Universidad Don Bosco
Facultad De Ingeniería
Escuela De Ingeniería Electrónica

Trabajo de Graduación para obtener el grado de Ingeniero en Electrónica

Tema:

Diseño y desarrollo de un software prototipo para simulación de circuitos utilizando el método de balance armónico.

Presentado por:
Antonio José Orantes Moreno

Asesor:
Ingeniero Jorge López

Soyapango, Noviembre del 2006.
Universidad Don Bosco

Facultad De Ingeniería

Rector
Ing. Federico Miguel Huguet Rivera

Vicerrector
Padre Víctor Bermúdez

Secretario General
Inga. Yesenia Xiomara Martínez Oviedo

Decano de la Facultad de Ingeniería
Ing. Ernesto Godofredo Girón

Soyapango, Noviembre del 2006.
Universidad Don Bosco

Facultad De Ingeniería

Subcomité Evaluador del Trabajo de Graduación

Tema:

Diseño y desarrollo de un software prototipo para simulación de circuitos utilizando el método de balance armónico.

F._____
Ing. Calixto Rodríguez
JURADO

F._____
Lic. Jorge Mauricio Coto
JURADO

F._____
Ing. Wenceslao Rivas
JURADO

F._____
Ing. Jorge López
ASESOR

RESUMEN

Diseño y desarrollo de un software para la simulación de circuitos utilizando el método de balance armónico.

El balance armónico es una de las técnicas más importantes en el análisis de circuitos no lineales. Este tipo de análisis es aplicable a una amplia gama de problemas en circuitos de microondas como amplificadores de potencia, multiplicadores y mezcladores de frecuencia. El balance armónico calcula la respuesta de estado estable del circuito. [1]

Trabaja particularmente bien cuando el circuito tiene una mezcla de constantes de tiempo pequeñas y grandes y de hecho, fue originalmente propuesto para resolver los problemas inherentes al analizar dichos circuitos. [2]

SUMMARY

Design and developing of circuit simulation software using harmonic balance method.

Harmonic-balance analysis is one of the most important techniques for analyzing nonlinear circuits.

Harmonic balance analysis is applicable to a wide variety of problems such as microwave circuits and power amplifiers, frequency multipliers, and mixers. Harmonic-balance calculates a circuit steady state response. [1]

It works particularly well when a circuit has a mix of long and short time constants and, in fact, was originally proposed to solve the problems inherent in analyzing such circuits. [2]

Agradecimientos

A Lourdes Estefanía Orantes, a quién sin su ayuda y colaboración, esta obra no hubiera sido posible.
A Blanca Fátima Orantes, por su apoyo en la compilación de este trabajo.
A Cristian Rosa, por su amistad y por su apoyo incondicional en cualquier circunstancia.
Al Ingeniero Jorge López, por su asesoramiento en la planificación de este trabajo.
Al Ingeniero Wenceslao Rivas
A mi esposa
A mis padres

La vida es servicio.

CONTENIDO

INTRODUCCION

La ingeniería de microondas tiene que ver con todos aquellos dispositivos, componentes y sistemas que trabajen en el rango de frecuencias de 300 MHz a 300 GHz. Debido a tan amplio margen de frecuencias, tales componentes encuentran aplicación en diversos sistemas de comunicación. Ejemplo típico es un enlace de Radiocomunicaciones terrestre a 6 GHz en el cual detrás de las antenas emisora y receptora, hay toda una circuitería capaz de generar, distribuir, modular, amplificar, mezclar, filtrar y detectar la señal. Otros ejemplos lo constituyen los sistemas de comunicación por satélite, los sistemas radar y los sistemas de comunicación móviles.

La tecnología de semiconductores, que proporciona dispositivos activos que operan en el rango de las microondas, junto con la invención de líneas de transmisión planares; ha permitido la realización de tales funciones por medio de circuitos híbridos.

La técnica del balance armónico es una técnica usada en un gran número de simuladores comerciales para el análisis de este tipo de circuitos.

Actualmente, existe un aumento considerable en el uso de sistemas de comunicación móvil, lo que ha expandido la necesidad de software de simulación eficiente y preciso al ser aplicado a los circuitos de RF.

El diseño de circuitos de microondas, ha exigido un modelado preciso de los diferentes elementos que forman el circuito. De especial importancia son los dispositivos activos (MESFET, HEMT, HBT); pues conocer su comportamiento tanto en pequeña señal como en gran señal (régimen no lineal), es imprescindible para poder predecir la respuesta de un determinado circuito que haga uso de él. El análisis, modelado y simulación de éstos dispositivos, constituye un factor a considerar en el estudio del método del balance armónico.

En este trabajo se desarrolla un software que aplica el método del balance armónico para encontrar la solución de estado estable de los circuitos, que permitirá analizar el comportamiento circuitos de RF que puedan ser modelados a través de esta técnica.

Examinaremos a continuación el contenido del documento. En la parte I, se enfocara la atención en los objetivos que se tratan de alcanzar al desarrollar esta aplicación; En la parte II, se estudiará la teoría del análisis de circuitos por medio del balance armónico, tecnologías de simulación de circuitos, técnicas de diseño de interfaces de usuario y herramientas de desarrollo; en la parte III se detallará el diseño de el prototipo del simulador, elementos que se tomaron en cuenta en su creación. Por ultimo en la parte IV se analizan los diferentes tipos de circuitos característicos que pueden ser simulados con la herramienta, su complejidad y la capacidad del simulador de resolverlos adecuadamente y el tiempo utilizado para resolverlos.

PARTE I: PROPUESTA DE IMPLEMENTACION DE UNA HERRAMIENTA DE SIMULACION EMPLEANDO EL METODO DEL BALANCE ARMONICO.

1. DEFINICION DEL PROYECTO

1.1. DESCRIPCION

Diseño y desarrollo de un software para la simulación de circuitos utilizando el método de balance armónico.

El balance armónico es una de las técnicas más importantes en el análisis de circuitos no lineales. Este tipo de análisis es aplicable a una amplia gama de problemas en circuitos de microondas como amplificadores de potencia, multiplicadores y mezcladores de frecuencia. El balance armónico calcula la respuesta de estado estable del circuito. [1]

Trabaja particularmente bien cuando el circuito tiene una mezcla de constantes de tiempo pequeñas y grandes y de hecho, fue originalmente propuesto para resolver los problemas inherentes al analizar dichos circuitos. [2]

1.2. OBJETIVOS

Objetivo General

Diseñar un software prototipo para simulación de circuitos utilizando el método de balance armónico.

Objetivos Específicos

- ✓ Desarrollar una interfaz gráfica de diseño de diagramas esquemáticos de circuitos.
- ✓ Desarrollar un modulo de simulación de circuitos en el dominio de la frecuencia para componentes lineales que sea compatible con la interfaz gráfica desarrollada.
- ✓ Implementar los algoritmos de software del método del balance armónico.
- ✓ Desarrollar una interfaz gráfica de presentación de resultados.

1.3. JUSTIFICACION

De todos es conocido el simulador SPICE de la universidad de Berkeley y sus descendientes. Este software pionero se ha convertido en una herramienta fundamental que ha hecho posible los grandes avances en el diseño y desarrollo de circuitos electrónicos de las últimas décadas; sin embargo este software aún con todas sus grandes prestaciones, tiene algunas limitantes, como las que se mencionan a continuación:

La escasez de mejoras posteriores de dominio público a la versión original con fines académicos, cuya ultima versión el spice3F5 fue creada en 1993, funciona en DOS y Unix. Los desarrollos posteriores han sido realizados casi exclusivamente con fines comerciales, entre los más destacados podemos mencionar el PSPICE ahora parte de la suite ORCAD de Cadence Systems.

Además, al ser el SPICE una herramienta de análisis de circuitos en el dominio del tiempo, no es la más apropiada para análisis del estado estable de circuitos RF. La misma existencia del análisis de balance armónico implica que el método de análisis transitorio no es el adecuado para muchos tipos de circuitos. De hecho, los métodos son mutuamente complementarios: el balance armónico tiene éxito donde el análisis transitorio no, y el análisis transitorio usualmente supera al balance armónico para el tipo de problemas donde este usualmente es aplicado.

Algunos tipos de problemas pueden hacer las técnicas en el dominio del tiempo imprácticas. Por ejemplo las constantes de tiempo del circuito pueden ser grandes comparadas al periodo de la frecuencia fundamental de excitación, cuando constantes de tiempo grandes existen, se vuelve necesario continuar la integración numérica de las ecuaciones durante muchos – quizá miles - de ciclos de excitación, hasta que la parte transitoria de la respuesta ha decaído y solo la parte de estado estable permanece. Esta larga integración es un uso extravagante tanto del tiempo de computación como de la paciencia del ingeniero; es más, errores en el redondeo en la larga integración puede volverse excesivamente grandes y reducir la exactitud de la solución.

Por ultimo, cada elemento reactivo lineal o no lineal en el circuito añade una ecuación diferencial al conjunto de ecuaciones que lo describe. Un largo circuito puede tener muchos elementos reactivos, así el conjunto de ecuaciones que debe ser resuelto puede ser muy grande. Por esta razón, el análisis en el dominio del tiempo es notoriamente lento. [1]

Así pues nace la necesidad de una herramienta eficiente para el análisis de circuitos de RF que haga uso de técnicas de balance armónico para llegar a soluciones de estado estable de circuitos de RF y pueda ser utilizado libremente en la investigación académica.

1.4. ALCANCES

Se hará un análisis de las herramientas de desarrollo que se utilizaran para escoger las que permitan de manera adecuada el alcance de los objetivos.

El software deberá ser capaz de hacer un análisis armónico, es decir que puede calcular los productos de frecuencia originados en un circuito cuando se aplica una señal de gran amplitud.

Solo se implementará el método matemático básico del balance armónico para una sola frecuencia de trabajo fundamental.

El software deberá contar con una interfaz gráfica que permita dibujar el circuito a analizar.

Se deberán poder definir subcircuitos, es decir que mediante un símbolo se representa todo un circuito.

Se podrán definir nuevos modelos de elementos mediante ecuaciones. Se espera que como mínimo esta funcionalidad incluya operaciones aritméticas básicas, exponenciación y las funciones trigonométricas seno y coseno.

Deberán estar definidos los elementos pasivos y activos clásicos (resistencias, capacitancias, inductancias, fuentes dependientes e independientes).

1.5. LIMITACIONES

No se aplicarán optimizaciones innovadoras cuando el balance armónico converge hacia la solución por el método de Newton.

No se podrán simular circuitos cuyas ecuaciones generadas por el método de balance armónico excedan 180 variables desconocidas. (Aproximadamente 3 o 4 componentes no lineales)

Únicamente se podrán definir mediante ecuaciones fuentes dependientes y elementos no lineales controlados por voltaje.

PARTE II: ESTADO ACTUAL DE LAS HERRAMIENTAS DE SIMULACION DE CIRCUITOS DE MICROONDAS Y RADIOFRECUENCIA.

2. MARCO TEORICO

2.1. SPICE

SPICE es un acrónimo inglés de Simulation Program with Integrated Circuits Emphasis (Programa de simulación con énfasis en circuitos integrados). Fue desarrollado por la Universidad de California, Berkeley en 1975 por Larry Nagel bajo la dirección de su asesor de investigación Donald Pederson.

Es un software estándar internacional de facto cuyo objetivo es simular circuitos electrónicos analógicos compuestos por resistencias, condensadores, diodos, transistores, etc. Para ello hay que describir los componentes, describir el circuito y luego elegir el tipo de simulación (temporal, en frecuencia, en continua, parámetrico, Monte Carlo...).

Historia

Spice fue desarrollado en el laboratorio de investigación electrónica (Electronics Research Laboratory) de la Universidad de California, Berkeley por Larry Nagel bajo la dirección de su asesor de investigación Donald Pederson. SPICE1 fue derivado del programa CANCER (acrónimo de Computer Analysis of Nonlinear Circuits, Excluding Radiation).

A principios de 1970, Ron Rohrer espera desarrollar un programa de simulación para su trabajo en optimización. Los estudiantes de Rohrer, incluido Larry Ángel, crean como proyecto de clase el programa CANCER.

Cuando el director original del proyecto, el profesor Rohrer, abandono Berkeley, el profesor Pederson tomo el puesto de director. Éste nuevo director consiguió que el programa fuera reescrito de su antecesor CANCER, el cuál era un programa con licencia de propietario, para poder poner esta nueva versión del programa bajo dominio público.

SPICE1 tuvo su primera presentación en una conferencia de 1973. Fue programado en FORTRAN y usaba la técnica de análisis de nodos para construir el sistema de ecuaciones del circuito. Ésta técnica de análisis tenía inconvenientes al representar inductancias, fuentes de tensión sin referencia y fuentes controladas. Esta versión del programa contaba con pocos elementos; usaba un paso fijo para los análisis transitorios.

SPICE se convirtió en la herramienta estándar de simulación de circuitos de la industria.

En el año 1975 apareció la versión SPICE2, con la cual se popularizó su uso. Esta versión del programa también estaba compilada en FORTRAN, tenía más elementos, análisis transitorio con paso variable, usaba las técnicas de integración trapezoidal o integración de Gear, conseguía las ecuaciones de los circuitos por una técnica modificada del tradicional análisis de nodos, la que permitía resolver los inconvenientes de su versión anterior y usaba una innovación del programa FORTRAN que permitía controlar la memoria. Este último adelanto fue desarrollado por el estudiante de posgrado Ellis Coheb.

La última versión de SPICE en FORTRAN fue la versión 2G.6 en 1983. La siguiente versión, SPICE3, fue desarrollada en lenguaje C por Thomas Quarless y como director A. Richard en el año de 1989. La versión SPICE3 usaba la misma sintaxis que sus antecesoras y tenia una interfaz gráfica X Window.

Como un programa de código abierto, SPICE fue ampliamente usado. El código de SPICE fue distribuido desde sus comienzos bajo un costo por la Universidad de Berkeley, el cual retribuía el costo de las cintas magnéticas. El programa tenía la restricción de no poderse distribuir en países que no eran considerados amigos por los Estados Unidos. Actualmente el programa está cubierto por la licencia BSD.

SPICE promovió y sirvió de base para otros programas de simulación en las universidades y la industria. La primera versión comercial del SPICE fue ISPICE. La versión comercial más destacada de SPICE fué PSPICE, producto lanzado por MicroSim, la primera versión para PC de SPICE. Las versiones académicas de SPICE incluían XSPICE, desarrollada en el Instituto Tecnológico de Georgia, versión en la que se agregaron códigos de análisis analógicos y digitales y CIDER que permitía simular dispositivos semiconductores.

Esta descripción y detalles de la historia de SPICE fueron tomados de Wikipedia, La enciclopedia libre. (http://www.wikipedia.org)

2.2. BALANCE ARMONICO

El balance armónico es una técnica analítica para encontrar la respuesta de estado estable de circuitos no lineales.

Esta técnica utiliza la transformada de Fourier para obtener una representación del conjunto de ecuaciones que definen al circuito en el dominio de la frecuencia; los componentes diferenciales de las ecuaciones en el dominio del tiempo se simplifican y el sistema de ecuaciones se resuelve por simples operaciones de algebra matricial compleja. Luego la solución completa se transforma al dominio del tiempo obteniendo la respuesta final.

Para esto se necesitan las ecuaciones que modelan el comportamiento de los dispositivos que se utilizan en el circuito. Dispositivos lineales como resistencias y capacitores tienen representaciones simples en el dominio de la frecuencia. Son los dispositivos no lineales, como los diodos y transistores, los que representan un problema. Se necesita la transformada de Fourier de las ecuaciones del dispositivo, pero en general éstas son desconocidas para funciones no lineales arbitrarias.

Esto ha sido manejado tradicionalmente en los simuladores de balance armónico convirtiendo las señales en el dominio de la frecuencia a su equivalente en el dominio del tiempo. Estas señales han sido aplicadas a las ecuaciones utilizadas por los modelos de dispositivos no lineales del SPICE para generar una respuesta en el dominio del tiempo, que es convertida de regreso al dominio de la frecuencia para el uso del simulador. [3]

Además, no es necesario un método que resuelva las ecuaciones completamente, todo lo que se necesita es uno que mejore una solución estimada. Entonces, solo se necesita repetir el proceso una y otra vez, usando el resultado de cada iteración como el estimado inicial para el siguiente. Eventualmente, el error en la respuesta se reduce al punto donde se considera despreciable.

Actualmente, existe un fuerte consenso en que el método de Newton es el preferido para la simulación de balance armónico y virtualmente todos los simuladores de balance armónico lo usan.

Cuando un circuito tiene un gran número de elementos lineales y no lineales, éstos pueden ser agrupados para formar dos subcircuitos, uno lineal y el otro no lineal. [1]

ALGORITMO[*]

Los pasos del método pueden resumirse como sigue:

1. Crear un estimado inicial de la solución $\mathbf{V}(k\omega_p)$ en el dominio de la frecuencia. Este estimado puede ser escogido extremadamente al azar. Por ejemplo, $\mathbf{V}(k\omega_p)=0$ para todo k.
2. Usar las ecuaciones que representan el subcircuito lineal para obtener $\mathbf{I}_{LIN}(k\omega_p)$.
3. Aplicar la transformada inversa de Fourier a $\mathbf{V}(k\omega_p)$ para obtener $\mathbf{V}(t)$.
4. Usar las ecuaciones que representan el subcircuito no lineal para determinar $\mathbf{I}_{NL}(t)$.
5. Aplicar la transformada de Fourier a $\mathbf{I}_{NL}(t)$ para obtener $\mathbf{I}_{NL}(k\omega_p)$.
6. Verificar que se satisface la ley de Kirchhoff $\mathbf{I}_{LIN}(k\omega_p) = -\mathbf{I}_{NL}(k\omega_p)$. Probablemente, el resultado será insatisfactorio. Definir una función de error f_k para cada armónico donde:

$$f_k = I_{LIN}(k\omega_p) + I_{NL}(k\omega_p) \qquad k = 0, 1, \ldots K$$

7. Modificar $\mathbf{V}(k\omega_p)$ y repetir el proceso desde el paso 2. Usar algún método apropiado (por ejemplo, el método de Newton) que pueda ser de confianza para reducir $| f_k |$.
8. Continuar hasta que todos los K + 1 errores f_k son despreciablemente pequeños.

Habiendo introducido el método para resolver simples problemas de circuitos no-lineales, se debe generalizar a circuitos más grandes. Examinaremos 'circuitos mono-tono', aquellos que tienen excitaciones periódicas a una sola frecuencia fundamental. Esto incluye excitaciones periódicas, no sinusoidales, mientras puedan ser expresadas como una serie de Fourier de una sola dimensión.

[*]Este algoritmo y detalles del método de balance armónico subsiguientes fueron tomados de [1].

PARTICIONAMIENTO DEL CIRCUITO.

En general, los circuitos de RF y microondas tienen un gran número de elementos lineales y no-lineales. Éstos pueden ser agrupados para formar dos subcircuitos, uno lineal y el otro no-lineal. El subcircuito lineal puede ser tratado como un elemento multipuerto y puede ser descrito por sus parámetros Y, sus parámetros S, o por cualquier otra matriz multipuerto. Los elementos no lineales son modelados por sus características globales I/V, y deben ser analizados en el dominio del tiempo. Así, el circuito es reducido a una red multipuerto (N+2), con elementos no lineales conectados a N de sus puertos y fuentes de voltaje conectadas a los otros dos puertos. [El (N+1) y el (N+2)-avo puerto representan, el puerto de entrada y salida en una red de dos puertos. Usualmente una fuente sinusoidal esta conectada únicamente a solo uno de esos puertos; sin embargo se conectan fuentes a ambos puertos para obtener una expresión más general.] $Z_s(\omega)$ y $Z_L(\omega)$, las impedancias de la fuente y la carga, respectivamente, son "absorbidas" dentro del subcircuito lineal; aún se encuentran en serie con los puertos de entrada y salida, y si por algún propósito es necesario, pueden ser representadas como entidades separadas. El voltaje y la corriente en cada puerto pueden ser expresados en el dominio del tiempo o de la frecuencia; por causa de los elementos no lineales, sin embargo, los voltajes y corrientes de los puertos tienen componentes en los diferentes armónicos de la frecuencia de excitación. Y aunque en teoría un número infinito de armónicos existe en cada puerto, debemos asumir para aplicar este método que la componente DC y los primeros K armónicos, describen todos los voltajes y corrientes adecuadamente. Consecuentemente, todos los armónicos de orden superior pueden ser ignorados.

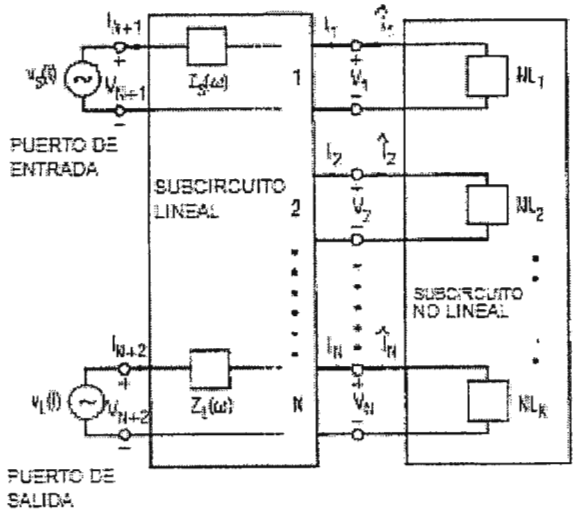


Fig. 1 – Diagrama Esquemático de un circuito divido en una parte lineal y otra no lineal para ser analizado por el método del balance armónico.

