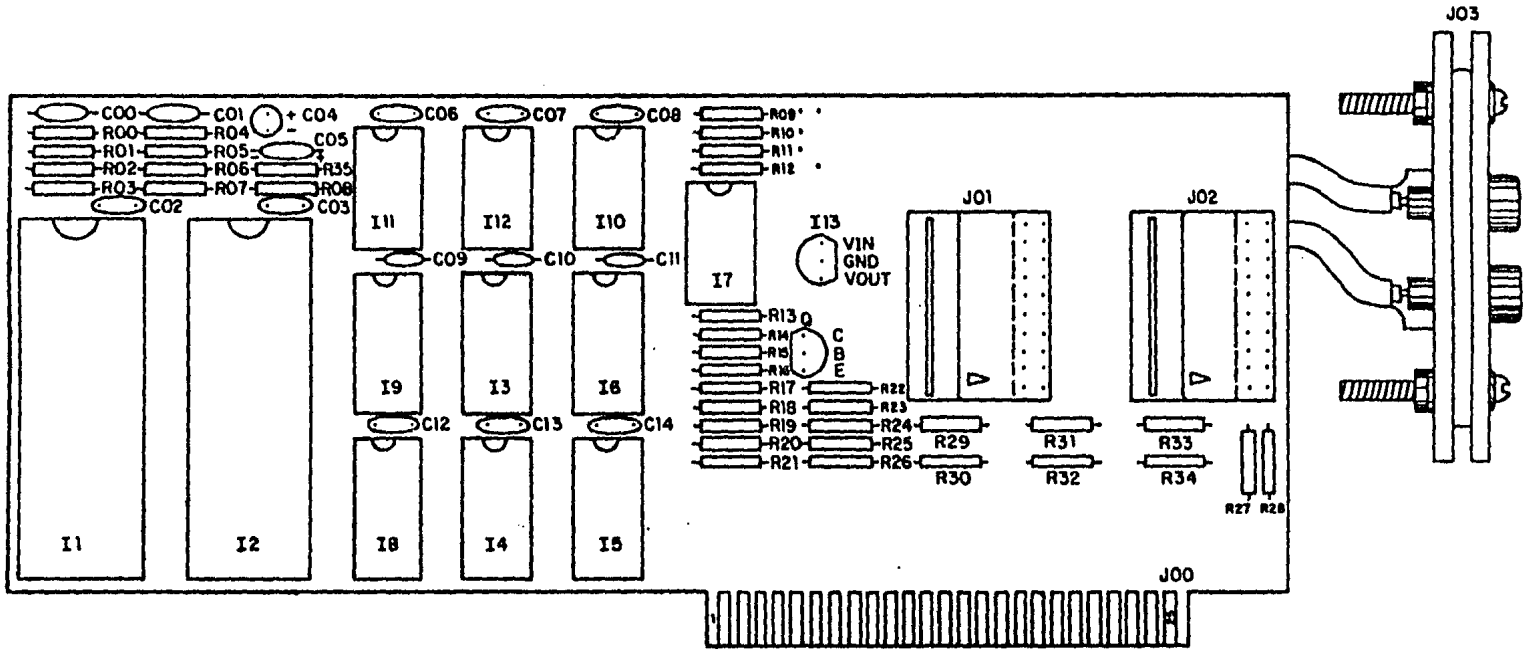


Diagrama de Ensamblado

Lista de partes.

Resistencia	kohms	watts
R00	10.00	0.25
R02	10.00	0.25
R03	10.00	0.25
R04	10.00	0.25
R05	1.00	0.25
R06	100.00	0.25
R07	10.00	0.25
R08	10.00	0.25
R09	10.00	0.25
R10	33.00	0.25
R11	33.00	0.25
R12	33.00	0.25
R13	2.20	0.25
R14	2.20	0.25
R15	2.20	0.25
R16	2.20	0.25
R17	2.20	0.25
R18	2.20	0.25
R19	2.20	0.25
R20	2.20	0.25
R21	15.00	0.25
R22	10.00	0.25
R23	10.00	0.25
R24	10.00	0.25
R25	10.00	0.25
R26	15.00	0.25
R27	1.50	0.25
R28	1.50	0.25
R29	0.47	0.50
R30	0.47	0.50
R31	0.47	0.50
R32	0.47	0.50
R33	0.47	0.50
R34	0.47	0.50
R35	100.00	0.25

Capacitor	uF
C00	0.10
C01	0.00
C02	0.01
C03	0.01
C04	10.00
C05	1.00
C06	0.01
C07	0.01
C08	0.01
C09	0.01
C10	0.01
C11	0.01
C12	0.01
C13	0.01