El circuito ha sido analizado satisfactoriamente cuando ya sea el voltaje de estado estable o las formas de onda de las corrientes en cada puerto son conocidas. Alternativamente, el conocimiento de los componentes de frecuencia en cada puerto constituye una solución, porque los componentes de frecuencia y las formas de onda en el tiempo están relacionados por las series de Fourier. Si, por ejemplo, conocemos los voltajes en los puertos en el dominio de la frecuencia, podemos usar la matriz de parámetros Y del subcircuito lineal para encontrar las corrientes en los puertos. Las corrientes en los puertos también pueden ser encontradas aplicando la transformada inversa de Fourier a los voltajes para obtener las formas de onda en el dominio del tiempo y calculando las formas de onda de corriente a través de las ecuaciones I/V de los elementos no lineales. La idea del balance armónico es encontrar un conjunto de formas de onda de voltaje en los puertos (o alternativamente, los componentes armónicos de cada voltaje) que producen las mismas corrientes tanto en las ecuaciones de la red lineal como de la red no lineal; esto es, las corrientes satisfacen la ley de corrientes de Kirchhoff. Cuando ese conjunto es encontrado, debe ser una solución.

(Nótese que cuidadosamente se dice una solución y no "la solución". Los circuitos no lineales, en general, tienen múltiples soluciones. Afortunadamente, en resultados prácticos, una sola solución usualmente domina el resultado. Sin embargo, debemos permanecer atentos a la posibilidad de múltiples soluciones en cualquier circuito no lineal.)

Si expresamos los componentes de frecuencia de las corrientes de los puertos como vectores, la ley de corrientes de Kirchhoff requiere que

$$\begin{bmatrix} I_{1,0} \\ I_{1,1} \\ I_{1,2} \\ \cdots \\ I_{1,K} \\ I_{2,0} \\ I_{2,1} \\ \cdots \\ I_{2,K} \\ \cdots \\ I_{N,K} \end{bmatrix} + \begin{bmatrix} \hat{I}_{1,0} \\ \hat{I}_{1,1} \\ \hat{I}_{1,2} \\ \cdots \\ \hat{I}_{1,K} \\ \hat{I}_{2,0} \\ \hat{I}_{2,1} \\ \cdots \\ \hat{I}_{2,K} \\ \cdots \\ \hat{I}_{N,K} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 0 \end{bmatrix} \qquad (2.1)$$

donde $I_{n,k}$ es un fasor, el k-avo componente armónico de la corriente en el puerto n en el subcircuito lineal. $\hat{I}_{n,k}$ con el acento circunflejo, es la corriente de puerto en el subcircuito no lineal. Esta ecuación muestra la forma general de los vectores de voltaje, corriente y carga; todos los vectores utilizan esta forma a menos que sea indique de otra manera. Los vectores incluyen solo componentes de frecuencia positivos, debido a

que los componentes de frecuencia negativos, siendo los complejos conjugados de los elementos positivos, pueden encontrarse inmediatamente de ser necesario. Eliminar los componentes negativos de frecuencia de la ecuación (2.1) reduce su complejidad considerablemente.

Primero debemos considerar el subcircuito lineal. Las ecuaciones de admitancia son

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \dots \\ I_N \\ I_{N+1} \\ I_{N+2} \end{bmatrix} = \begin{bmatrix} Y_{1,1} & Y_{1,2} & \dots & Y_{1,N} & Y_{1,N+1} & Y_{1,N+2} \\ Y_{2,1} & Y_{2,2} & \dots & Y_{2,N} & Y_{2,N+1} & Y_{2,N+2} \\ Y_{3,1} & Y_{3,2} & \dots & Y_{3,N} & Y_{3,N+1} & Y_{3,N+2} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Y_{N,1} & Y_{N,2} & \dots & Y_{N,N} & Y_{N,N+1} & Y_{N,N+2} \\ Y_{N+1,1} & Y_{N+1,2} & \dots & Y_{N+1,N} & Y_{N+1,N+1} & Y_{N+1,N+2} \\ Y_{N+2,1} & Y_{N+2,2} & \dots & Y_{N+2,N} & Y_{N+2,N+1} & Y_{N+2,N+2} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \dots \\ V_N \\ V_{N+1} \\ V_{N+2} \end{bmatrix} \tag{2.2}$$

El vector de corriente I de (2.1), ha sido escrito como un conjunto de subvectores, donde

$$\mathbf{I}_n = \begin{bmatrix} I_{n,0} \\ I_{n,1} \\ \dots \\ I_{n,K} \end{bmatrix} \tag{2.3}$$

Esto es $\boldsymbol{I}_n$ es el vector de corrientes por cada armónico en el puerto n. Similarmente,

$$\mathbf{V}_n = \begin{bmatrix} V_{n,0} \\ V_{n,1} \\ \dots \\ V_{n,K} \end{bmatrix} \tag{2.4}$$

Los elementos de la matriz de admitancia $\boldsymbol{Y}_{m,n}$ en (2.2) son todas submatrices: cada submatriz es una diagonal, cuyos elementos son los valores $Y_{m,n}$ para cada armónico de la frecuencia de excitación fundamental, $k\omega_p$, $k=0...K$:

$$Y_{m,n} = diag[Y_{m,n}(k\omega_p)] \qquad k = 0, 1, 2, \dots, K \tag{2.5}$$

esto es,

$$\mathbf{Y}_{m,n} = \begin{bmatrix} Y_{m,n}(0) & 0 & 0 & \dots & 0 \\ 0 & Y_{m,n}(\omega_p) & 0 & \dots & 0 \\ 0 & 0 & Y_{m,n}(2\omega_p) & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & Y_{m,n}(K\omega_p) \end{bmatrix} \tag{2.6}$$

V_{N+1} y V_{N+2} los vectores de excitación, tienen la forma,

$$\begin{bmatrix} \mathbf{V}_{N+1} \\ \mathbf{V}_{N+2} \end{bmatrix} = \begin{bmatrix} V_{b1} \\ V_z \\ 0 \\ 0 \\ \cdots \\ V_{b2} \\ 0 \\ \cdots \\ 0 \end{bmatrix} \quad (2.7)$$

donde V_{b1} y V_{b2} son los voltajes DC en los puertos N+1 y N+2, respectivamente, y V_z es el voltaje de excitación en el puerto N+1. La ecuación (2.7) implica que el puerto N+1 incluye una excitación DC y una fuente a la frecuencia fundamental, mientras que el puerto N+2 incluye únicamente una componente de DC. Esta es la situación usual; corresponde por ejemplo, a un amplificador FET que tiene una compuerta y un drenaje polarizado y una excitación de compuerta. Un dispositivo de dos terminales normalmente solo tiene una fuente de polarización, y en ese caso el puerto N+2 podría no existir. Por otro lado, un IC muy complejo podría tener un número de fuentes de DC y probablemente varias fuentes de RF también. Finalmente, si la excitación fuera periódica pero no sinusoidal, el vector a la derecha de (2.7) podría incluir componentes armónicos en vez de ceros. La extensión en esos casos es directa.

Particionar la matriz **Y** en (2.2) nos da una expresión para **I**, el vector de corrientes en los puertos 1 hasta N:

$$\begin{bmatrix} \mathbf{I}_1 \\ \mathbf{I}_2 \\ \cdots \\ \mathbf{I}_N \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{1,N+1} & \mathbf{Y}_{1,N+2} \\ \mathbf{Y}_{2,N+1} & \mathbf{Y}_{2,N+2} \\ \cdots & \cdots \\ \mathbf{Y}_{N,N+1} & \mathbf{Y}_{N,N+2} \end{bmatrix} \begin{bmatrix} \mathbf{V}_{N+1} \\ \mathbf{V}_{N+2} \end{bmatrix} + \begin{bmatrix} \mathbf{Y}_{1,1} & \mathbf{Y}_{1,2} & \cdots & \mathbf{Y}_{1,N} \\ \mathbf{Y}_{2,1} & \mathbf{Y}_{2,2} & \cdots & \mathbf{Y}_{2,N} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{Y}_{N,1} & \mathbf{Y}_{N,2} & \cdots & \mathbf{Y}_{N,N} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \cdots \\ \mathbf{V}_N \end{bmatrix} \quad (2.8)$$

ó

$$\mathbf{I} = \mathbf{I}_s + \mathbf{Y}_{N\times N}\mathbf{V} \quad (2.9)$$

donde Y_{NxN} es la submatriz NxN de Y correspondiente a las primeras N filas y columnas. I_s representa un conjunto de fuentes de corriente en paralelo con los primeros N puertos; el primer termino de la matriz en (2.9) transforma las excitaciones del puerto de entrada y salida en este conjunto de fuentes de corriente, así los puertos (N+1) y (N+2) no necesitan volver a ser considerados más adelante. La representación equivalente se muestra en la figura.

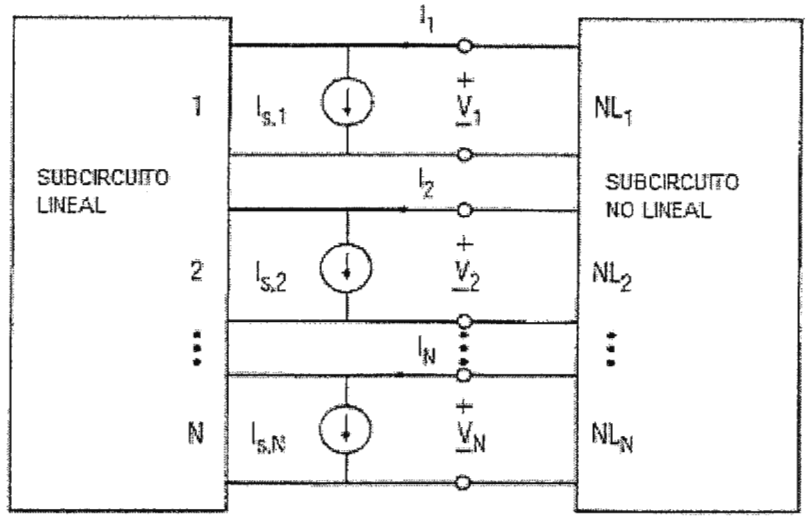


Fig. 2 – Figura en la que se muestran los equivalentes Thevenin de las corrientes y voltajes en los puertos.

Esta transformación permite expresar las ecuaciones del balance armónico como funciones de las corrientes en solo los primeros N puertos, aquellos conectados a los elementos no lineales.

EL SUBCIRCUITO NO LINEAL

Las corrientes de los elementos no lineales, representadas por el vector de corriente a la derecha de (2.1), pueden resultar de capacitores no lineales, resistencias, fuentes dependientes, u ocasionadamente inductores. Debido a que inductores no lineales aparecen raramente en circuitos de RF y microondas, no es necesario considerarlos en este momento. Además, se asume que los elementos no lineales son todos controlados por voltaje. Estas asunciones no constituyen una limitante severa, puesto que se pueden emplear métodos simples para evitarlas. Aplicando la transformada inversa de Fourier a los voltajes en cada puerto se obtienen las formas de onda de los voltajes en el dominio del tiempo en cada puerto:

$$\mathcal{F}^{-1}\{\mathbf{V}_n\} \rightarrow v_n(t) \tag{2.10}$$

La corriente en una conductancia no lineal o una fuente de corriente dependiente es:

$$i_{g,n}(t) = f_n(v_1(t), v_2(t), \ldots, v_N(t)) \quad (2.11)$$

Aplicando la transformada de Fourier resulta

$$\mathcal{F}\{i_{g,n}(t)\} \rightarrow \mathbf{I}_{G,n} \quad (2.12)$$

y el vector

$$\mathbf{I}_G = \begin{bmatrix} \mathbf{I}_{G,1} \\ \mathbf{I}_{G,2} \\ \ldots \\ \mathbf{I}_{G,N} \end{bmatrix} \quad (2.13)$$

Sustituyendo (2.9), (2.10) *, y (2.13) en (2.1) se obtiene la expresión

$$\mathbf{F}(\mathbf{V}) = \mathbf{I}_s + \mathbf{Y}_{N\times N}\mathbf{V} + j\Omega\mathbf{Q} + \mathbf{I}_G = \mathbf{0} \quad (2.14)$$

La ecuación (2.14) constituye una prueba para determinar en qué momento un conjunto de componentes de voltaje de prueba es el correcto; eso es, si F(V)=0, entonces V es una solución valida. También representa una ecuación que puede resolverse para obtener el vector de voltajes en los puertos, V. F(V), llamado el vector de error en las corrientes, representa la diferencia entre la corriente calculada de las subredes lineal y no lineal, en cada puerto y en cada armónico, a partir del vector de solución que se puso a prueba.

EL SUBCIRCUITO LINEAL

Existen muchos métodos para generar la matriz de N puertos de admitancias de un circuito lineal. Quizás el más simple es generar una matriz indefinida y convertirla en una matriz de puertos. El proceso es directo y puede ser implementado rápidamente en una computadora. Por supuesto, se debe producir una matriz para cada frecuencia armónica en el análisis.

La matriz indefinida se crea colocando un patrón de admitancias para cada elemento en la matriz. Por ejemplo, si tenemos un elemento simple de dos terminales conectado desde el nodo n1 a n2, cuya admitancia es Y, añadimos Y a las posiciones (n1, n1) y (n2, n2) y añadimos –Y a las posiciones (n1, n2) y (n2, n1). Se utilizan procedimientos similares para elementos más complejos, como fuentes dependientes y multipuertos. Un circuito de N nodos resulta en una matriz de NxN.

Para convertir la matriz de admitancias indefinida en una matriz de admitancias de puertos, se debe crear una matriz de impedancias de puertos e invertirla. Para obtener la matriz de impedancias, primero se selecciona los nodos correspondientes al puerto 1, se excitan con corriente unitaria y se miden los voltajes entre los nodos que representan a cada puerto. Esto produce la primera columna la matriz de impedancia.

Moviendo la excitación al puerto 2 se produce la segunda columna y procediendo de esta manera hasta el último puerto se produce la matriz completa.

Específicamente, supóngase que el nodo 1 es el terminal positivo del puerto 1 y el nodo 3 es el negativo. La ecuación matricial es

$$\begin{bmatrix} Y_{1,1} & Y_{1,2} & Y_{1,3} & \cdots & Y_{1,N} \\ Y_{2,1} & Y_{2,2} & Y_{2,3} & \cdots & Y_{2,N} \\ Y_{3,1} & Y_{3,2} & Y_{3,3} & \cdots & Y_{3,N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Y_{N,1} & Y_{N,2} & Y_{N,3} & \cdots & Y_{N,N} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \cdots \\ V_N \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ \cdots \\ 0 \end{bmatrix} \qquad (2.15)$$

Resolviendo (2.15) se obtienen todos los Vn. Debido a que el puerto 1 es excitado por una corriente unitaria, Z1, un 1 en la matriz de impedancias de puertos es simplemente el voltaje en el puerto: $Z_{1,1}$=V1-V3. Similarmente, si el puerto 2 estuviera conectado a los nodos 4 y 7, $Z_{2,1}$=V4-V7. Cuando todos los $Z_{n,1}$ han sido encontrados, se mueve la excitación al puerto 2 (I4=1 y I7=-1, para continuar el ejemplo) y se repite el proceso. Finalmente, la matriz de impedancias debe ser invertida. Estro raramente es un proceso largo, debido a que el número de puertos es usualmente mucho menor que el número de nodos.

Primero, la solución de (2.15) es obtenida de la mejor manera al factorizar la matriz. Esto permite que la ecuación de matrices sea resuelta repetidamente, por cada nuevo vector derecho (de corriente), utilizando solamente una operación de sustitución de fondo simple, en lugar de un largo proceso de reducción de matrices. Como la matriz indefinida de impedancias es muy dispersa, se puede reducir dramáticamente el tiempo de cómputo requerido utilizando técnicas para matrices dispersas, con relación a los métodos tradicionales. Segundo, frecuentemente ocurre que ciertos elementos, como transformadores y fuentes de voltaje dependientes, no tienen una representación en forma de admitancia y no pueden ser usados directamente para formar la matriz de admitancias. Una solución común es aproximar los elementos irrealizables con otros realizables; por ejemplo, una fuente dependiente de voltaje controlada por voltaje puede ser aproximada como una fuente dependiente de corriente controlada por voltaje con una pequeña resistencia en paralelo con la fuente de corriente. Finalmente, un problema persistente al formular la matriz es que romper las conexiones con los elementos no lineales casi siempre resulta en nodos desconectados. Si nada fuera hecho para acomodarlos, la matriz indefinida seria singular. La solución convencional es interconectar todos los nodos por resistencias de alto valor, pero eso conlleva generalmente a una matriz pobremente condicionada. Una mejor solución es colocar en cada puerto una resistencia moderada, típicamente de 100 Ohmios, cuando se forma la matriz indefinida. Para remover las resistencias, simplemente se sustrae sus admitancias de la diagonal principal de la matriz de admitancias de los puertos.

ALGORITMOS DE SOLUCION

El único problema restante, y que constituye la parte más trabajosa de todo el proceso, consiste en resolver (2.14) para obtener V. Cada uno de los K+1 componentes de frecuencia de V en cada puerto es una variable, y cada componente tiene una parte real y una imaginaria. Entonces, existen 2N(K+1) variables a ser determinadas (los componentes de DC no tienen partes imaginarias; sin embargo, normalmente resulta más fácil en el análisis agregar las partes imaginarias que aparecen en los componentes DC que tratar de evitarlos). Por ejemplo, un análisis de un multiplicador de frecuencia FET, puede incluir elementos no lineales en tres puertos, y tiene ocho armónicos significativos más DC en cada puerto. Así, N=3, K=8, y existen 54 variables en (2.14)!. Resolver un conjunto de ecuaciones que contiene tantas variables, especialmente en vista de la naturaleza no lineal del circuito, no es una tarea sencilla.

Una buena cantidad de algoritmos han sido propuestos para resolver (2.14). Algunos de estos son aplicaciones obvias de técnicas numéricas existentes pero otros muestran una gran ingenuidad. Hoy, existe un fuerte consenso en que el método de Newton es el preferido para la simulación de balance armónico, y virtualmente todos los simuladores de balance armónico lo usan. Describiremos el método de Newton en una de las secciones siguientes.

OPTIMIZACION

A primera vista, resolver (2.14) parece como si fuera un problema de optimización. Entonces, se debería ser capaz de resolverla minimizando la magnitud del error cuadrático de la función de error en las corrientes: esto es minimizar ε donde

$$\varepsilon = \mathbf{F}^{*T}(\mathbf{V})\mathbf{F}(\mathbf{V}) \tag{2.16}$$

y *T representa transpuesta compleja conjugada del vector. Las librerías de subrutinas científicas frecuentemente incluyen una rutina de optimización funcional de propósito general. Así, con este método, se tiene ya disponible una porción grande y difícil del programa. Sin embargo, la función de error en (2.16) destruye una gran cantidad de información acerca de la contribución individual de cada variable al error; entonces las rutinas de optimización pueden tener problemas de convergencia, especialmente cuando un gran número de variables tienen que ser optimizadas simultáneamente. Debido a estas limitaciones, la optimización es una estrategia razonable solo para problemas relativamente sencillos, en los cuales la facilidad de programación compensa la ineficiencia.

METODOS DE RELAJACION

Se ha propuesto un buen número de métodos de relajación, los cuales son fáciles de implementar y intuitivamente satisfactorios. Como el nombre implica, los métodos de relajación usan algoritmos simples

que encaminan a los voltajes gradualmente (o relajadamente) hacia la solución. Frecuentemente los métodos son en su mayoría heurísticos. Por ejemplo, el método de la bisección, usado para encontrar el cero de una función no lineal y descrito virtualmente en todos los textos básicos de análisis numérico, es un tipo de método de relajación. Una ventaja de estos métodos es que son simples de implementar, frecuentemente no requieren la generación de derivadas de la función I/V, ni siquiera un estimado inicial de la solución.

Dos de los métodos de relajación más populares son los de Hicks y Khan y el de Kerr. Aunque a primera vista son muy diferentes, es posible demostrar que estos métodos son equivalentes, y que el método de Kerr es una forma refleja del de Hicks y Khan.

Los métodos de Relajación son en su mayoría obsoletos. Los principales problemas son (1) características de convergencia impredecibles (y con frecuencia decepcionantes), y (2) son inaplicables a circuitos grandes. Su importancia hoy es principalmente histórica, pues representan un paso importante en el desarrollo de la tecnología de simulación de circuitos no lineales. De una manera más practica. Estos podrían ser todavía útiles como una solución rápida y sucia de un problema especial.

EL METODO DE NEWTON

El método de Newton es un poderoso algoritmo para encontrar los ceros de un conjunto de funciones no lineales de varias variables. Debido a que el método del balance armónico implica encontrar los ceros de F(V), el método de Newton es una opción obvia como algoritmo de solución. El método de Newton es una técnica iterativa; ésta estima el cero de una función usando su primera derivada para extrapolar al eje de la variable independiente. Su poder viene del uso de todas las derivadas de F(V), con respecto a los componentes de voltaje de V, en cada iteración. El método de Newton se utiliza virtualmente en todos los software modernos de balance armónico.