C Integrado	Tipo
I1	PIA MC6821
I2	PIA MC6821
I3	DAC MC1408-8
I4	DAC MC1408-8
I5	DAC MC1408-8
I6	DAC MC1408-8
I7	uA 747
I8	74LS123
I9	74LS123
I10	7406
I11	7406
I12	74LS72
I13	78L05

Varios	
Q	BC547
J00	Peine para bus APPLE
J01	Jumper header 90o 20 pins
J02	Jumper header 90o 20 pins
J03	Socket RCA dual

Total de Costos fijos

Cantidad	Tipo	Costo Unitario	Costo Total	Parciales
2	PIA	650.00	1300.00	
2	7406	100.00	200.00	
1	74LS32	90.00	90.00	
2	74LS123	200.00	400.00	
4	MC1408	490.00	1960.00	
1	uA747	150.00	150.00	
1	78L05	150.00	150.00	
1	BC547	30.00	30.00	
36	resistencias	10.00	360.00	
14	capacitor	30.00	420.00	
				5060.00
2	header(20)	500.00	1000.00	
1	con. RCA dual	300.00	300.00	
1	DB25 macho	1200.00	1200.00	
1	DB25 hembra	1400.00	1400.00	
1	Tarjeta APPLE	3500.00	3500.00	
				7400.00
4	Bases WW (14)	130.00	520.00	
6	Bases WW (16)	150.00	900.00	
2	Bases WW (40)	400.00	800.00	
				2220.00
			TOTAL	14680.00 MN

Apéndice D
Bibliografia

1. Apple Computer Inc.
   Apple Pascal. Languaje Reference Manual
   Apple Computer, 1980

2. Apple Computer Inc.
   Apple Pascal. Operating System Reference Manual
   Apple Computer, 1980

3. Apple Computer Inc.
   Apple II. Reference Manual
   Apple Computer, 1979

4. Armstrong,J.R.
   Design of a Graphic Generator for Remote Terminal Application.
   IEEE Trans. C-22(5) Mayo,1973

5. Beatty,J.C. and Booth,K.S.
   Tutorial: Computer Graphics
   IEEE EH0194-1 2da ed. 1982

6. Bresenham,J.E.
   Algorithm for Computer Control of a Digital Plotter
   IBM Systems Journal 4(1) 1965.

7. Brunner, Herbert
   Introduction to Microprocesors
   Prentice-Hall 1982

8. Carlbom,I. and Paciorеck,J.
   Planar Geometric Projections and Viewing Transformations. ACM Computing Surveys, vol 10 no.4
   ACM Dec. 1978

9. Chasen,S.H.
   Geometric Principles & Procedures for Computer Graphic Applications
   Prentice-Hall 1978

10. Harrington, Steven
    Computer Graphics, A programming aproach
    McGraw-Hill 1ra ed. 1983

11. Hobbs,I.C.
    Computer Graphics Display Hardware. Computer Graphics and Applications, vol 1, no. 1
    IEEE 1981

12. Houston Instrument
COMPLOT Model DP-1-5 . Instruction Manual
Houston Instrument 1969

13. Hungerford J.
Graphic Manipulatio using Matrices.
Programming Technics, vol 3. Numbers in Theory & Practice
Byte Books. McGraw-Hill. 1979

14. Lucido,A.P.
An Overview of Directed beam Graphics Display Hardware.
Computer, vol 11 no. 11
IEEE 1978

15. Motorola Inc.
Semiconductor Data Library, vol 6, Linear Integrated Circuits
Motorola 1975

16. Motorola Inc.
The Complete Motorola Microcomputer Data Library
Motorola 1978

17. Newman, William M. & Sproull, Robert F.
Principles of Computer Graphics
McGraw-Hill, 2ª ed. 1979.

18. Peckham, H.D.
Computer Graphics: Three-Dimensional Projections
Hewlett-Packard 1974

19. Posdamen J.
The Mathematics of Computer Graphics.
Programming Technics, vol 3. Numbers in Theory & Practice
Byte Books. McGraw-Hill. 1979

20. Rogers,D.F. and Adams,J.A.
Mathematical Elements for Computer Graphics
McGraw-Hill 1976

21. Tektronix
Type 611 Storage Display Unit. Instruction Manual
Tektronix 1968

22. Texas Instruments Inc.
The TTL Data Book for Design Engineers
Texas Instruments Inc. 2ª ed. 1981.