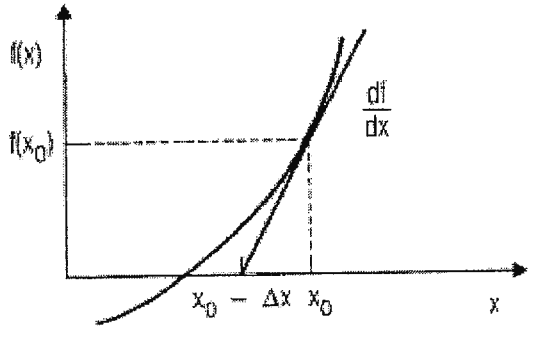


Fig. 3 - El método de Newton de una dimensión en el proceso encuentra el cero haciendo repetido estimados lineales de su posición.

El proceso iterativo se ilustra más fácilmente aplicándolo a un problema de una sola dimensión. La figura nos muestra una función de una variable, f(x), y el estimado de Newton del cero. Se puede escribir para la extrapolación lineal,

$$f(x_0) - \left.\frac{df}{dx}\right|_{x=x_0} \Delta x = 0 \qquad (2.17)$$

si $f(x_0)$ y su derivada son conocidas, (2.17) puede resolverse fácilmente para obtener Δx, y se encuentra un nuevo estimado del cero como $x_0 - \Delta x$. La función y su derivada se evalúan nuevamente, y se estima el cero, y el proceso se repite hasta que se determina el cero con la precisión requerida. Es importante darse cuenta que el método de Newton puede fallar. Por ejemplo, en la siguiente figura se muestra qué puede pasar cuando el proceso comienza cerca de un mínimo relativo de la función: el segundo estimado del cero retorna el proceso al punto cercano al punto original y oscila dentro de una región limitada. El proceso puede verse atrapado fácilmente en esa región y nunca encontrar el cero. Puede también aterrizar cerca del mínimo, donde df/dx es casi cero, así el próximo estimado del cero podría encontrarse sin esperanza muy lejos de la ubicación real del cero u ocasionar una excepción numérica. El método de Newton puede ser confiable de converger únicamente cuando ha comenzado suficientemente cerca del cero. "Suficientemente cerca" puede ser difícil de determinar. Pero generalmente significa que (1) la función tiene un buen comportamiento entre el cero y el punto de inicio, y (2) no es extremadamente no lineal. Por esto el método de Newton requiere generalmente un conocimiento aproximado del lugar del cero antes de comenzar.

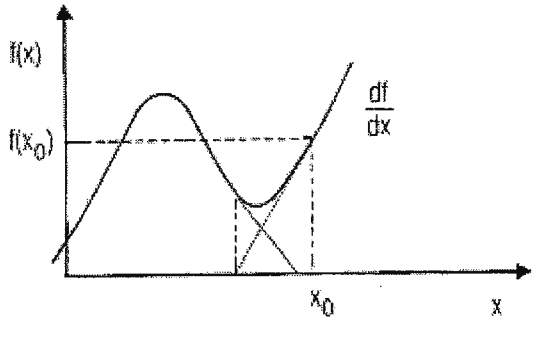


Fig. 4 - El método de Newton puede quedar atrapado en una singularidad en su búsqueda del cero.

SOLUCION POR EL METODO DE NEWTON DE LA ECUACION DEL BALANCE ARMONICO

PROCESO ITERATIVO Y FORMULACION DEL JACOBIANO

La función de error F(V), es de hecho un conjunto de funciones multidimensional, y se deben encontrar todos los ceros simultáneamente. El análogo de la ecuación (2.17) aplicado a un conjunto de funciones multidimensionales es

$$\mathbf{F}(\mathbf{V}^p) - \left.\frac{d\mathbf{F}(\mathbf{V})}{d\mathbf{V}}\right|_{\mathbf{V}=\mathbf{V}^p} \Delta\mathbf{V} = \mathbf{0} \tag{2.18}$$

donde Vp es el p-avo estimado del vector de solución. Con

$$\mathbf{V}^p - \mathbf{V}^{p+1} = \Delta\mathbf{V} \tag{2.19}$$

El vector actualizado Vp+1 es

$$\mathbf{V}^{p+1} = \mathbf{V}^p - \left(\frac{d\mathbf{F}(\mathbf{V})}{d\mathbf{V}}\right)^{-1}\mathbf{F}(\mathbf{V}^p) \tag{2.20}$$

La ecuación (2.20) envuelve la derivada de un vector, F, con respecto a otro vector, V. El resultado es una matriz, llamada el Jacobiano de F, designado J_F:

$$\mathbf{J}_\mathbf{F} = \left.\frac{d\mathbf{F}(\mathbf{V})}{d\mathbf{V}}\right|_{\mathbf{V}=\mathbf{V}^p} \tag{2.21}$$

El Jacobiano contiene las derivadas de todos los componentes del vector de error con respecto a los componentes de V. De este modo, contiene información de la sensibilidad a los cambios en cada componente de F que resultan de cambios en cualquier componente de V. Esta cantidad de información es la máxima posible de un sistema linealizado de ecuaciones.

La forma del Jacobiano es

$$\mathbf{J_F} = \begin{bmatrix} \frac{\partial F_{1,0}}{\partial V_{1,0}} & \frac{\partial F_{1,0}}{\partial V_{1,1}} & \frac{\partial F_{1,0}}{\partial V_{1,2}} & \cdots & \frac{\partial F_{1,0}}{\partial V_{1,K}} & \frac{\partial F_{1,0}}{\partial V_{2,0}} & \cdots & \frac{\partial F_{1,0}}{\partial V_{N,K}} \\ \frac{\partial F_{1,1}}{\partial V_{1,0}} & \frac{\partial F_{1,1}}{\partial V_{1,1}} & \frac{\partial F_{1,1}}{\partial V_{1,2}} & \cdots & \frac{\partial F_{1,1}}{\partial V_{1,K}} & \frac{\partial F_{1,1}}{\partial V_{2,0}} & \cdots & \frac{\partial F_{1,1}}{\partial V_{N,K}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_{1,K}}{\partial V_{1,0}} & \frac{\partial F_{1,K}}{\partial V_{1,1}} & \frac{\partial F_{1,K}}{\partial V_{1,2}} & \cdots & \frac{\partial F_{1,K}}{\partial V_{1,K}} & \frac{\partial F_{1,K}}{\partial V_{2,0}} & \cdots & \frac{\partial F_{1,K}}{\partial V_{N,K}} \\ \frac{\partial F_{2,0}}{\partial V_{1,0}} & \frac{\partial F_{2,0}}{\partial V_{1,1}} & \frac{\partial F_{2,0}}{\partial V_{1,2}} & \cdots & \frac{\partial F_{2,0}}{\partial V_{1,K}} & \frac{\partial F_{2,0}}{\partial V_{2,0}} & \cdots & \frac{\partial F_{2,0}}{\partial V_{N,K}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_{2,K}}{\partial V_{1,0}} & \frac{\partial F_{2,K}}{\partial V_{1,1}} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial F_{2,K}}{\partial V_{N,K}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_{N,K}}{\partial V_{1,0}} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial F_{N,K}}{\partial V_{N,K}} \end{bmatrix} \quad (2.22)$$

Los elementos del Jacobiano son las derivadas

$$\frac{\partial F_{n,k}}{\partial V_{m,l}} \quad (2.23)$$

donde n y m son los índices de los puertos (1,N), y k y l son los índices de los armónicos (0,..., K). Determinar estas cantidades requiere de algún esfuerzo. Se comienza por tomar la derivada de (2.14):

$$\mathbf{J_F} = \frac{d\mathbf{F}(\mathbf{V})}{d\mathbf{V}} = \mathbf{Y}_{N\times N} + \frac{\partial \mathbf{I}_G}{\partial \mathbf{V}} + j\Omega\frac{\partial \mathbf{Q}}{\partial \mathbf{V}} \quad (2.24)$$

Entonces, se debe encontrar $\partial I_G/\partial V$ y $\partial Q/\partial V$. Se comienza con el primero. $\partial I_G/\partial V$, una matriz, tiene la misma forma que $\partial F/\partial V$; esto es, sus elementos son $\partial I_{n,k}/\partial V_{m,l}$.

Primero, notamos que las series de Fourier pueden ser expresadas en diferentes modos; aquí se ocupará la que sigue:

$$i_n(t) = \sum_{k=-K}^{K} I_{n,k}\exp(jk\omega_p t) \quad (2.25)$$

donde $k\neq 0$. Los componentes son

$$I_{n,k} = \frac{1}{T}\int_0^T i_n(t)\exp(-jk\omega_p t) \quad (2.26)$$

Sin embargo, $I_{n,k}$, k>0, es la mitad del valor del favor del componente de corriente en $k\omega_p$. Para determinar sus derivadas sobre $V_{m,l}$, nosotros debemos considerar ambos los componentes de frecuencia positivos y negativos de $V_{m,l}$. Entonces,

$$dI_{n,k} = \frac{\partial I_{n,k}}{\partial V_{m,l}} dV_l + \frac{\partial I_{n,k}}{\partial V_{m,-l}} dV_{-l} \tag{2.27}$$

Donde

$$dV_{-l} = dV_l^* \tag{2.28}$$

de (2.26), el componente de Fourier $\partial I_{n,k}/\partial V_{m,l}$ es

$$\frac{\partial I_{n,k}}{\partial V_{m,l}} = \frac{1}{T}\int_0^T \frac{\partial i_n}{\partial v_m}\frac{\partial v_m}{\partial V_{m,l}} \exp(-jk\omega_p t) \tag{2.29}$$

Con

$$v_m(t) = \sum_{l=-K}^{K} V_{m,l} \exp(jl\omega_p t) \tag{2.30}$$

Vemos inmediatamente que

$$\frac{\partial v_m}{\partial V_{m,l}} = \exp(jl\omega_p t) \tag{2.31}$$

Sustituyendo (2.31) en (2.29) da

$$\frac{\partial I_{n,k}}{\partial V_{m,l}} = \frac{1}{T}\int_0^T \frac{\partial i_n}{\partial v_m} \exp(-j(k-l)\omega_p t) \tag{2.32}$$

Para los componentes de frecuencia negativa

$$\frac{\partial v_m}{\partial V_{m,-l}} = \exp(-jl\omega_p t) \tag{2.33}$$

Así

$$\frac{\partial I_{n,k}}{\partial V_{m,-l}} = \frac{1}{T}\int_0^T \frac{\partial i_n}{\partial v_m} \exp(-j(k+l)\omega_p t) \tag{2.34}$$

Encontramos que las derivadas son componentes de la expansión de Fourier de la forma de onda derivativa.

Los términos (2.34) tienden a ocurrir a frecuencias altas que aquellos en (2.32) y entonces tenemos menos efectos en el proceso de convergencia. En muchas situaciones simples, (2.34) pueden ser descartados. Sin embargo, incluyéndolos mejora notablemente la convergencia en circuitos fuertemente no lineales.

Todavía se necesita poner todo esto en la misma forma que en (2.3). Primero, notamos que los términos derivativos son complejos, así podemos escribir

$$\frac{\partial I_{n,k}}{\partial V_{m,l}} = G^R_{k-l} + jG^I_{k-l}$$

$$\frac{\partial I_{n,k}}{\partial V_{m,-l}} = G^R_{k+l} + jG^I_{k+l} \qquad (2.35)$$

Donde $G_p = G_p^R + jG_p^I$ es la p-avo componente de la serie de Fourier de $g(t) = \partial I_n/\partial V_m$ evaluado a $v_m(t)$, y

$$dV_{m,l} = dV^R_{m,l} + jdV^I_{m,l}$$
$$dV_{m,-l} = dV^R_{m,l} - jdV^I_{m,l} \qquad (2.36)$$

Sustituyendo en (2.27) nos da:

$$dI^R_{n,k} = (G^R_{k-l} + G^R_{k+l})dV^R_{m,l} + (-G^I_{k-l} + G^I_{k+l})dV^I_{m,l}$$
$$dI^I_{n,k} = (G^I_{k-l} + G^I_{k+l})dV^R_{m,l} + (G^R_{k-l} - G^R_{k+l})dV^I_{m,l} \qquad (2.37)$$

Así, cada término en (2.22) debe ser tratado como una submatriz 2x2,

$$\begin{bmatrix} dI^R_{n,k} \\ dI^I_{n,k} \end{bmatrix} = \begin{bmatrix} G^R_{k-l} + G^R_{k+l} & -G^I_{k-l} + G^I_{k+l} \\ G^I_{k-l} + G^I_{k+l} & G^R_{k-l} - G^R_{k+l} \end{bmatrix} \begin{bmatrix} dV^R_{m,l} \\ dV^I_{m,l} \end{bmatrix} \qquad (2.38)$$

Desafortunadamente, no es posible presentar el (2.37) como una simple, ecuación compleja, Peor aún cuando $k = l = 0$, la matriz en (2.38) se convierte

$$\begin{bmatrix} G_0 & 0 \\ 0 & 0 \end{bmatrix} \qquad (2.39)$$

En la segunda fila y la segunda columna del Jacobiano, para cada puerto, ambos son ceros, el Jacobiano es, por lo tanto, singular y (2.18) ¡no puede ser resuelto! El problema se presenta del hecho que los componentes dc deben tener partes imaginarias cero, para evitar esta dificultad podemos fijar a las posiciones (2,2) de (2.58) algún valor arbitrario; por ejemplo,

$$\begin{bmatrix} G_0 & 0 \\ 0 & G_0 \end{bmatrix} \tag{2.40}$$

Entonces, mientras $G_0 \neq 0$ y las partes imaginarias del voltaje dc y los actuales componentes se fijan constantemente a cero, todo debe estar bien. Por supuesto, otra solución es simplemente eliminar la fila y la columna.

El tercer término de (2.24) es manejado en la misma manera. Para obtener $\partial Q_{n,k}/\partial V_{m,l}$, usamos la señal - pequeña de la forma de la onda de capacitancia. $\partial q_{n,}/\partial v_{m,}$ en lugar de la forma de onda de conductancia $\partial i_{n,}/\partial v_{m,}$ en (2.32), y proceda idénticamente. El resultado puede ser escrito,

$$\begin{bmatrix} dI^R_{n,k} \\ dI^I_{n,k} \end{bmatrix} = \begin{bmatrix} 0 & -k\omega_p \\ k\omega_p & 0 \end{bmatrix} \begin{bmatrix} C^R_{k-l} + C^R_{k+l} & -C^I_{k-l} + C^I_{k+l} \\ C^I_{k-l} + C^I_{k+l} & C^R_{k-l} - C^R_{k+l} \end{bmatrix} \begin{bmatrix} dV^R_{m,l} \\ dV^I_{m,l} \end{bmatrix} \tag{2.41}$$

Donde $C_p = C_p^R + jC_p^I$ representa la p-avo componente de la serie de Fourier de c(t)= $\partial q_{n,}/\partial v_{m,}$ la matriz contiene los términos $k\omega_p$ que representan una sola entrada de la matriz Ω (2.10)*.

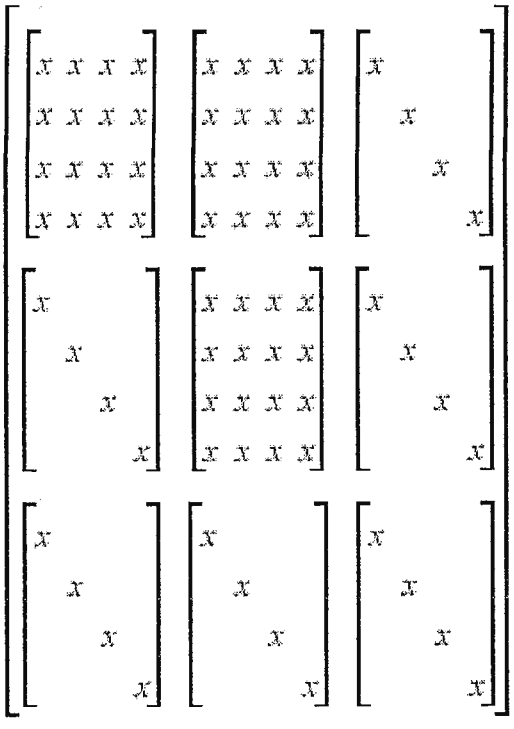
Desde que hemos separado las partes reales y imaginarias de $dI_{n,k}$ y $\partial V_{m,l}$, necesitamos tratar la matriz Y similarmente. La submatriz 2x2 que representa el parámetro de Y tiene la forma

$$Y_{n,m}(k\omega_p) \rightarrow \begin{bmatrix} Y^R_{n,m}(k\omega_p) & -Y^I_{n,m}(k\omega_p) \\ Y^I_{n,m}(k\omega_p) & Y^R_{n,m}(k\omega_p) \end{bmatrix} \tag{2.42}$$

Para dc, solamente necesitamos el componente de la matriz en la posición (1.1).

ESTRUCTURA JACOBIANA.

El Jacobiano consiste en una matriz $N \times N$ de submatrices cuadradas, cada una de las cuales tiene una dimensión $K + 1$. Cada submatriz representa los componentes armónicos para un puerto particular del elemento no lineal, es decir, Si la corriente o carga en el puerto *n* depende sobre el voltaje del puerto *m,* la submatriz *(n, m)* se llena con los términos de Fourier. Se añadido a cada submatriz *(n, m)* una matriz diagonal de $Y_{n,m}$, en cada armónico $k\omega_{p,}$ así, algunas submatrices están llenas y otras son diagonales. Puede ser posible que algunas estén vacías.



(2.43)

La ecuación (2.43) muestra una forma posible de la matriz cuando *N = 3 y K = 3. L*os bloques llenados a lo largo de la diagonal principal ocurren cuando un puerto tiene un elemento no lineal de dos terminales conectado con él. Los bloques llenados fuera de la diagonal resultan de fuentes de corriente controladas no lineales. La matriz diagonal en la posición (2.3) implica que el voltaje en este puerto es un voltaje de control para una de las otras no linealidades, pero no hay ningún elemento no lineal conectado con él.

ANALISIS DEL BALANCE ARMONICO Y METODOS RELACIONADOS.

Es interesante notar que esta matriz es bastante dispersa, así que los métodos de matriz dispersa pueden ser útiles al solucionarla. Los métodos de matriz dispersa desafortunadamente, por lo general trabajan bien solamente cuando la matriz es extremadamente dispersa y el Jacobiano, en los problemas de balance armónico, no es usualmente lo bastante disperso para beneficiarse más que modestamente de tales métodos. También puede ser posible aprovechar la estructura especial de esta matriz en otras, maneras más elegantes de acelerar su factorización.

SELECCIÓNAR EL NÚMERO DE ARMÓNICOS Y MUESTRAS EN EL DOMINIO DEL TIEMPO.

En teoría, las formas de la onda generadas en el análisis no lineal tienen un número infinito de armónicos, entonces una descripción completa de la operación de un circuito no lineal aparecería requerir vectores de corriente y de voltaje de dimensiones infinitas. Afortunadamente las magnitudes de los componentes de frecuencia disminuyen invariablemente al aumentar la frecuencia; De otra forma las formas de la onda en el tiempo representarían infinita energía. Por consiguiente es siempre posible ignorar todos los armónicos encima de algún un número máximo. El cual hemos designado como *K.* Una importante consideración en la

implementación de un análisis del balance armónico es la selección de *K*. Seleccionando valores de *K* demasiados pequeños produce resultados de poca exactitud, y a menudo poca convergencia; inversamente, seleccionando valores de K demasiado grandes retarda el proceso de solución, el cual bajo las mejores circunstancias es un consume mucho tiempo, y incrementa el uso de la memoria de la computadora.

Quizás el criterio más simple para seleccionar *K* es considerar las magnitudes de las capacitancias en el circuito equivalente del dispositivo. Sobre una cierta frecuencia las susceptancias capacitivas son mayores que las conductancias del circuito, así que efectivamente son cortocircuitos, y sus componentes de voltaje son insignificantemente pequeños. Este criterio puede ser fácilmente aplicado a un diodo, por ejemplo donde la capacitancia de unión cortocircuita todos los componentes de voltaje a través de la única otra no linealidad, la unión resistiva.

Otra importante consideración en la selección de K es la fuerza de la no linealidad dominante y la magnitud de la excitación. Es frecuentemente posible generar un circuito equivalente simplificado para el dispositivo no lineal y aproximar los voltajes y las corriente en él lo suficientemente bien para formar a groso modo un estimado de las magnitudes de los componentes de frecuencia. Por ejemplo en un fuertemente manejado FET, uno puede frecuentemente aproximar el voltaje de compuerta como una sinusoide y la corriente de drenaje como un tren de pulsos rectangulares. La longitud de cada pulso es igual a la longitud del tiempo que el voltaje de compuerta esta arriba de V_t, el voltaje de conducción. Un tren de pulsos de una serie de Fourier es encontrado fácilmente y por que el pulso actual de corriente de drenaje es invariablemente más suave que un pulso rectangular. Esta serie establece un límite superior a las magnitudes relativas de los componentes armónicos de la corriente de drenaje. Nosotros conocemos que una no linealidad de n-avo grado genera solamente n armónicos directamente, sin embargo armónicos mas altos son posibles como producto de la mezcla entre estas frecuencias. Estos armónicos mas altos son usualmente mucho más débiles que aquellos generados directamente, así que raramente tiene sentido tomar un *K* mucho más grande que la no linealidad de más alto grado en el circuito. En caso contrario, si nosotros deseamos determinar los niveles de los armónicos altos debemos cuidadosamente modelar las no linealidades del circuito usando polinomios (u otras funciones que tienen expansión polinomial) de un grado lo suficientemente grande para generar estos armónicos.

La naturaleza del problema a ser resuelto frecuentemente pone algunas limitaciones en *K*. Si la corriente o el voltaje en algunos armónicos *k han* de ser encontrados. $K > k$ es un requerimiento obvio. Esto es quizás menos obvio que los errores introducidos por la eliminación de armónicos son usualmente mayores en armónicos altos que en los pequeños. Así que realmente debemos escoger K considerablemente mayor. Calcular las magnitudes de los altos armónicos con precisión también requiere que la convergencia pueda ser más completa, así que los errores en todos los componentes de los armónicos altos sean pequeños.

Las propiedades del algoritmo de la transformada rápida de Fourier (FFT), usada para obtener los componentes de frecuencia de las formas de la onda en el tiempo, también pone limitaciones en K. Un requerimiento del FFT es que el numero de armónicos debe ser siempre un entero potencia de dos. (Formas del FFT han sido propuestas que no tienen este requerimiento, pero no son usadas mucho en simuladores de balance armónicos.) El segundo requerimiento, una consecuencia del teorema de muestreo, es que el número de las muestras en el dominio del tiempo debe ser el doble que el número de los componentes de frecuencia. No es necesario incluir todos estos armónicos en las ecuaciones de balance armónico; es posible usar, por ejemplo, solamente 10 armónicos en las ecuaciones pero calcular 16 utilizando el FFT. Esto es esencial, sin embargo, usar todas la muestras en el dominio del tiempo requeridas por el FFT. Es más, hay buenas razones para usar incluso más muestras en el dominio del tiempo.

Usando el número mínimo de muestras requeridas por el teorema de muestreo puede resultar en errores de aliasing, donde los altos armónicos subestimados afectan la precisión de los componentes de bajos armónicos. La manera más simple de minimizar los errores de aliasing es sobre – muestrear, esto es, usar una tasa de muestreo 25% o 30% más alta que el mínimo, ó 2.5 a 2.6 veces el número mínimo de muestras requeridas. En el ejemplo anterior, 10 armónicos requieren una tasa de muestreo de 25 o 26 muestras en el dominio del tiempo por ciclo. La siguiente potencia de dos es 32, así 32 muestras deben ser usadas, con 16 armónicos en el FFT. Los 6 armónicos más altos son simplemente descartados en la formulación del vector de error en la corriente.

El uso que se le pretende dar al análisis también afecta el número de armónicos que deben ser considerados. Podemos ver que un análisis de matrices de conversión que envuelve productos de mezcla alrededor de el k-avo armónico del oscilador local requiere K= 2k armónicos, mas el componente dc en el análisis de gran señal. Obtener una buena precisión en el análisis IM de mezcladores o otros circuitos que varían con el tiempo frecuentemente requiere incluso más armónicos, sin embargo, frecuentemente es muy difícil estimar K, de antemano. En estos problemas uno podría determinar K empíricamente incrementándolo hasta que resultados consistentes sean obtenidos independientemente de K.

¿Cuál es el efecto de descartar los armónicos k>K? Esto implica que el voltaje a través de los elementos no lineales a estas frecuencias es cero, así, la impedancia vista de la red equivalente desde las terminales de los elementos es un corto circuito. Es algunas veces posible, aunque raramente practico, formular un caso dual como el que hemos descrito. Donde las corrientes de los elementos y no los voltajes, son las variables independientes. En este caso, el descartar armónicos pondría las corrientes a cero, lo que implicaría circuitos abiertos a altas frecuencias.

MÉTODOS MATRICIALES PARA RESOLVER (2.18)

Resolver (2.18) implica resolver una serie de ecuaciones lineales. Ciertamente, no hay escasez de literatura que describe los métodos para resolver ecuaciones lineales. Sin embargo ciertos métodos han sido especialmente útiles para el análisis de balance armónico, así que los examinaremos aquí:

SOLUCIONES DIRECTAS

Las soluciones directas o "completas" son las que se encuentran descritas en los más básicos textos de álgebra lineal. Más específicamente, cuando los métodos de reducción de norma son usados, el más práctico es la descomposición LU, ya que una solución puede ser obtenida por multiplicar múltiples lados derechos con una única factorización.

El principio detrás de la descomposición LU es muy simple. Supongamos que debemos resolver la ecuación matricial:

$$\mathbf{Ax} = \mathbf{b} \quad (2.44)$$

Para x, donde A es la matriz y x, b son vectores. Factorizamos la matriz A en una matriz triangular inferior, L, en la que las entradas arriba de la diagonal son cero, y una matriz triangular superior, U, en la que las entradas debajo de la diagonal, son cero. Entonces, tenemos que:

$$\mathbf{LUx} = \mathbf{b} \quad (2.45)$$

Sí

$$\mathbf{Ux} = \mathbf{m} \quad (2.46)$$

Donde m es un vector y resolvemos, en dos pasos,

$$\mathbf{Lm} = \mathbf{b} \quad (2.47)$$

$$\mathbf{Ux} = \mathbf{m} \quad .$$

Los dos pasos en (2.47) pueden ser resueltos por operaciones de sustitución opuesta, los cuales son de poco costo computacional; virtualmente todo el trabajo se encuentra en factorar la matriz. Una vez se encuentra factorizada, puede ser utilizada repetidamente para resolver (2.44) a muy bajo costo. Esta propiedad es apreciable especialmente en el análisis de balance armónico. Otra buena propiedad es que la factorización LU puede ser realizada "en el lugar". Esto es, sin usar más memoria que la requerida para guardar la matriz original A. A es destruida en el proceso de factorización y reemplazada por L, y U.

Soluciones directas no son escalables para el análisis de balance armónico. El tiempo requerido para factorizar la matriz es proporcional aproximadamente al cubo de su dimensión, entonces al doblar el tamaño de la matriz se incrementa el tiempo computacional en un factor de ocho. Esta característica claramente hace que las soluciones directas sean poco practicas para los análisis de circuitos grandes.

SOLUCIONES DE MATRIZ DISPERSA.

Una matriz dispersa es aquella que contiene casi solo elementos que valen cero. Convencionalmente las soluciones de matriz dispersa usan la descomposición LU para factorar la matriz pero aprovechan la dispersión de ciertos tipos de matrices para mejorar la eficiencia. La mejora viene de evitar la necesidad de multiplicar y añadir una gran cantidad de elementos que valen cero. En la mayoría de estos métodos, los elementos que valen cero, no son guardados, así resulta también en un ahorro de memoria.

Cuando una matriz dispersa es reducida, Esta tiende a llenarse, esto es, elementos que originalmente fueron cero, son convertidos en números que no son ceros. Evitar tales "llenados" es importante para el éxito del método de una matriz dispersa. Usualmente, hay un intercambio entre el llenado y el optimo pivoteo, así los métodos de matriz dispersa pueden ser no tan robustos como los métodos directos, o completos.

Si la matriz es muy dispersa, el tiempo requerido para factorizarla puede ser proporcional a tan poco como su dimensión elevada a 1.5 potencia. Este es una considerable mejora sobre los métodos directos. Sin embargo, es raramente posible lograr una adecuada dispersión en el Jacobiano para lograr este tipo de rendimiento.

LAS TÉCNICAS DEL SUB-ESPACIO DE KRYLOV Y LA ITERACIÓN INEXACTA DE NEWTON.

Las técnicas del sub-espacio de Krylov son una clase de métodos iterativos para resolver sistemas de ecuaciones lineales dispersas. Ahora existen un consenso general que una técnica llamada, el residuo mínimo generalizado. O GMRES, es el preferido, de muchos disponibles, para el análisis de balance armónico. Aunque algunos de los materiales en esta sección pueden ser validos para otros métodos, estos podrían ser considerados específicos a GMRES.

Los métodos iterativos minimizan el residuo, r, de (2.44)

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} \quad (2.48)$$

Donde $\hat{x}$ es un estimado de la solución. Este puede ser hecho eficientemente solo cuando $\hat{x}$ puede ser estimado con al menos moderada precisión, cuando r no es muy grande. Para obtener estas condiciones, nosotros debemos precondicionar la matriz, esto es, multiplicar por un estimado de la inversa. Por lo tanto.

$$\mathbf{PAx} = \mathbf{Pb} \quad (2.49)$$

Donde P, el precondicionante, es un estimado de $\mathbf{A}^{-1}$. Otros métodos de Krylov requieren diferentes tipos de precondicionamiento; en GMRES, es también posible realizar precondicionamiento derecho:

$$\mathbf{A}\mathbf{P}^{-1}\mathbf{y} = \mathbf{b} \quad (2.50)$$

Obtener **y**, y luego resolver

$$\mathbf{y} = \mathbf{P}\mathbf{x} \quad (2.51)$$

En este caso es esencial que (2.51) sea solucionable con poco costo computacional. En el análisis de balance armónico, un precondicionamiento apropiado es la inversa de la matriz de admitancia del subcircuito lineal, la cual es generada en el proceso de crear la matriz de puerto Y, y necesita ser invertida solamente una vez en el proceso de solución. Otra opción para el precondicionamiento es la inversa de una versión severamente recortada del Jacobiano, pero esta debe ser regenerada periódicamente cuando el Jacobiano cambia.

Una ventaja de las técnicas de Krylov es que (2.18) no necesita ser completamente resuelto en cada iteración, el proceso de iterativo necesita solo proceder hasta que ΔV decrece la función de error. Este acercamiento a la solución, llamado Newton inexacto, puede proveer significativamente una eficiencia mejorada. En las primeras iteraciones del balance armónico, un paso de Newton es, en el mejor de los casos, un pobre estimado del cero, así una que una solución precisa de (2.18) tiene poco valor. A cada paso, la matriz necesita solo ser resuelta hasta que se alcance una condición predeterminada sobre la solución mejorada, el criterio usual es:

$$\|\mathbf{F}(\mathbf{V}) - \mathbf{J}(\mathbf{V})\Delta\mathbf{V}\| \leq \alpha\|\mathbf{F}(\mathbf{V})\| \quad (2.52)$$

Donde α es seleccionada al inicio del p-ava iteración del balance armónico y es:

$$\alpha = \frac{\|\mathbf{F}(\mathbf{V}^p) - \mathbf{F}(\mathbf{V}^{p-1}) + \mathbf{J}(\mathbf{V}^{p-1})\Delta\mathbf{V}^{p-1}\|}{\|\mathbf{F}(\mathbf{V}^{p-1})\|} \quad (2.53)$$

Poniendo α= 0 en (2.52) corresponde a ordinarias iteraciones de Newton, las cuales son exactas.
Se ha observado que las soluciones de Krylov tienen un rendimiento inferior a las soluciones directas al manejar matrices jacobianas pobremente condicionadas.

CONDICIONAMIENTO DE MATRICES.

Es bien conocido que, si A es singular y b no es cero, (2.44) no tiene una solución única. En muchos casos, sin embargo, A no es singular, pero esta tan cerca de ser singular que la solución no es clara. En este caso, nosotros decimos que la matriz esta mal condicionada, y el resultado es una solución imprecisa **X**.

La precisión de la solución es controlada por el número de condicionamiento, K(A). Entonces:

$$\frac{\delta\|\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(\mathbf{A})\frac{\delta\|\mathbf{b}\|}{\|\mathbf{b}\|} \tag{2.54}$$

Donde $\|x\|$ es la norma máxima del vector N- dimensional, x.

$$\|\mathbf{x}\| = \max|x_i| \qquad 1 \leq i \leq N \tag{2.55}$$

Y

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{2.56}$$

Donde

$$\|\mathbf{A}\| = \max\frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \tag{2.57}$$

Sobre todos los x que no son cero.

Una interpretación no rigurosa de (2.54) es que cierto error fraccional en b, el cual puede venir de una pérdida de precisión numérica o limitando el grado de las series armónicas en el FFT, resulta en un error proporcionalmente mayor en x. Cuando K(A) es grande, un error en un componente particular de b; Por ejemplo b_1, no simplemente afecta a x_1, pero puede crear errores en cualquiera, o mas comúnmente en todos los componentes de x. Si K (A) es muy grande, como es en el caso donde A es casi singular. El error puede ser más grande que x en si misma, volviendo la solución inútil. El fallo en la convergencia en un

análisis de balance armónico basado en Newton es frecuentemente causado por un Jacobiano mal condicionado.

Es inquietante lo fácil, que en un análisis de balance armónico, se puede encontrar un Jacobiano mal condicionado o una matriz Y.

REDUCCIÓN DE NORMA.

La necesidad de métodos de reducción de norma puede ser ilustrada al revisar la operación del método de Newton en una dimensión. Por ejemplo, considere el problema de encontrar el cero de la función mostrada en la figura. Aunque las no linealidades son débiles, y no hay mínimos relativos cerca del cero, el proceso puede todavía quedar atrapado cerca del cero, o incluso divergir.

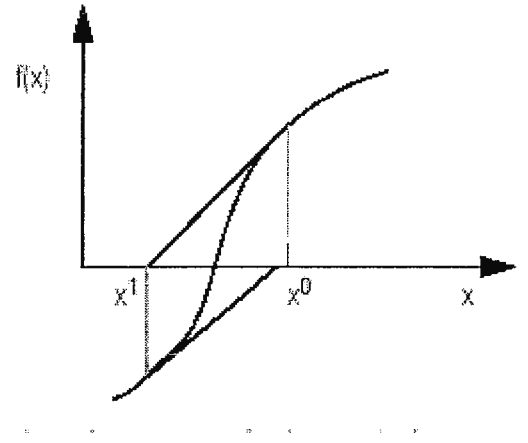


Fig. 5 - Una situación donde los métodos de reducción de norma previenen la falla del método de Newton. El método de Newton puede fallar cuando las funciones tienen un punto de inflexión cerca del cero. En este caso, reduciendo el tamaño del paso puede asegurar el éxito.

Sin embargo, suponga que, en lugar de un completo paso de Newton en (2.17), nosotros tomamos un paso parcial, precisamente,

$$\Delta x = \beta f(x_0) \left.\frac{df}{dx}\right|^{-1}_{x = x_0}$$

(2.58)

Donde β es una constante, entre cero y uno, que nosotros podemos ajustar como sea necesario. Ahora, si ajustamos β para obtener el tamaño del paso que minimice *f(X)* y simplemente continuamos los pasos de

Newton. A menos que nuestro algoritmo para seleccionar β sea extraordinariamente malo, este proceso siempre encuentra el cero en la situación ilustrada en la figura anterior.

En el caso multidimensional, (2.18) es modificado para formar

$$\mathbf{V}^{p+1} = \mathbf{V}^{p} - \beta\left(\frac{d\mathbf{F}(\mathbf{V})}{d\mathbf{V}}\right)^{-1}\mathbf{F}(\mathbf{V}^{p}) \tag{2.59}$$

Donde β, como en el caso de una dimensión, es una constante real. El proceso usual para ajustar β es comenzar con un paso completo de Newton (β =1). Si este paso reduce el error de la corriente, es retenido, Si no, β es reducido por otro factor, y el proceso es repetido hasta que el error se reduzca. El proceso se convierte en una búsqueda directa y lineal sobre una sola variable β. Note que no es necesario resolver (2.18) cada vez que β es modificado. β simplemente se multiplica por ΔV y F(V) es recalculado. Entonces el proceso es mucho menos costoso computacionalmente que un paso completo de Newton.

OPTIMIZANDO LA CONVERGENCIA Y LA EFICIENCIA.

Incluso bajo las mejores circunstancias, problemas de convergencia son a veces encontrados en el algoritmo, especialmente en circuitos que son complejos, fuertemente no lineales, o fuertemente excitados. Varios métodos han sido desarrollados para mejorar la convergencia y hacer el proceso más eficiente.

La eficacia del método de Newton viene del uso de todas las derivadas de la función de error con respecto a cada componente de frecuencia en cada puerto. En principio, esto permite seleccionar un vector ΔV que decremente todos los componentes de la función de error en cualquier paso. Como resultado el método de Newton es capaz de lograr la convergencia con un muy largo numero de variables, siempre y cuando las no linealidades no sean muy fuertes. La desventaja de este algoritmo es la gran cantidad de memoria y tiempo de computación requerido para generar el Jacobiano y resolver la ecuación matricial (2.18). El Jacobiano es una matriz cuadrada de dimensión 2N (K + 1), en el ejemplo de un circuito FET teniendo tres puertos no lineales y ocho armónicos mas dc, el Jacobiano es 54 x 54. Porque el jacobiano es complejo, resolver (2.18) para este caso tan simple envuelve resolver un conjunto de 54x 54 ecuaciones lineales reales. No es inusual para un único RFIC tener varios cientos de transistores y, con excitaciones de multitono, decenas o cientos de componentes de frecuencia. Analizar tales circuitos es una proposición computacionalmente costosa.

En muchos casos, la matriz entera, el vector solución, y el vector actualizado debe permanecer en memoria simultáneamente, por lo tanto, El método de Newton requiere una gran cantidad de memoria de computadora. (Muchas de los elementos de la matriz son cero en problemas grandes, así el uso de las técnicas de matriz dispersa pueden aminorar esta situación. De algún modo, como también el uso de los métodos de Krylov)

Finalmente, generar el Jacobiano requiere tomar un gran número de derivadas. Muchos modelos de dispositivo de estado sólido son muy complejos, y expresiones para las derivadas requiere muchas veces el tiempo de computación de funciones estáticas I/ V y Q/ V. Evaluar tales funciones puede ser una parte significativa de el tiempo requerido para el análisis completo.

RECORTANDO LAS MATRICES. REMOVIENDO VALORES PEQUEÑOS DEL JACOBIANO.

Por que el método escala pobremente, con el tamaño de la matriz, la factorización directa del Jacobiano es utilizada raramente en vez de eso, algún tipo de método de matriz dispersa, ya sea convencional o iterativo, es preferido. Tales métodos son mas eficientes cuando la matriz es bastante dispersa, ellos puede ser peores que métodos directos si la dispersión de la matriz es inapropiada.

Frecuentemente, muchos de los elementos del Jacobiano son muy pequeños y tienen poco efecto sobre la actualización de Newton, así tiene sentido incrementar la dispersión simplemente eliminando todos los elementos cuyas magnitudes están debajo de un valor limite. Estos son invariablemente los elementos más lejanos desde las diagonales de los bloques del Jacobiano. Eliminándolos convierte cada bloque en una matriz de banda.

La matriz puede frecuentemente ser recortada severamente sin afectar la convergencia. Como veremos un cierto grado de imprecisión en el Jacobiano es frecuentemente tolerable. Un paso de Newton es, después de todo, solamente un aproximación del cero, y es esperado solamente decrecer la función de error. El Jacobiano necesita solamente ser lo suficientemente preciso de manera que el vector, ΔV actualizado, decremente la función de error.

ITERACIÓN SAMANSKII.

Es una observación empírica que, después de las primeras pocas iteraciones, y especialmente cerca de la convergencia, el Jacobiano no cambia mucho entre iteraciones. Nosotros podemos aprovechar este hecho simplemente reutilizando el Jacobiano por bastantes iteraciones consecutivas de Newton. Esta práctica trabaja bien, la llave es tener un método inteligente para decidir cuando el proceso se convierte en tan ineficiente que es lo mejor volver a calcular el Jacobiano. Si un Jacobiano es usado por mucho tiempo, el mejoramiento en F(V) se vuelve gradualmente más pequeño. Si es recalculado muy frecuentemente, la eficiencia sufre. Generalmente, el Jacobiano es recalculado cuando la norma de F(V) falla en ser reducida por alguna cantidad predeterminada.

La iteración de Samanskii debe ser usada con cuidado en un proceso de reducción de norma que utilice la norma NU. SI el Jacobiano es impreciso, la precisión de la norma sufre acordemente, y entonces puede volverse difícil decidir si un paso de reducción de norma resulta en una mejora en el error.

MÉTODOS DE CONTINUACIÓN.

Métodos de continuación evitan los problemas de convergencia al costo de un tiempo de computación mas elevado. En un método de continuación, Algunos parámetros de circuito son variados gradualmente, así la convergencia puede ser lograda en cada paso. En el primer paso, el parámetro es ajustado para hacer el circuito casi lineal y la convergencia es lograda fácilmente. La solución de este paso es usada como el estimado inicial para el siguiente paso. El parámetro es ajustado para hacer el circuito de alguna madera más fuertemente no lineal, y el proceso es repetido. El proceso continúa en esta manera hasta que la convergencia es lograda con el valor completo del parámetro del circuito.

Comúnmente el método de continuación mas usado es el incremento de la fuente. En este proceso, la magnitud de una fuente RF (o ocasionalmente una o mas fuentes dc) es el parámetro de continuación. En el proceso de continuación, la excitación es variada paso a paso desde un bajo nivel al nivel de excitación deseado, de tal manera que la convergencia es lograda a cada paso.

La cantidad de incremento, por paso, depende de la fuerza de la no linealidad del circuito.

Los métodos de continuación trabajan mejor cuando (1) es adaptativa (por Ej. Si la convergencia falla, el tamaño del paso es reducido y el proceso es repetido), y (2) Un estimado de la nueva solución mas que simplemente la solución del paso anterior, es usada como valor de comienzo para el siguiente paso.

La nueva solución es estimada como sigue. De (2.14) tenemos

$$\mathbf{F}(\mathbf{V}) = \mathbf{I}_s + \mathbf{Y}_{N\times N}\mathbf{V} + j\Omega\mathbf{Q} + \mathbf{I}_G = \mathbf{0} \tag{2.60}$$

Cuando el siguiente paso ha convergido.

$$\mathbf{I}_s = -(\mathbf{Y}_{N\times N}\mathbf{V} + j\Omega\mathbf{Q} + \mathbf{I}_G) \tag{2.61}$$

Diferenciado, tenemos

$$\frac{\partial \mathbf{I}_s}{\partial \mathbf{V}} = -\frac{\partial}{\partial \mathbf{V}}(\mathbf{Y}_{N\times N}\mathbf{V} + j\Omega\mathbf{Q} + \mathbf{I}_G) = -\mathbf{J}_\mathbf{F} \tag{2.62}$$

Ó

$$\frac{\partial \mathbf{V}}{\partial \mathbf{I}_s} = -\mathbf{J}_\mathbf{F}^{-1} \tag{2.63}$$

Entonces, el Jacobiano invertido puede ser usado al final de cada paso del método de continuación para estimar los voltajes de puerto del siguiente paso.

NORMA NU DE YEAGER Y DUTTON.

En un circuito lineal o casi lineal, cada paso de Newton puede reducir todos lo componentes de F(V). En realidad, como siempre, algunos componentes de F(V) decrementarán, mientras que otros pueden cambiar muy poco o incluso incrementar. ¿Como entonces podemos determinar cuando un paso de Newton es uno bueno? Esa determinación es esencial cuando los métodos de reducción de normas son usados.

Un posible método es calcular la norma Euclidiana de F (V), designada $|F(V)|$. Sucede, como siempre, que $|F(V)|$ es una opción pobre, debido a que la dirección del paso de Newton, en el espacio multidimensional, no necesariamente minimiza $|F(V)|$. La dirección del paso que minimiza $|F(V)|$ es su gradiente; sin embargo, nuestro paso de Newton es el gradiente de F(V) , no de su magnitud $|F(V)|$.

En un papel clásico, Yeager y Dutton demostraron que un paso de Newton, en general, no coincide con el gradiente de la sumatoria de $|F(V)|$, y puede ser incluso perpendicular a él. Ellos proponen, entonces una nueva norma, llamada la norma NU (o la norma de actualización de Newton), que es calculada por el Jacobiano y entonces tiene un gradiente que coincide con el paso de Newton. El uso de esta norma mejora el rendimiento de un simulador de balance armónico significativamente.

La norma NU, en la iteración *p*, es definida como

$$N_{NU} = \left|\mathbf{J}^{-1}(\mathbf{V}^p)\mathbf{F}(\mathbf{V})\right| \tag{2.64}$$

Donde $|*|$ indica la L_2, la norma (euclidiana), V^p es el vector de voltaje en la p-ava iteración, y $V = V^p - \beta\Delta V$; esto es, V es evaluada en el paso de reducción de la norma. El descenso más pronunciado en esta norma siempre coincide con la dirección del paso ΔV.

MODELOS PARAMETRICOS

Si el error multidimensional de la superficie puede ser "alisado" (por Ej. La no linealidad reducida), las características de convergencia del análisis de balance armónico pueden ser mejoradas. Rizzoli ha demostrado que esto puede ser hecho haciendo ambos las funciones de corriente y el voltaje un parámetro abstracto; Si el parámetro es x, nosotros formamos $i = f_i(x)$ y $v\, f_v(x)$, como un ejemplo, consideremos un diodo teniendo una relación corriente voltaje.

$$I = I_{sat}(\exp(\delta V) - 1) \tag{2.65}$$

La ecuación I/ V puede ser escrita en forma paramétrica:

$$v(t) = \begin{cases} V_1 + \frac{1}{\delta}\ln(1 + \delta(x(t) - V_1)) & x(t) > V_1 \\ x(t) & x(t) \le V_1 \end{cases}$$

$$i(t) = \begin{cases} I_s \exp(\delta V_1)(1 + \delta(x(t) - V_1)) - I_s & x(t) > V_1 \\ I_s(\exp(\delta x) - 1) & x(t) \le V_1 \end{cases} \tag{2.66}$$

V_1 puede ser seleccionado arbitrariamente, pero es mejor un número pequeño, típicamente 1/ δ. Cuando x (t) < V_1, x(t) = v(t) y (2.66) es idéntico a (2.65). A altos voltajes, sin embargo, *i(t)* se vuelve una función lineal de *x(t)*, y el voltaje del diodo, *v(t)*, se vuelve una función logarítmica de *x(t)*. Esto preserva la relación exponencial de la ecuación (2.65), mientras que dividiendo la fuerte no linealidad de (2.65) entre *v(t)* y *i(t)*, se vuelve efectivamente mucho más suave.

En este proceso, hemos reemplazado una variable dependiente *i(t),* con dos, *v(t)* y *i(t),* ambas las cuales son funciones de *x(t)*, la variable independiente. Esto hace el problema más grande, sin embargo la mejora en la convergencia puede compensar el incremento del tamaño del problema. La gran limitación de este método es que, los modelos estándar de la industria existente no están formulados en esta manera, y en muchos casos es difícil transformar estos modelos a esta forma. El simulador entonces debe ser formulado para trabajar óptimamente con ambos modelos paramétricos y no paramétricos, una complicación adicional.

FORMULACIÓN DE NODOS Y MAL CONDICIONAMIENTO EN LA MATRIZ Y.

Nosotros asumimos que la matriz Y su inversa ambas existen. En muchos casos, sin embargo particionar el circuito resulta en un subcircuito desconectado, que no tiene matriz de impedancia o admitancia. Si nada es hecho acerca de esta situación, el análisis seguramente fallara.

Una solución simple es reemplazar cada rama no lineal con un resistor de moderado valor, una resistencia de 100 Ω usualmente trabaja bien. Esta resistencia previenen que la matriz Y se desconecte, así una matriz singular Y es mucho menos probable que ocurra. Las resistencias pueden ser removidas de la matriz Y de N puertos simplemente substrayendo sus conductancias de la diagonal principal.

Otra solución es usar una formulación nodal en lugar de una formulación a base de puertos, que en este caso, los voltajes de nodo y no las ramas de voltaje de los elementos no lineales, se vuelven las cantidades independientes. Una formulación nodal puede ser más tolerante de desconexiones en el circuito, aunque nodos aislados pueden aun resultar en una matriz singular.

En una formulación nodal, cada voltaje de nodo en el circuito se vuelve una variable independiente. El número de variables se mantiene 2 N (K +1), pero N es el número de nodos no el número de voltajes de control. En circuitos de RF y microonda, el número de nodos, es más probable que sea mayor que el número de elementos no lineales, así que la formulación nodal incrementa el tamaño del Jacobiano. En circuitos con ICs análogos, sin embargo, el numero de no linealidades puede estar en el mismo orden que el numero de nodos, así que las desventajas pueden ser menores o incluso inexistentes. Cuando una formulación nodal es usada en el análisis de ICs o microonda o RF, grandes partes pueden frecuentemente ser reducidas a bloques de nodos más pequeños, reduciendo el tamaño del problema.

CRITERIO DE TERMINACIÓN.

En el análisis de balance armónico basado en el método de Newton, nosotros decrementamos el vector de error F (V) hasta que los errores son insignificantes. Definiendo, precisamente, lo que se puede considerar *insignificante* es a veces un dilema. El problema surge del hecho que los componentes de corriente en varios puertos y en varios armónicos pueden variar ampliamente en magnitud; Un factor de 10^6 o incluso 10^8 de diferencia entre los componentes más pequeños y los más grandes no es inusual.
Para ilustrar las dificultades, examinaremos unas pocas posibilidades.

LIMITAR LA NORMA EUCLIDIANA.

Una posibilidad es requerir que la norma Euclidiana de la función de error sea menor que aun valor máximo. Matemáticamente, nosotros requerimos que

$$|\mathbf{F}(\mathbf{V})| < \varepsilon \tag{2.67}$$

Donde ε es un valor escalar. En este caso, los componentes grandes, de corriente del circuito los cuales normalmente tienen grandes errores, dominan en establecer la magnitud del error; los componentes pequeños contribuyen poco. Entonces incluso cuando la magnitud del error es pequeña, los errores de los componentes de poca corriente pueden ser bastante grandes. Los errores en las corrientes menores pueden ser controlados requiriendo que la magnitud del error sea mas pequeña, pero esto pone demandas irrealistas en los errores en los componentes grandes. Entonces, los máximos errores permisibles para los componentes de gran corriente pueden ser tan pequeños que la convergencia sea imposible.

LIMITAR LAS MAGNITUDES ABSOLUTAS DE LOS COMPONENTES DE CORRIENTE.

Otra posibilidad es requerir que las magnitudes absolutas de todos los componentes de F(V) sean menores que algún valor máximo; Específicamente,

$$F_{n,k} < \varepsilon \quad \text{Para toda n, k} \tag{2.68}$$

Como en el criterio anterior, un valor de ε que es adecuado para garantizar la precisión de los componentes de pequeña corriente pueden ser demasiado restringido para los grandes.

LIMITAR LAS MAGNITUDES RELATIVAS DE LOS COMPONENTES DE CORRIENTE.

Una posible solución es limitar las magnitudes relativas de los componentes de error de la corriente en lugar de los valores absolutos. Específicamente, nosotros requerimos que

$$2\left|\frac{(I_{n,k}+\hat{I}_{n,k})}{I_{n,k}-\hat{I}_{n,k}}\right| < \varepsilon$$

Para toda *n, k* (2.69)

Esto es, nosotros comparamos el error de la corriente $|I_{n,k}+\hat{I}_{n,k}|$, al valor de corriente absoluto, definido, como el promedio de la corriente en los sub -circuitos lineales y no lineales, $|\hat{I}_{n,k} - I_{n,k}| / 2$, aunque una mejora sobre los criterios anteriores, es que este criterio tiene el problema opuesto; un valor razonable de error para los componentes de gran cantidad de corriente hace irrazonables demandas en la convergencia de los componentes pequeños. Por ejemplo, en un amplificador de potencia, nosotros bien podríamos querer que en la frecuencia fundamental, el error sea menos que 1%, pero un 1% de error es demasiado severo para los componentes de intermodulación débil, cuya precisión es del orden de unos pocos decibles cuando mucho.

Otro problema es que (2.69) tiene significado solo cerca de la convergencia cuando el proceso iterativo esta lejos de converger, los errores relativos

$I_{n,k+} I_{n,k-} I_{n,k-} I_{n,k}$ se mantienen estoicamente en un valor de 2. Esto no afecta las iteraciones de Newton, pero no le da al usuario ninguna información acerca del progreso de la convergencia, así el no puede saber si el problema esta procediendo normalmente hacia la convergencia.

COMBINANDO CRITERIOS ABSOLUTOS Y RELATIVOS.

Una solución final es combinar los criterios absolutos y relativos. En este esquema, cada componente de error de la corriente $|I_{n,k}+\hat{I}_{n,k}|$, es observado. Si satisface ambos el criterio de error de valor absoluto y relativo, el componente es considerado que ha convergido. El análisis termina cuando los componentes convergen en este sentido. Claro, que el vector de error entero debe ser examinado en cada prueba de convergencia.

Este esquema naturalmente acomoda ambos componentes de gran y pequeña corriente. Los componentes de gran corriente usualmente convergen en la base del error relativo, y los pequeños en el error fraccional, así el criterio relativo es mas débil para componentes grandes y el criterio absoluto es mas débil para componentes pequeños. Es simplemente cuestión de seleccionar el criterio así asegurar que todos los errores son suficientemente pequeños al terminar.

ESTIMADO INICIAL.

Una propiedad importante del método de Newton es que la velocidad y confiabilidad de la convergencia depende fuertemente del estimado inicial del vector de solución. Formular un estimado inicial puede no ser difícil al analizar un tipo específico de circuito, pero puede ser difícil de concebir una manera de formar estimados iniciales en un programa de análisis de circuito de propósito general, el cual debe acomodar una gran variedad de circuitos que tienen una gran variedad de posibles respuestas.

Para circuitos casi lineales, como amplificadores de potencia de clase-A la respuesta lineal es un buen estimado inicial. La respuesta puede ser encontrada al poner el nivel de excitación a un valor pequeño y el número de armónicos, K, igual a uno, así el tamaño del problema es relativamente pequeño. Cuando la solución ha sido completada, los resultados son escalados al correcto nivel de excitación y K es reseteado al valor deseado para un análisis de gran señal.

En circuitos fuertemente no lineales como amplificadores de clase-B o C, multiplicadores de frecuencia, o mezcladores, un estimado inicial es mas difícil de generar. Ocasionalmente la naturaleza del circuito nos permite un buen estimado; por ejemplo, en los mezcladores de diodo, la forma de onda del voltaje del diodo invariablemente es una sinusoide acortada. En casos difíciles, es mejor hacer primero un análisis de dc, entonces aplicar la señal de RF y incrementarla usando algún método de continuación.

3. DISEÑO Y CONSTRUCCION DE INTERFACES DE USUARIO

3.1. ANALISIS COMPARATIVO DE HERRAMIENTAS DE DESARROLLO

Existe un amplio conjunto de lenguajes de programación entre los cuales escoger al realizar una aplicación, entre los cuales podemos destacar:

FoxPro

Visual FoxPro es un lenguaje con especialización en aplicaciones de Base de Datos. Posee un potente entorno orientado a objetos para la creación de bases de datos y la programación de aplicaciones. Visual FoxPro proporciona todas las herramientas necesarias para administrar datos, tanto si va a organizar tablas de información y ejecutar consultas, como si va a crear un sistema de base de datos relacional integrado o programar una aplicación para la administración de datos de usuarios.

Matlab

Matlab (Matrix Laboratory) es un lenguaje con especialización en aplicaciones matemáticas y científicas. Posee un poderoso compilador que genera aplicaciones rápidas y eficientes y librerías de C\C++ que permiten que sus características sean utilizadas desde estos lenguajes para mayor portabilidad.

Java

Java es un lenguaje de programación orientado a objetos, basado en clases concurrentes y de propósito general. Esta específicamente diseñado para tener tan pocas dependencias con la plataforma en la que será implementado como sea posible. Java permite a los desarrolladores de aplicaciones escribir un programa una vez y ser capaces de correrlo en cualquier plataforma de hardware o sistema operativo. Esta diseñado para ser lo suficientemente simple para que los programadores puedan alcanzar fluencia en el lenguaje con relativa facilidad. Java esta íntimamente relacionado con C y C++ pero esta organizado de una manera completamente diferente con un pequeño numero de aspectos de C y C++ omitidos y una pocas ideas de otros lenguajes incluidos.

Visual Basic .NET

Visual Basic, es un lenguaje que permite de una manera rápida y sencilla crear aplicaciones para Microsoft Windows®. Tanto si es un profesional experimentado como un recién llegado a la programación en Windows, Visual Basic le proporciona un juego completo de herramientas que le facilitan el desarrollo rápido de aplicaciones.

¿Qué es Visual Basic? La palabra "Visual" hace referencia al método que se utiliza para crear la interfaz gráfica de usuario (GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los elementos de la interfaz, simplemente puede agregar objetos prefabricados en su lugar dentro de la pantalla. Si ha utilizado alguna vez un programa de dibujo como Paint, ya tiene la mayor parte de las habilidades necesarias para crear una interfaz de usuario efectiva.

La palabra "Basic" hace referencia al lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code), un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática o computación. Visual Basic ha evolucionado a partir del lenguaje BASIC original y ahora contiene centenares de instrucciones, funciones y palabras clave, para realizar infinidad de tareas. Los principiantes pueden crear aplicaciones útiles con sólo aprender unas pocas palabras clave, pero, al mismo tiempo, la eficacia del lenguaje permite a los profesionales acometer cualquier objetivo que pueda alcanzarse mediante cualquier otro lenguaje de programación de Windows.

Delphi

Object Pascal es un lenguaje de alto nivel, compilado, con un fuerte control de sintaxis que soporta diseño de programación estructurada y orientada a objetos. Sus beneficios incluyen código fácil de leer, compilación rápida y el uso de múltiples archivos para la programación modular.

Tiene características especiales que soportan el ambiente de trabajo RAD y la librería de componentes de Borland. Es utilizado en herramientas de desarrollo de Borland como lo es Delphi.

C/C++

El lenguaje C es un lenguaje de propósito general conocido por su eficiencia, economía y portabilidad. Mientras que estas características lo hacen una buena opción para cualquier tipo de programación, C tiene una amplia trayectoria de ser especialmente útil en la programación de sistemas debido a que facilita escribir programas compactos y eficientes que son rápidamente adaptados a otras plataformas. Programas C bien escritos son frecuentemente tan rápidos como programas en lenguaje ensamblador, y son típicamente más fáciles para los programadores para leer y mantener.

En la tabla comparativa a continuación se comparan los diferentes lenguajes en cuanto a las características más importantes con respecto a su uso en este proyecto en particular.

Análisis Comparativo	FoxPro	Visual Basic	Visual .NET	Delphi	Matlab	Java	C/C++
Soporte de Análisis Numérico	●○○○○	●●●○○	●●●○○	●●●○○	●●●●●	●●●○○	●●●○○
RAD (Rapid Application Deployment)	●●●●○	●●●●●	●●●●●	●●●●●	●●●○○	●●●○○	●○○○○
Obsolescencia (End of Life)	●○○○○	●○○○○	●●●●○	●●●○○	●●●●○	●●●●○	●●●●○
Soporte Multiplataforma	●○○○○	●○○○○	●○○○○	●●●○○	●●●○○	●●●●●	●●●●○
Eficiencia de Compilación	●●●○○	●●●○○	●●●○○	●●●●●	●●●●○	●●●○○	●●●●●
Compatibilidad a nivel de Fuentes.	●○○○○	●●○○○	●●●○○	●●●○○	●●○○○	●●○○○	●●●●●
Totales	11	15	19	22	21	20	22

Las características que se compararon entre los diferentes lenguajes fueron los siguientes:

Soporte de Análisis Numérico: La capacidad que los diferentes lenguajes tienen en lo que se conoce como "data crunching" o el manejo de grandes volúmenes de datos numéricos para su procesamiento, se tomo como ejemplo modelo "matlab" por su capacidad sobresaliente en manejar números.

RAD (Rapid Application Deployment): La capacidad que tiene una herramienta de desarrollo de facilitar la creación rápida de interfaces de usuario y de aplicaciones en su totalidad, se tomo como ejemplo modelo "Visual Basic" como la herramienta por excelencia para la creación rápida de aplicaciones de una manera fácil e intuitiva.

Obsolescencia (End of Life): Una de las situaciones más preocupantes en el área de la informática es el rápido avance del campo que permite que una herramienta de software se vuelve obsoleta al cabo de unos cuantos años. Es como que un documento escrito hoy en unos cuantos años fuera visto como jeroglíficos egipcios o precolombinos en una piedra roseta escritos en una lengua muerta. Se trata de asesorar la capacidad que tendrá una persona de encontrar una herramienta de desarrollo equivalente a la que se uso en este proyecto en el futuro y su capacidad de "mantener" el proyecto, pudiendo generar un ejecutable moderno actualizado sin necesidad de reescribir todo el código. Se tomo como ejemplo modelo "Visual FoxPro" como una herramienta que aunque se encuentra en su versión 9.0 (año 2005), desde hace varios años; Microsoft, el principal fabricante de herramientas de este tipo anunció que descontinúa el desarrollo de esta en favor de las tecnologías ".NET". Esta es una característica que se ha medido de manera inversa, es decir, entre menos obsoleto, mejor. Sin embargo, una tecnología excesivamente nueva, puede volverse obsoleta rápidamente por falta de aceptación del mercado, así lo que en realidad se busca es estabilidad.

Soporte Multiplataforma: Se evaluó la capacidad del lenguaje de ser compilado en diferentes plataformas de hardware y sistemas operativos. Se tomó como ejemplo modelo "Java" por ser un lenguaje diseñado con ese objetivo como característica fundamental.

Eficiencia de Compilación: Se evaluó la capacidad de un lenguaje de generar código que sea tan rápido como programas en lenguaje ensamblador. Se tomó como modelo "C/C++" como un lenguaje con un record sobresaliente en esta área.

Compatibilidad a nivel de Fuentes: Se evaluó la capacidad que tiene un lenguaje de compartir código escrito en otros lenguajes diferentes y para diferentes plataformas en un solo proyecto. Es decir, la capacidad que se tiene por ejemplo de utilizar controles "ActiveX", "OLE", utilizar "APIs", "comunicación con DLLs", compatibilidad con librerías objeto estáticas compiladas en otros lenguajes y capacidad de generar ejecutables para diferentes plataformas a partir de código unificado. Se tomó nuevamente como modelo a "C\C++" por su excelencia en esta área.

Después de haber realizado el correspondiente estudio cuyos resultados pueden verse resumidos en la tabla anterior se escogió utilizar "Borland Delphi" para realizar la interfaz gráfica, y "C\C++" para el núcleo del simulador. Ambos alcanzaron una puntuación de 22 puntos en el estudio realizado.

3.2. DISEÑO Y CONSTRUCCION DE INTERFACES DE USUARIO

CONSTRUCCIÓN DE INTERFACES DE USUARIO[*1]

Las característica básica de una interfaz gráfica de usuario es la integración de un número de elementos que proporcionan soluciones para realizar tareas y trabajos según se requiera, y son los siguientes:

1. Se tiene un mapa de bits y este se muestra en alta resolución[*2].
2. Se tiene un dispositivo apuntador, usualmente un mouse.
3. Se promueve la consistencia de interfaces entre programas.
4. Los usuarios pueden ver gráficos y texto en la pantalla y pueden imprimirla así como la observan.
5. Se sigue el paradigma de interacción objeto-acción.
6. Se permite transferir información entre programas.
7. Puede haber manipulación directa de información sobre la pantalla y objetos.
8. Se proporcionan elementos de interfaz estándares como menús y cajas de diálogo.
9. Existe visualización de información y de objetos (íconos y ventanas).
10. Se provee de una retroalimentación visual para tareas y acciones de los usuarios.
11. Existe una muestra visual de las acciones y modos usuario/sistema (menús y paletas).
12. Se permite el uso de controles gráficos ("widgets"), para selecciones y entradas del usuario.
13. Se permite a los usuarios personalizar las interfaces y las interacciones.
14. Se permite flexibilidad entre el teclado y otros dispositivos de entrada.

Finalmente se elige la interfaz gráfica apropiada para el sistema operativo y los programas que se utilizan, basándose en cómo se clasifican de acuerdo a varios criterios, las prioridades de las características anteriormente listadas, para el software en que se emplean.

[*1]La descripción del proceso de construcción de una interfaz de usuario fue tomada de [4].

[*2]En las interfaces de usuario se encuentran dos tipos de mapas de bits, el primero, en el que están representados todos los objetos en la interfaz, este se encuentra almacenado en el buffer de video, y el segundo, el que representa el área de trabajo de la aplicación, el cual es manipulado directamente en el caso de un programa de dibujo o indirectamente a través de funciones proporcionadas por el sistema operativo u objetos de encapsulación("widgets") en caso de que represente texto ú otro tipo de información simbólica soportada por el sistema.

DISEÑO DE INTERFACES DE USUARIO[*]

Cuando se va a diseñar una plataforma Windows, el diseño de la interfaz puede describirse en tres fases: Diseño, Prototipo y Pruebas.

La fase de diseño progresa a través de un número bien definido de pasos que siguen una secuencia.

1. Definir las metas de usabilidad del producto y los objetivos

   Existen cuatro áreas a tomar en cuenta en este paso: Utilidad, Eficacia, Habilidad de Aprendizaje y Actitud.

2. Desarrollar escenarios y tareas para el usuario.

3. Definir objetos y acciones de la Interfaz.

   a. Derivar objetos, datos y acciones del usuario y llevarlo como sea posible a un escenario de tareas.

   b. Revisar y redefinir objetos y acciones listadas por el usuario

   c. Dibujar un objeto en un diagrama de Relaciones.

   d. Crear una matriz para lograr una manipulación directa del objeto.

4. Determinar objetos como íconos, vistas y representaciones visuales.

5. Diseño de objetos y ventanas de menús.

6. Refinar el diseño visual.

La fase de diseño culmina con la fase de construcción de un prototipo de la Interfaz. Luego se realizan las pruebas para verificar que el diseño cumple con los requisitos de comportamiento, desempeño y satisfacción del usuario.

[*]La descripción del proceso de diseño de una interfaz de usuario fue tomada de [4].

PARTE III: PROTOTIPO DE UN SOFTWARE DE SIMULACION DE CIRCUITOS UTILIZANDO EL METODO DEL BALANCE ARMONICO

4. DISEÑO DE LA INTERFAZ DE USUARIO

4.1. HERRAMIENTAS DE DISEÑO Y RECURSOS EMPLEADOS

Herramienta de desarrollo utilizada para la creación de la Interfaz de Usuario.

Para cumplir con los objetivos se crea una interfaz de usuario del programa en Borland Delphi 6.0.

Borland Delphi es una herramienta RAD (Rapid Application Development) basada en OOP (Object Oriented Pascal). Permite el desarrollo rápido de aplicaciones con el poder de un lenguaje orientado a objetos, haciéndolo una excelente elección para desarrollar la interfaz gráfica. La arquitectura de software de la interfaz se describe empleando la librería de componentes cross-platform de Borland CLX, que permite el desarrollo de aplicaciones en Windows y Linux; ésta se desarrolló siguiendo las fases principales del proceso de diseño de una interfaz de usuario tal como se describen en el marco teórico.

Recursos de hardware empleados.

Computadora AMD Athlon 64 2800+, con sistema operativo Windows Vista, con 512MB de memoria RAM y un disco duro de 400GB

Plataforma de 32 bits.

4.2. DESCRIPCIÓN DEL DISEÑO POR FASES

FASE I: FASE DE DISEÑO.

Definición de las metas de usabilidad del producto y los objetivos.

- ✓ Los usuarios esperan poder dibujar y simular circuitos, que la interfaz sea intuitiva, y de fácil uso.

Definición de objetos y acciones de la interfaz.

El usuario espera ver en pantalla símbolos de los diferentes dispositivos eléctricos y electrónicos, con los que pueda interactuar a fin de dibujar un diagrama esquemático de un circuito a simular, pudiendo mover estos objetos alrededor del área de dibujo, rotarlos y colocarlos de manera arbitraria, dibujar uniones y conexiones entre los mismos, establecer sus propiedades eléctricas, nombres designados y otras propiedades, a fin de establecer todas las características del circuito necesarias para hacer una simulación de su funcionamiento.

Determinar objetos como iconos, vistas y representaciones visuales.

En la figura se muestran los iconos que representan la selección de los diferentes dispositivos, además de los iconos que representan las diferentes acciones que se pueden realizar en la barra de herramientas de la aplicación.

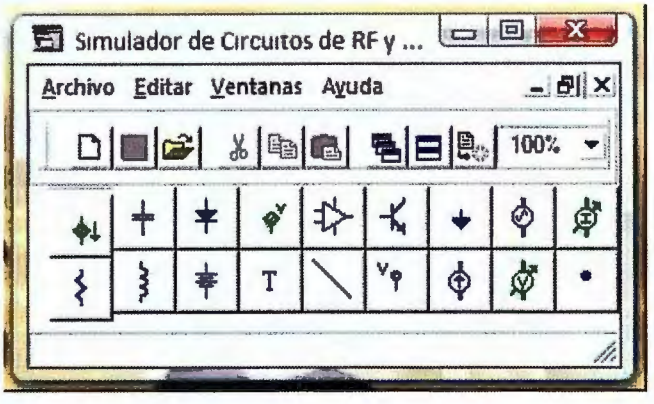


Fig. 6 - Iconos dibujados en la computadora para la representación de los diferentes dispositivos eléctricos y electrónicos.

Diseño de objetos y ventana de menús.

Para el diseño de los menús dentro de la ventana donde se visualizara el área de dibujo esquemático se utilizaran las selecciones que se detallan en la siguiente tabla.

Objeto Menú	Menú Desplegable	Propósito
Archivo	Nuevo	Crear un nuevo archivo.
	Abrir	Abrir un archivo.
	Cerrar	Cerrar el archivo actual.
	Guardar	Guardar el archivo actual.
	Guardar Como	Guardar el archivo actual con un nombre diferente.
	Salir	Salir de la aplicación.
Editar	Deshacer	Deshacer la última acción realizada.
	Repetir	Repetir la última acción realizada.
	Cortar	Corta el último elemento seleccionado.
	Copiar	Copia el último elemento seleccionado.
	Pegar	Pega el contenido del portapapeles.
	Borrar	Borrar el último elemento seleccionado.
Ventanas	Diagrama de árbol	Muestra la estructura de objetos.
	Capas	Muestra las diferentes capas del esquemático.
	Mosaico	Ordena las ventanas activas en mosaico.
	Minimizar Todas	Minimiza todas las ventanas.
Ayuda	Acerca de...	Despliega información del programa, número de versión y copyright.

RESULTADOS DE LA FASE DE DISEÑO.

Se describen las características básicas que se deben encontrar en una interfaz gráfica de un sistema para el dibujo de esquemáticos, entre las que se pueden mencionar:

Manejo vectorial de gráficos.

Existen dos tipos de programas de dibujo: los que utilizan gráficos de píxeles y los que utilizan gráficos vectoriales. Los gráficos vectoriales (también conocidos como modelados geométricos o gráficos orientados a objetos) son los que se conforman con primitivas geométricas tales como puntos, líneas, curvas o polígonos.

Todos los ordenadores actuales traducen los gráficos vectoriales a gráficos rasterizados para poderlos visualizar en pantalla. La imagen rasterizada posee un valor determinada para cada píxel que la conforma, esta información se guarda en la memoria ocupando un espacio específico.

Los gráficos vectoriales son ideales para dibujos simples y compuestos que necesitan tener formas independientes o que no necesitan tener un carácter de realismo fotográfico, lo que los hace la opción ideal en un programa para el diseño de dibujos esquemáticos.

Se deben dibujar cada uno de los dispositivos.

En los gráficos vectoriales la imagen se genera como descripción de trazos, por ejemplo: para crear un segmento de línea recta se indica su punto inicial (x1,y1), su punto final (x2,y2), su grosor, color, etc. En cambio, en una imagen bitmap, esa misma línea estaría formada por un número determinado de puntos (píxeles) de color contiguos.

Rotar, voltear, mover, capacidades de zoom.

Al contrario que un bitmap, una imagen vectorial puede ser escalada, rotada o deformada, sin que ello perjudique en su calidad. Normalmente, un conjunto de trazos se puede agrupar, formando objetos, y crear formas más complejas.

Los típicos graficadores vectoriales permiten rotar, mover, reflejar, estirar, inclinar, realizar finas transformaciones de los objetos, cambiar en orden en el eje z (eje que define la dimensión de profundidad en 3D) y combinar objetos primarios para componer objetos más complejos

Cortar, copiar, uso del portapapeles, rehacer, deshacer.

El portapapeles es un área de memoria, definida por el sistema operativo, donde se almacenan los datos temporalmente a fin de que puedan ser copiados en otro lugar. Todo tipo de aplicaciones usan el portapapeles mediante las operaciones «copiar y pegar (copy and paste)» o «cortar y pegar (cut and paste)»

Características proyectadas en la aplicación y componentes de software empleados.

Se utilizó de plataforma el componente CommonCAD cuyo desarrollo ha sido dirigido por Pal Sitkei, en su versión 0.129. El componente CommonCAD es el resultado de una serie de artículos publicados en el foro de discusión de programación Húngaro "prog.hu" que se titulan "Desarrollando un programa CAD en 24 horas". El componente Delphi es un proyecto de código fuente abierto y su uso es libre para cualquier propósito.

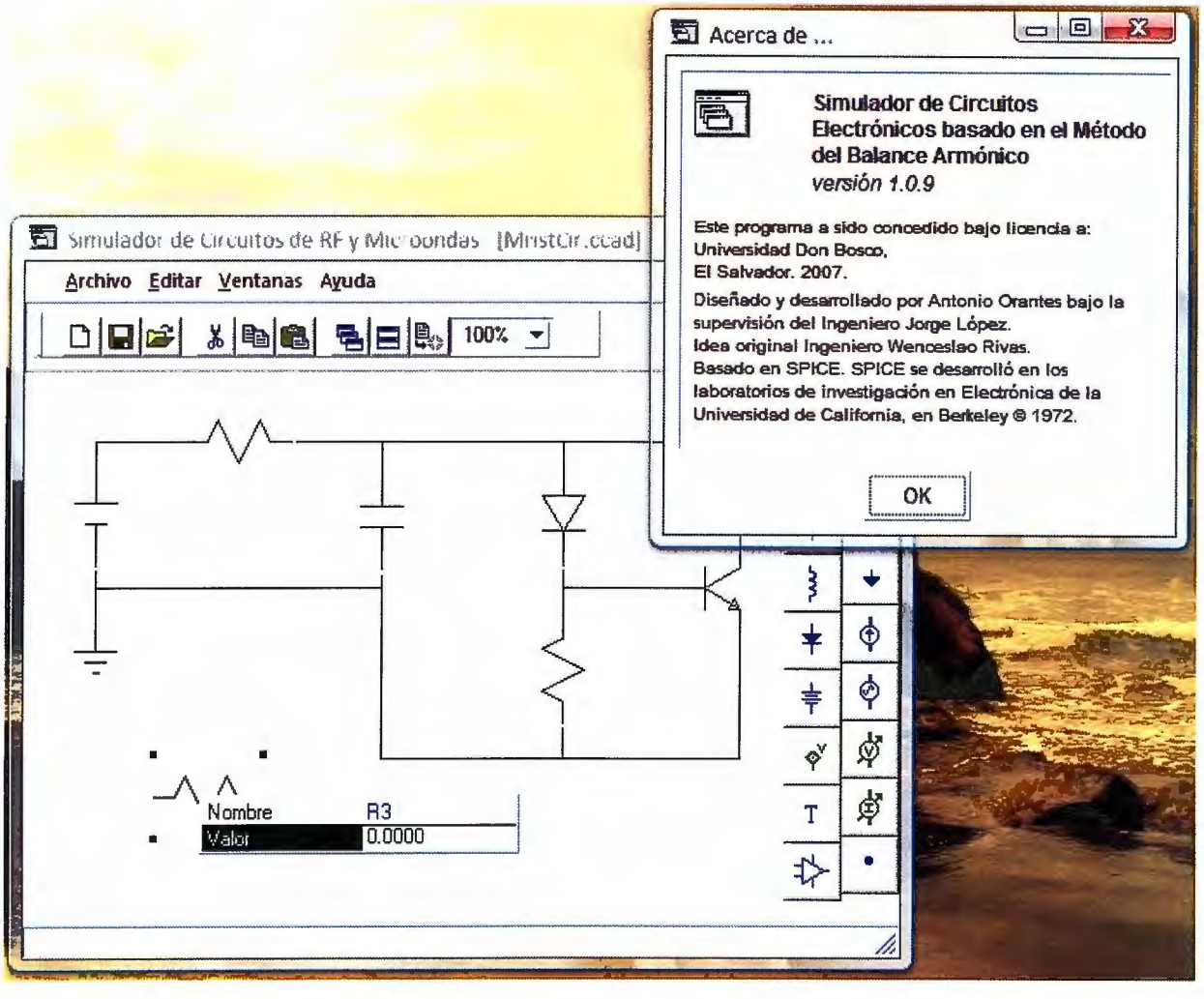
El sistema completo manejará:

- Dibujo.
- Nombrado de objetos, vista de estructura de árbol.
- Reconocimiento de objetos.
- Ampliado (Zoom) , Desplazamiento.
- Selección de Objetos, Puntos de anclaje (Grip points).
- Relaciones jerárquicas entre los objetos.
- Selección múltiple.
- Algunos dispositivos prácticos: resistencias, capacitancias, inductancias, fuentes dependientes e independientes, cableado, nodos.
- Propiedades de los componentes.
- Formato de almacenado y recuperación de archivos.
- Capas (Layers)
- Objetos vinculados, Detección de unión de objetos, manejo de las uniones entre objetos (nodos).
- Funciones básicas: copiar, voltear, mover, rotar, escalar.
- Funciones del portapapeles: copiar y pegar.
- Deshacer y rehacer
- Comunicación con el motor de simulación.

FASE II: FASE DE CREACIÓN DEL PROTOTIPO

Construcción del prototipo de la interfaz

En la figura se muestra la idea del prototipo, en la que se refleja la funcionalidad del modelo.



RESULTADOS DE LA FASE DE PROTOTIPO.

Para la codificación de esta aplicación se definieron o reutilizaron las siguientes clases:

tForm – La clase tForm representa una ventana de aplicación estándar(Formulario), todas las ventanas son implementadas como descendientes de la clase tForm, esto incluye la ventana de la aplicación principal, cajas de dialogo y ventanas hijas de una aplicación MDI(Interfaz de múltiples documentos). Un formulario puede contener otros objetos de control como botones, cajas de texto, etc.

tMainForm – Representa la ventana principal de la aplicación, engloba todos los elementos en un todo coherente, combina menús, botones, íconos y líneas de estado con un objeto tDraw con un área de cliente adecuada, que a su vez, permite la creación e inserción de cualquier cantidad de objetos para el dibujo para el desarrollo de un esquemático.

tMDIChild – Es la clase base sobre la cual se derivan todos las ventanas de los esquemáticos, por razones propias del diseño de la CLX, para permitir ser insertados y gestionados dentro del área cliente de una aplicación MDI.

tComponent – Esta es la clase común de la que descienden todos los objetos que pueden aparecer en un formulario, constituye también parte del diseño de la CLX.

tBasic – Esta es la clase común de la que descienden todos los objetos de dibujo de la aplicación, descendiente de tComponent implementa las capacidades básicas requeridas para un dibujo vectorial.

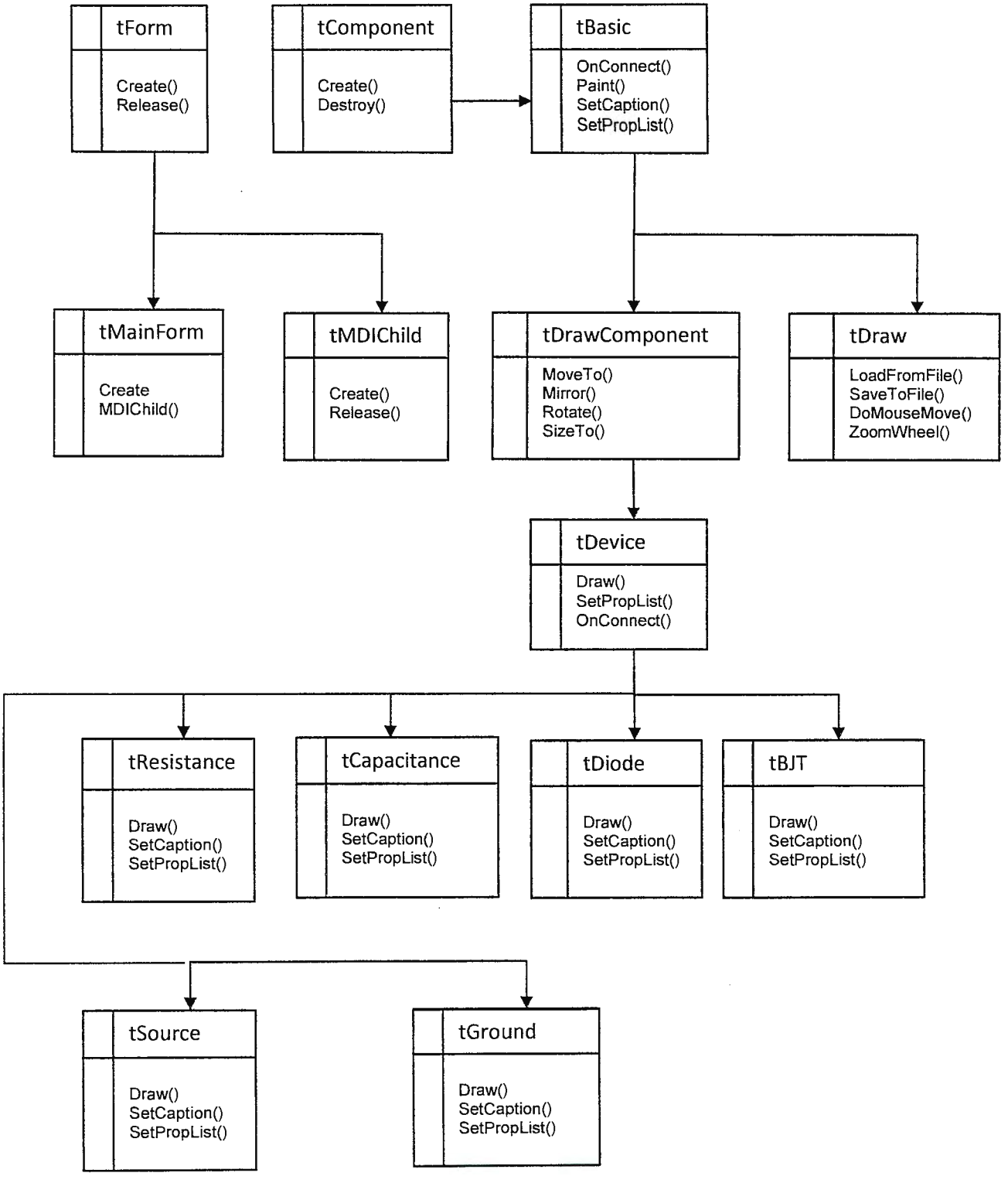
tDraw – Esta es la clase que representa el dibujo esquemático actual, poseedora de un objeto Canvas, provee un espacio abstracto de dibujo para los otros objetos que deben renderizar sus propias imágenes.

tDrawComponent – Clase común que representa a todos los objetos de dibujo que pueden aparecer en el Canvas provisto por tDraw.

tDevice – Clase que representa todos los objetos de dispositivo, que se dibujan en el esquemático, brinda la base para los requerimientos de dibujo, propiedades de dispositivo, manejo de uniones con otros dispositivos, que esta clase de objetos requieren.

tResistance, tCapacitance, tDiode, tSource, tGround, tBJT – Descendientes de tDevice, representan las características únicas de cada uno de los dispositivos mencionados, modelos matemáticos y formas únicas de comunicación del motor de simulación; Representan dispositivos a los que se les dedico un manejo especial en la fase de diseño del simulador. Los elementos o dispositivos no considerados en la fase de diseño, serán manejados directamente desde la clase tDevice o una descendiente diseñada para tal fin, tGeneralDevice, aunque carezcan de algunas propiedades únicas de los que fueron desarrollados en la fase de diseño.

Diagramas de Clases.



5. DISEÑO DEL MOTOR DE SIMULACION

5.1. HERRAMIENTAS DE DISEÑO Y RECURSOS EMPLEADOS

Herramienta de desarrollo utilizada para la creación del motor de simulación.

Para cumplir con los objetivos se desarrolla el núcleo del motor de simulación del programa en Microsoft Visual C++ 2005.

Visual C++ provee un ambiente de desarrollo flexible y poderoso para crear aplicaciones Windows, permitiendo compilar las aplicaciones en las plataformas de 32 y 64 bits. C++ es el lenguaje más popular del mundo para desarrollar aplicaciones al nivel de sistema y Visual C++ brinda al desarrollador una herramienta profesional de clase mundial para desarrollar aplicaciones. La arquitectura del núcleo del motor interfaz se describe empleando la librerías en tiempo de ejecución de C (CRT), permitiendo el desarrollo de una aplicación de gran portabilidad que se podrá compilar en Windows y a su vez, en Linux utilizando el compilador GNU de C (gcc); ésta se desarrolló siguiendo las fases de diseño que se describen a continuación.

Recursos de hardware empleados.

Computadora AMD Athlon 64 2800+, con sistema operativo Windows Vista, con 512MB de memoria RAM y un disco duro de 400GB

Plataforma de 32 bits.

5.2. ELECCIONES DE DISEÑO.

Hay muchas elecciones que deben ser hechas en el desarrollo de un programa como este. Algunas de estas selecciones son obvias, como el intercambio entre espacio de memoria por velocidad al pre-computar y guardar muchos valores intermedios, mientras que otras son mucho menos obvias como el intercambio de rendimiento por simplicidad al programar. En cada caso, hay muchas posibles soluciones de las cuales escoger pero en este programa se escogió un pequeño número de ellas para presentarlas al usuario, con cada opción adicional presentada al usuario produciendo un nivel extra de complejidad en el programa. Porque este simulador de circuitos es de propósito general y debe funcionar bien con una amplia gama de circuitos, de diferentes tamaños y con diferentes opciones de análisis, las elecciones de diseño realizadas también deben producir una solución robusta. Seleccionar el conjunto correcto de decisiones de diseño puede ser tan importante al hacer un sistema como este simulador funcional como escoger correctamente los algoritmos. En general, las elecciones han sido hechas de manera conservadora con una opción de relajar los criterios elegidos cuando un beneficio significativo puede ser derivado de ello.

5.3. SIMULADOR DE PROPÓSITO GENERAL.

Este programa ha sido diseñado como un simulador de propósito general, por lo que se ha intentado producir una solución aceptable a cualquiera de una amplia variedad de circuitos dados para resolver. Es posible, en algún caso, implementar un programa que pueda mejorar el rendimiento del actual para una clase de problemas específicos tomando ventajas de conocimiento especial del circuito o de la tecnología usada, así como que reduzca el tiempo de simulación sin una reducción significante en la precisión de la simulación, usando técnicas especiales que no son usadas en este simulador en general debido al deseo de que maneje una amplia gama de circuitos y que trabaje bien en un amplio rango de plataformas de hardware. De hecho, como se mencionó previamente, si una versión de este programa se diseñara optimizada para una máquina particular o un sistema operativo podría esperarse mejoras significativas en el rendimiento sobre la versión de propósito general que ha sido desarrollada en este proyecto.

Finalmente, muchas de las decisiones de diseño consideradas en la programación del sistema que se aplicaron a este simulador pueden ser aplicados a otra variedad de simuladores, el hecho de ser basado en SPICE permite que el usuario avanzado del simulador puedan encontrar características de interfase estandarizadas desde el núcleo del simulador al paquete de interfaz de usuario útiles también como permite que varios simuladores presenten una interfase uniforme al usuario, haciendo fácil para el usuario seleccionar la simulación correcta para el trabajo sin preocuparse acerca de nuevos lenguajes, nuevas descripciones de circuitos, o nuevas maneras de producir salida.

5.4. FILOSOFÍA DE CAJA DE HERRAMIENTAS.

Este sistema ha sido desarrollado usando un acercamiento de caja de herramientas, cada módulo de rutinas es relativamente independiente de cualquier otro módulo, permitiendo mantener y desarrollar el programa, seleccionando rutinas que mejor se adecuan a la tarea de un amplio rango de opciones disponibles. Para uso general, una versión del programa puede ser ensamblada dándole al usuario la mayoría de las características del programa. En un ambiente de desarrollo donde el espacio es una característica apremiante, una opción más restringida puede ser ensamblada dejando de lado rutinas que pueden no ser necesitadas, inclusive módulos enteros de análisis y tipo de dispositivos. Cuando se requiera añadir capacidades adicionales a este sistema, esta variedad de rutinas minimiza el código que debe ser escrito desde cero permitiendo que varias rutinas preexistentes puedan ser rehusadas.

Al seguir este acercamiento desde el principio, Este sistema incluye rutinas que representan funciones que ya no son necesarias, por haber sido reemplazadas por otras rutinas con una función ligeramente diferente o funciones escritas totalmente para ser utilizadas en la etapa de desarrollo. Estas rutinas permanecen como una parte de las librerías y pueden ser usadas cuando un programador desee modificar el código y

precise de una herramienta para el desarrollo o testeo de cualquier parte de este programa. Estas rutinas también proveen una guía para que se creen rutinas similares de ser necesarias. De hecho, estas herramientas y rutinas son excluidas del programa durante su uso en producción así la utilización de ambos de memoria y tiempo de ejecución son minimizadas.

5.5. IDENTIFICACIÓN Y AISLAMIENTO DE LOS MÓDULOS.

Al descomponer este programa en sus diferentes módulos, todas las rutinas que manejan un aspecto del problema pueden ser reunidas juntas pero también pueden ser aisladas del código que debe lidiar con otras partes del problema. Los módulos identificados en este programa son listados en la figura 5.1.

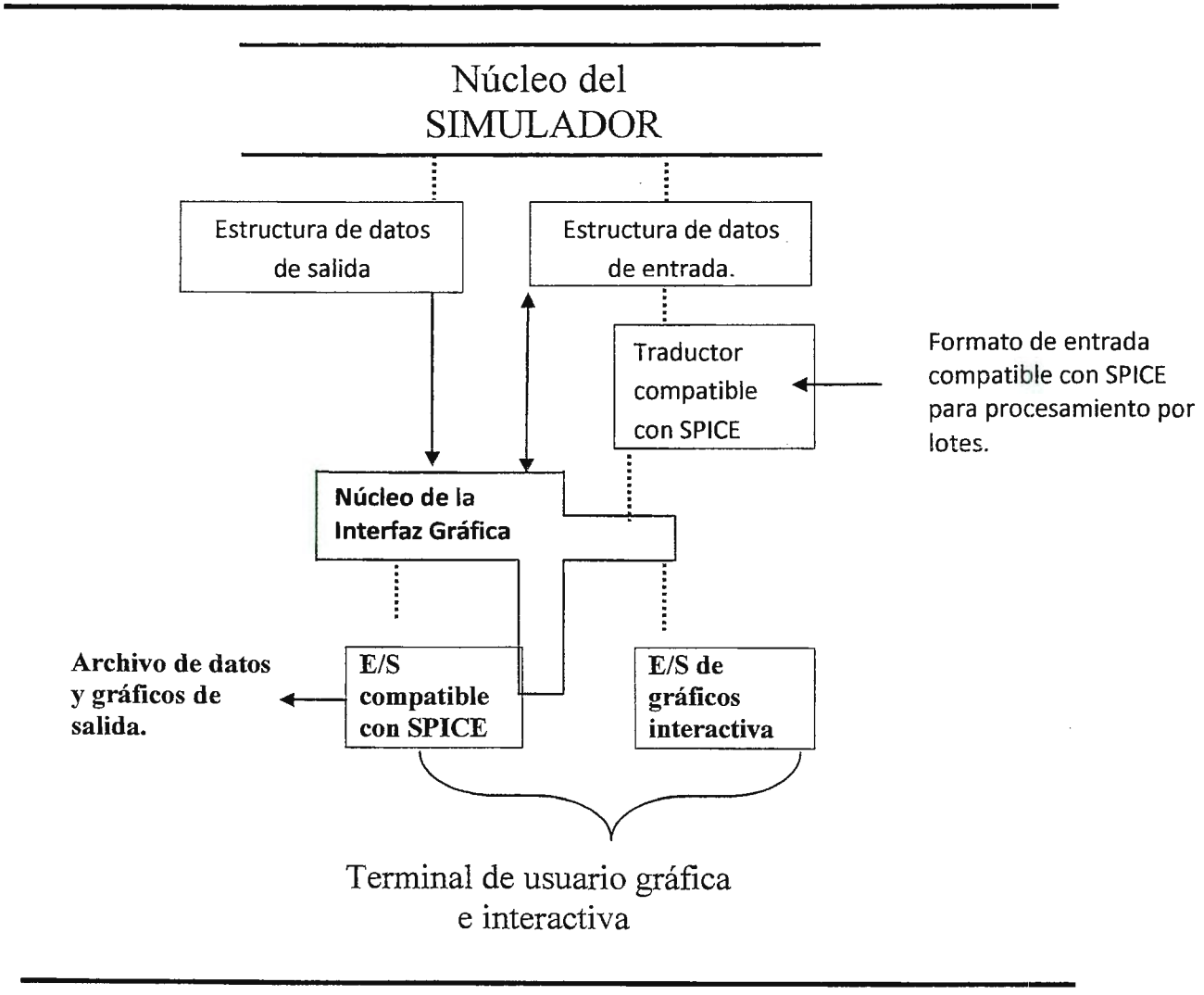


Figura 5.1

Módulos principales en el programa.

Debido a que cada uno de estos módulos es independiente de las operaciones o detalles de los otros módulos los detalles de la implementación son de poca importancia siempre y cuando cada paquete se mantenga como un todo consistente, manteniendo una interface predeterminada con el resto del programa. Una de las mejores ventajas de esto son las habilidades de dividir el mantenimiento de este programa y el esfuerzo de desarrollo en equipos de programadores, debido a que los módulos pueden ser mantenidos independientemente. En adición, cuando mejores algoritmos sean desarrollados, solo una pequeña parte de los módulos necesitará ser reemplazada en lugar del programa entero.

5.6. LA INTEGRACIÓN DE LOS MÓDULOS.

La integración de todos lo módulos individuales juntos en un total integrado ha sido completado cuidadosamente seleccionando cada módulo por las necesidades y capacidades de los módulos con que este interactúa. Las interfaces entre los módulos han sido cuidadosamente documentadas y ha hecho posible maximizar su facilidad de uso sin sacrificar demasiado la eficiencia. Así el módulo para el manejo de matrices no tendrá la flexibilidad de un paquete general más avanzado, pero provee todas las características que se requieren internamente de una manera compacta y eficiente. Debido a que todos lo módulos son autocontenidos, es también posible reemplazarlos con cualquier otro paquete que provean un conjunto de características más amplio de las capacidades correspondientes al módulo que se ha desarrollado, por ejemplo, cambiar el módulo de manejo de matrices por uno más avanzado.

Para entender la estructura de un programa grande como este, es necesario dividirlo en módulos, entonces estudiarlos individualmente, y finalmente observar como ellos interactúan con cada otro para producir el resultado final. En el caso de este programa, la estructura debe ser examinada desde diferentes perspectivas y desde varios niveles de complejidad.

Debido a que el campo del diseño de circuitos esta siempre cambiando y con nueva tecnología constantemente requiriendo cambios en el software de simulación, es importante para el simulador ser tan flexible como sea posible, manteniendo constante un alto rendimiento. Esto requiere la habilidad de añadir nuevos dispositivos, nuevos tipos de simulación, nuevos algoritmos de análisis, y nuevas formas de entrada de datos y formas de hacer gráficas de salida. Para este fin, este programa se ha hecho modular, haciendo posible hacer esto con tan poco trabajo adicional como sea posible. Esto requiere un gran número de estructuras de datos descriptivas publicas que describen capacidades de un nivel del código a otros niveles con otras estructuras de datos privados que se encuentran ocultos de otros niveles. Estas estructuras frecuentemente solo distantemente relacionadas son descritas independientemente. Finalmente, para entender la jerarquía del llamado de rutinas y el modo que los módulos se comunican entre sí.

La estructura básica de interfaces y llamadas entre las rutinas del programa es mostrada en la figura 5.2

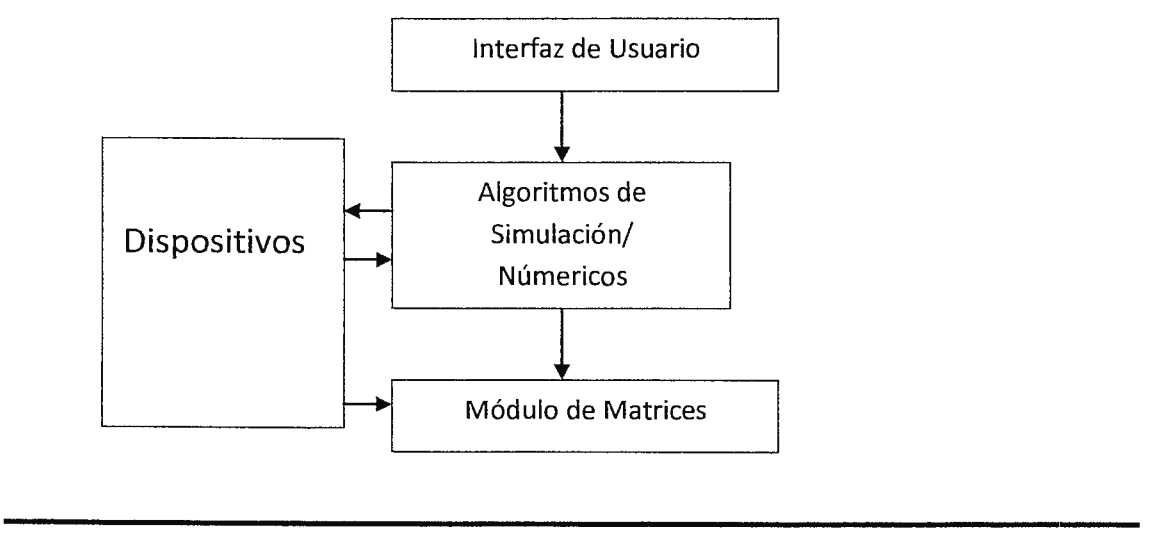


Figura 5.2

Estructura básica de interfaces entre módulos.

En esta figura, el bloque llamado "dispositivos" representan todos los paquetes por tipo de dispositivo que están incorporados en el programa. El uso de estos paquetes y su uso por los algoritmos numéricos del simulador. Ambos el código de programa de dispositivo y el código de las rutinas numéricas manipulan la matriz de datos del circuito a través del paquete de matrices. Los algoritmos numéricos y de simulación están ínter mezclados, debido a que están tan cercanamente relacionados y tienen un gran efecto el uno sobre el otro. La interfaz de usuario solo conoce los puntos de entradas provistos por la rutina de control de la simulación, solo las llama y controla la salida sin errores.

Una descripción básica de las estructuras de datos es mostrada en la figura 5.3

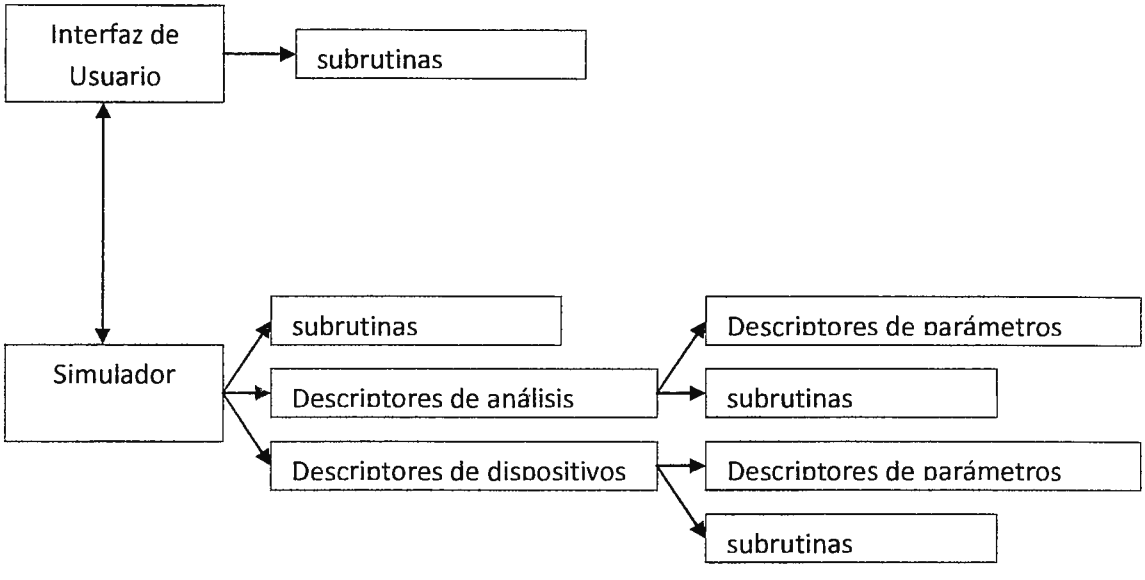


Figura 5.3

Diseño de las estructuras de datos.

En una estructura descriptiva de los datos, cada uno de los paquetes o módulos pequeños exporta una estructura que contiene una descripción de las capacidades de ese paquete en una forma estándar; Así la estructura para todos los paquetes similares, ya sean modelos de dispositivos o análisis, pueden ser recogidos en un solo arreglo en el siguiente nivel. En este pequeño nivel de paquete o módulo, la descripción consiste de uno o más arreglos de descripciones de parámetros y un grupo de punteros a funciones que implementan parte de las capacidades del modulo.

Al siguiente nivel, la información de todos los módulos en el simulador la cual se necesita por la interfaz de usuario será recogida por una sola estructura y exportada. Similarmente, la interfaz de usuario colecta todos los datos que serán necesitados por el simulador en una única estructura que se exporta al simulador durante la inicialización. En el tiempo de ejecución, tanto la interfaz de usuario, como el simulador, cada uno usan los descriptores provistos por el otro para determinar las capacidades disponibles, los parámetros usados para varias llamadas, y los tipos de argumentos pasados por los parámetros.

Finalmente, la estructura de datos privados es mostrada en la figura 5.4.

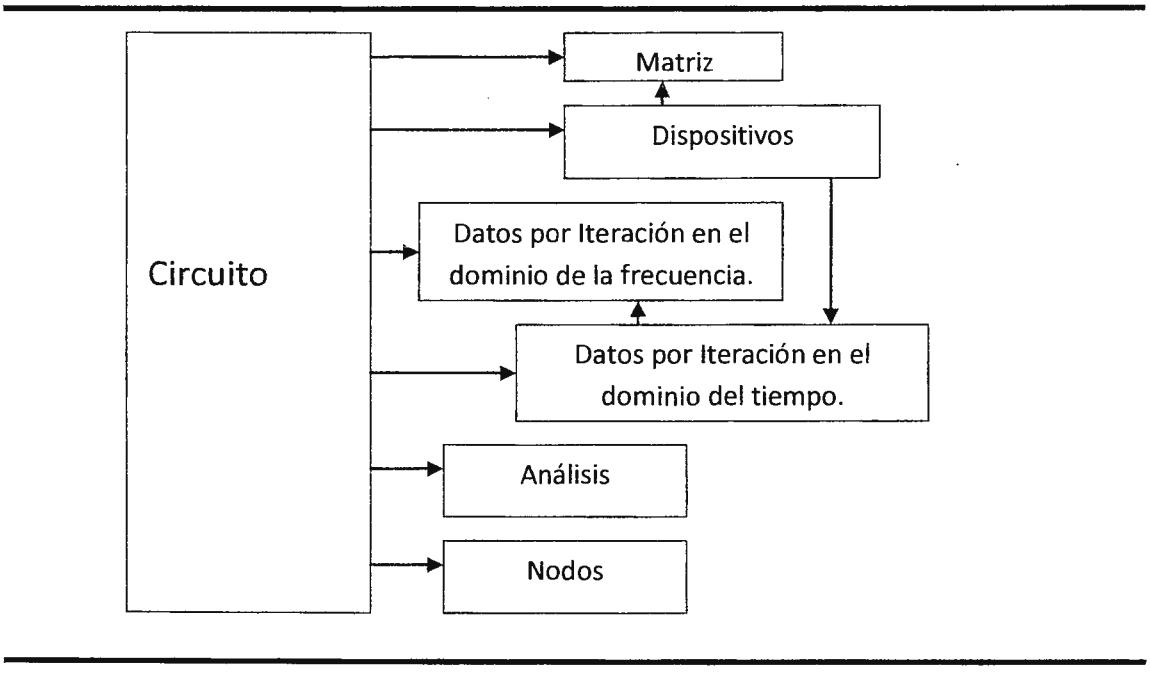


Figura 5.4

Estructura local de datos privados.

En esta estructura, la estructura del circuito es usada para encapsular todos los datos relacionados, desde el punto de vista del simulador, a un único circuito, esto puede ser actualmente ser un subcircuito provisto por otro simulador más complejo que usa este simulador por un evaluador de subcircuitos, pero nuestro que para nuestro simulador se trata internamente como un circuito independiente.

Esta estructura contiene una considerable cantidad de datos globales, así como punteros de estructuras más especializadas de módulos individuales. Las otras estructuras son usadas para mantener datos que son privados a un único módulo, o indirectamente referidos a través de un puntero. La estructura de la matriz del circuito es completamente privada al paquete de manejo de matrices. Aunque otros paquetes pueden obtener punteros a elementos específicos dentro de una matriz. Los dispositivos tienen una estructura de datos más complicada descrita en detalle más adelante.

Los datos son renombrados en cada iteración, el lado derecho y los vectores de solución, son mantenidos en un par de arreglos dinámicos locales, los cuales son referidos a través del programa. Los datos de cada análisis son mantenidos en una lista de estructuras dependientes del tipo de análisis para el uso privado del código fuente implementado para cada uno de ellos. Finalmente las descripciones de los nodos en el circuito son mantenidas en una tabla de nodos y mantenidas por un pequeño número de rutinas dentro del módulo de simulación mismo.

La arquitectura completa del sistema del simulador es relativamente simple. El programa está dividido en un grupo independientes de módulos, cada uno de los cuales conoce la interfaz presentada por otros módulos y la operación que ellos realizan, no los detalles de sus operaciones o estructura de datos.

Esto permite a los módulos ser mantenidos, mejorados, o reemplazados independientemente. La llave para entender la operación y estructura de los módulos y del simulador en su conjunto es la descripción de cada uno de estos paquetes y su estructura de datos correspondientes así como su interacción. Los detalles de estas interfaces son encontradas en esta documentación y frecuentemente aparecen del código usando el paquete ó módulo. La cosa más importante de entender y conocer es en que parte de la documentación, en cual módulo hay que buscar, cuando una funcionalidad requerida es necesitada, un error es encontrado, y este se encuentra asociado a un tipo particular de operación.

5.7. ESTRUCTURAS PRINCIPALES DE DATOS.

El primer paso para entender las estructuras básicas y su rango de operación. En muchas maneras, la estructura del programa sigue una estructura de datos, y es fácilmente entendida por las otras.

El área de datos del programa está divido en tres categorías:

- Datos estáticos que son usados para describir módulos del simulador a otros módulos del simulador, el simulador a la interfaz de usuario, y ciertas constantes físicas. Esta área esta reservada con anterioridad y inicializada en varias partes del simulador y grabada en variables estáticas globales. Todo dato global verdadero es constante.
- Otro dato que puede ser normalmente global en un simulador. No existe variables globales en este programa. Datos usados ampliamente a través de los diferentes módulos son guardados en una estructura de datos, un puntero el cual es pasado a casi a cualquier rutina para permitir múltiples instancias de ellos existir independientemente. Esto facilita el uso del programa en circuitos múltiples o como un analizador de subcircuitos dentro de un sistema más grande. Esta estructura de datos es la estructura conocida como circuito global y la mayoría de rutinas dentro del simulador que conocen cualquier detalle del simulador tienen conocimiento de esta estructura que hacen referencia a los valores "globales".
- Finalmente, cada paquete o grupo de rutinas pueden tener sus propias estructuras privadas que pueden ser definidas sin afectar ningún otro código.

Algunos paquetes, como el paquete de operaciones numéricas, simplemente guardan sus datos en la estructura del circuito, otros tienen sus propias estructuras privadas que pueden ser reservadas en una o más instancias de ellos. Adicionalmente, algunos módulos son simplemente instancias de una clase de módulos similares o tiene una estructura que pueden complementarse con reglas para esta clase de módulos. Las siguientes secciones proveen una pequeña descripción de las estructuras usadas. Más detalles de las estructuras son necesarias para hacer modificaciones y serán explicadas más adelante.

5.8. EL FLUJO DE CONTROL.

El manejo del flujo de control del simulador es bastante similar a su estructura de datos de variables. La interfaz de usuario permite la definición de las funciones de la interfaz de usuario y la consiguiente clasificación de las tareas que el usuario desea realizar. Usando la especificación de la interfaz de usuario hacia el simulador, la interfaz de usuario pasa la descripción del circuito y los análisis deseados al módulo de simulación de circuitos. En muchos casos la información es específica a un dispositivo en particular o análisis, y después de encontrar las propias estructuras de datos usando los prefijos comunes, la rutina específica del dispositivo ó del análisis es llamada para realizar la operación actual. Cuando la operación de análisis es ejecutada, el modulo del circuito llama a las subrutinas específicas de cada dispositivo para realizar todas las operaciones necesarias para configurar sus parámetros así como para inicializar la matriz para la simulación. Los módulos específicos a cada dispositivo entonces ocupan el módulo de operación de matrices para recoger los punteros específicos a los lugares específicos de la matriz que serán referenciados regularmente. Finalmente, el módulo de iteración numérica es llamado y atraviesa los pasos a través de la iteración de Newton- Raphson en cada punto de solución, usando las rutinas especificas de los dispositivos y de la matriz para realizar las operaciones a bajo nivel.

5.9. MÓDULOS PRINCIPALES.

Los módulos principales usados en este programa pueden ser catalogados en dos grandes categorías, las que son especificas y tienen una sola instancia, y aquellas que son genéricas y tiene muchas instancias, todas las cuales forman parte del ambiente básico.

Paquete de resolución de matrices.

Este módulo maneja las matrices del tipo que son encontradas en la simulación de circuitos. Este es un paquete general, que solo maneja las funciones básicas para la aplicación específica. El paquete genera las variables para guardar las matrices, crea los elementos en ellas, realiza la heurística para el reordenamiento basado en el conocimiento de la estructura de las matrices usada en la simulación del circuito, permite la factorización L-U con y sin reordenamiento, y permite la substitución de atrás hacia delante requerida para completar las solución de las ecuaciones del circuito.

El módulo de manejo del circuito.

Este conjunto de rutinas maneja el control de la simulación en el sistema, guía la secuencias de análisis y ciclos a través de los variados dispositivos y provee el punto de acceso de la interfaz de usuario a la estructura de datos del simulador.

El módulo de dispositivos.

Cada dispositivo es representado por un módulo que puede realizar todas las acciones necesarias para el simulador en las instancias de los modelos de software de los dispositivos del tipo especificado. Este módulo provee una interfaz estándar al simulador, permitiendo manejar todos los dispositivos en una manera uniforme. Este tipo de acciones que son disponibles para el dispositivo deben ser un conjunto de aquellas necesitadas por cualquier dispositivo para ser implementadas en este software, así el paquete puede tener entradas no usadas de datos para casi todos los tipos de dispositivos, puesto que estos no lo requieren, aunque otro tipo de dispositivo si los necesite.

El paquete o módulo de análisis.

Por cada tipo de análisis que es realizado por el simulador, existe un módulo que realiza la secuencia a través del algoritmo básico del análisis, llamando a las funciones de los dispositivos, las operaciones del paquete numérico, las operaciones de matrices, y las operaciones de salida necesarias para realizar el análisis requerido.

El módulo de análisis numérico.

Este módulo contiene la iteración numérica básica y la rutina de integración usada por el simulador. Este calcula los coeficientes de integración para los predictores y los correctores, a fin de mejorar la predicción actual y llevar el ciclo de iteraciones hacia la respuesta correcta.

El Módulo de graficas de salida.

Este se encarga de traducir los vectores de salida del sistema en gráficas que puedan ser vistas por el usuario.

PARTE IV: VALIDACION

6. PRUEBAS DEL SISTEMA

Prueba del simulador con circuitos acordes a los alcances y limitaciones previamente definidos.

6.1. CARACTERIZACIÓN DEL PROBLEMA Y LAS PRUEBAS DE DESEMPEÑO.

Uno de los problemas para analizar una herramienta de ingeniería compleja como este simulador es la dificultad de definir exactamente los problemas que esta destinado a resolver. Hay muchas diferentes medidas del rendimiento de un programa como este no todas de estas satisfactorias. Es imposible probar un simulador contra todos los posibles circuitos, y dada la complejidad del programa y los circuitos que esta intentando resolver, intentar probar que un conjunto seleccionado de circuitos es suficiente para demostrar la robustez general y el rendimiento del programa es también bastante difícil. En adición, las diferencias entre las plataformas de hardware en términos de rendimiento y precisión de puntos flotantes, tiempo de referencia, cantidad de memoria y el rendimiento con las operaciones de enteros pueden afectar el rendimiento general del programa. La prueba de este programa ha sido realizada usando pequeños circuitos de testeo diseñados para probar dispositivos particulares o características del programa.

Los resultados de estos análisis son mostrados en la tabla siendo una muestra representativa con énfasis en los circuitos que generan problemas y así como una selección de circuitos que pueden ser resueltos con la ayuda de este simulador.

Circuito	Nodos	Puertos	Armónicos +DC	Elementos No Lineales	Variables	Iteraciones	Convergencia		Tiempo Final
							Sin Est. Inicial	Con Est. Inicial	
Filtro y Diodo	2	2	20	1	20	178	SI	-	186 secs
Puente de Diodos	4	3	8	4	16	305	SI	-	186 secs
Amplificador Clase E	6	5	12	1	36	174	NO	SI	296 secs

7. CONCLUSIONES

Una técnica básica en el análisis de circuitos de RF es el método del balance armónico. El presente trabajo fue desarrollado con la visión de ser un trabajo pionero en el país en el desarrollo de software de simulación de circuitos. Aunque las técnicas y herramientas de simulación de circuitos llevan más de 30 años de desarrollo a nivel mundial, no existe en el país interés en desarrollar herramientas de este tipo.

Este trabajo no intenta comenzar un herramienta de cero, ignorando los años de trabajo en otras partes del mundo, más bien busca a partir de toda la información de dominio público, comenzar un trabajo en esta área, que sirva de base para comenzar a alcanzar de manera local, el avance mundial en este tipo de herramientas, para luego servir de base para avances tecnológicos futuros en el país.

Por lo que este trabajo en si mismo no es un fin, sino un medio para posteriormente alcanzar el resultado deseado. Entre las herramientas comerciales actuales existen simuladores de balance armónico de una sola frecuencia fundamental y de varias, con modelos de dispositivo basados en parámetros físicos, o en forma de parámetros Y o S, que incorporan análisis de pequeña/gran señal, análisis por medio de Series de Volterra. Existen además en la investigación académica prototipos de simuladores basados en el método de polinomios de Chebishev, diferentes formas de mejorar el método de balance armónico, por medio de mapeo en el tiempo, o utilizando wavelets.

La contribución que este trabajo aporta a la investigación académica, es que existen pocas herramientas de dominio público para la simulación de circuitos de RF. Además, deja la base para un prometedor desarrollo posterior de la aplicación, que se diseño desde sus inicios con el potencial de desarrollar una aplicación de calidad comercial, no como otra herramienta extravagante de investigación académica. Dependerá del interés de los encargados del proyecto, de ver el potencial de la aplicación que necesitará varios años de desarrollo y la aportación de varias personas para alcanzar su verdadero potencial, el ser un simulador multipropósito de circuitos de RF. Actualmente, a nivel de demostración, puede servir como herramienta, en cursos de análisis y diseño de circuitos de RF, para mostrar una aplicación práctica del método del balance armónico y los resultados que se pueden derivar de este análisis.

7.1. PROBLEMAS ENFRENTADOS

Debido a los diferentes obstáculos que fueron surgiendo en el diseño de este prototipo, las siguientes características no fueron desarrolladas al nivel optimo, aunque si lo necesario para cumplir los objetivos que se habían planteado.

La interfaz gráfica solo alcanzo un nivel básico de funcionalidad, el manejo de los nodos del circuito es rudimentario, no se automatizo el calculo del numero de puertos y fuentes.

Si se compara la interfaz gráfica con la de un programa comercial como PSPICE, pueden observarse las siguientes deficiencias:

- No se ha implementado un gris en el esquemático.
- La interfaz no mantiene las conexiones unidas en el esquemático al mover los componentes
- No se implemento la presentación de textos en el esquemático.

En cuanto al sistema de simulación no se logro integrar de manera completa y transparente el método del balance armónico como un análisis adicional del SPICE, ni se alcanzo un nivel de optimización y eficiencia del código para su uso a nivel comercial.

7.2. LOGROS ALCANZADOS

Los avances logrados por este trabajo fueron:

- ✓ Una interfaz gráfica básica con capacidad de crecimiento, que puede ser compilada en Windows y Linux, y que tiene la capacidad de generar archivos de descripción de circuitos con el estándar de spice3f5, que con un trabajo posterior puede convertirse en un programa de esquemáticos de calidad comercial.
- ✓ Un módulo de análisis de balance armónico de una sola frecuencia fundamental funcional, que genera archivos de salida tipo "raw", en el formato de spice3f5; que con un mayor trabajo puede ser optimizado para la velocidad y eficiencia, e implementado de manera transparente al paquete de simulación de circuitos Spice.

7.3. TRABAJOS FUTUROS

Los trabajos posteriores, que continúan luego a la finalización de este trabajo, son en el siguiente orden:

- ✓ Modificar el módulo de balance armónico actual para poder realizar análisis de balance armónico en varias frecuencias o multitono. Agregar dispositivos como líneas de transmisión con o sin perdidas, microcintas al programa de esquemáticos y las optimizaciones necesarias para integrar transparentemente el módulo en el programa spice.
- ✓ Portar el código al ngspice u otro motor que incorpore análisis mixto análogo/digital.
- ✓ Agregar características de modelos de dispositivos basados en sus parámetros físicos.

BIBLIOGRAFÍA

[1] Nonlinear microwave and RF circuits / Stephen A. Maas.—2nd ed. Artech House, 2003

[2] "A Piecewise Harmonic Balance Technique for Determination of Periodic Response of Nonlinear Systems" / M. S. Nakhla and J. Vlach, IEEE Trans. Circ. Syst., Vol. CAS-23, 1976

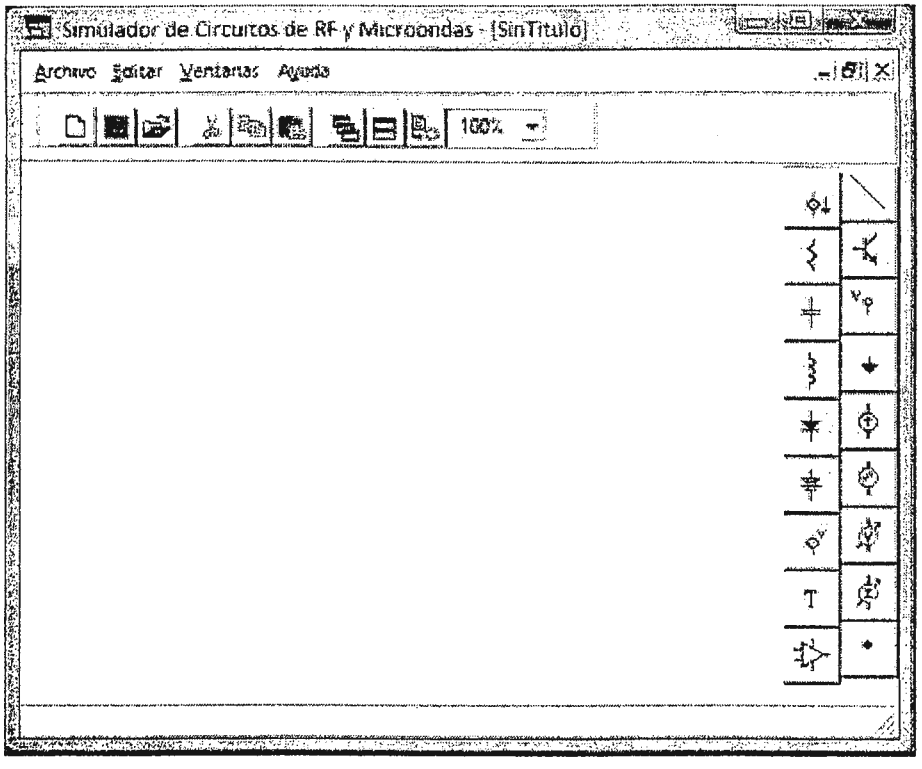
[3] Bridging the frequency-time domain gap / Paul Tuinenga, EEdesign.com, 2003
http://www.eetimes.com/news/design/columns/eda/showArticle.jhtml?articleID=17408760

[4] Patrones de Diseño para Interfaces de Usuario / Sandra Dinora Orantes, Tesis Maestría Instituto Politécnico Nacional, Centro de Investigación en Computación, México, 1999

[5] Analysis of Performance and Convergence Issues for Circuit Simulation / Thomas Linwood Quarles Memorando No. UCB-ERL M89-42, Abril 1989

Anexos

- **Manual de Usuario**



La ventana principal del programa del simulador de circuito consta de 5 partes para su funcionalidad.

- Barra de menú.
- Barra de herramientas.
- Area de dibujo.
- Paleta de simbología.
- Barra de Estado.

Manual del Usuario Spice

0.1 Introducción

Este Manual de Spice pretende ayudar al usuario que se encuentre por primera vez con esta herramienta. El simulador de circuitos por el método de balance armónico(HBSIM) está basado en SPICE3 (desarrollado en la Universidad de Berckeley) y, en consecuencia, las breves descripciones y ayudas sobre las posibilidades de este programa de simulación y los ejemplos que aquí se exponen son compatibles con cualquier otra versión basada en SPICE3. Aprenderemos ahora cómo crear correctamente los ficheros de las características de entrada de un circuito, y cómo se realizan cada uno de los distintos tipos de análisis que podemos llevar a cabo. El entorno gráfico es una herramienta potente diseñado para hacer el análisis de circuitos con SPICE3 más interactivo y eficiente. El dominio de estas dos herramientas será de gran ayuda para el lector a la hora de comprender y diseñar circuitos electrónicos

TIPOS DE ANÁLISIS

ANÁLISIS DC

La parte de análisis DC de Spice determina el punto de operación del circuito en DC con las inducciones en cortocircuito y los condensadores en abierto. Las opciones de análisis están especificadas en las líneas de control .DC, .TF, y .OP. Antes de un análisis transitorio se realiza un análisis DC para determinar las condiciones transitorias iniciales, y antes de un análisis AC en pequeña señal para determinar la linealidad del modelo en pequeña señal en dispositivos no lineales. Si queremos, el valor DC en pequeña señal de la función de transferencia (salida/entrada), resistencia de entrada y resistencia de salida son también calculadas como parte de la solución del análisis DC. El análisis DC también puede usarse para generar curvas de transferencia DC: especificando voltaje independiente o fuente de corriente en un rango de valores dados por el usuario, la salida DC se almacena para cada valor secuencial de la fuente.

ANÁLISIS EN PEQUEÑA SEÑAL DE UNA CORRIENTE ALTERNA

La parte de análisis DC de Spice determina el punto de operación del circuito en DC con las inducciones en cortocircuito y los condensadores en abierto. Las opciones de análisis están especificadas en las líneas de control .DC, .TF, y .OP. Antes de un análisis transitorio se realiza un análisis DC para determinar las condiciones transitorias iniciales, y antes de un análisis AC en pequeña señal para determinar la linealidad del modelo en pequeña señal en dispositivos no lineales. Si queremos, el valor DC en pequeña señal de la función de transferencia (salida/entrada), resistencia de entrada y resistencia de salida son también calculadas como parte de la solución del análisis DC. El análisis DC también puede usarse para generar curvas de transferencia DC: especificando voltaje independiente o fuente de corriente en un rango de valores dados por el usuario, la salida DC se almacena para cada valor secuencial de la fuente.

ANÁLISIS TRANSITORIO

El análisis transitorio de Spice calcula el transitorio de las variables de salida en el intervalo de tiempo especificado por el usuario. Las condiciones iniciales son automáticamente determinadas por el análisis DC. Todas las fuentes que no son dependientes del tiempo (por ejemplo, fuentes de alimentación) son puestas a su valor DC. El tiempo transitorio esta especificado en la línea de control .TRAN.

ANÁLISIS DE POLOS Y CEROS

El análisis de polos y ceros de Spice calcula los polos y/o ceros en pequeña señal de la función de transferencia en AC. El programa calcula el punto de operación en DC y luego trata de linearizar el modelo en pequeña señal para todos los dispositivos no lineales del circuito. Este circuito es usado para encontrar los polos y/o ceros de la función de transferencia.

Hay dos tipos de función de transferencia: Uno es (Voltaje de salida)/(Voltaje de entrada) y el otro tipo es (Voltaje de salida)/(Corriente de entrada). Estos dos tipos de función de transferencia cubren todos los casos, en donde se puede encontrar los polos/ceros de funciones como la impedancia de entrada /salida y la ganancia de salida. Los puertos de entrada y salida están especificados como dos pares de nodos.

El análisis de polos y ceros trabaja con resistencias, capacidades, inductancias, fuentes lineales, fuentes independientes, BJTs, MOSFETs, JFETs y diodos. Las líneas de transmisión no son soportadas.

El método usado en el análisis es una búsqueda numérica poco optimizada. Para grandes circuitos este proceso lleva un tiempo considerable en encontrar todos los polos y ceros. Para algunos circuitos el método se vuelve "irrescatable" encontrando un excesivo número de polos y ceros.

ANÁLISIS DE DISTORSIÓN EN PEQUEÑA SEÑAL

La parte de análisis de distorsión de Spice calcula el estado estable del armónico y la intermodulación para las pequeñas magnitudes de la señal de entrada. Si las señales de entrada son de una sola frecuencia, entonces los valores complejos del segundo y tercer armónicos son calculados en cada punto en el circuito.

Si hay señales de dos frecuencias, entonces el análisis busca fuera de los valores complejos de las variables del circuito, en la suma y la diferencia de las frecuencias de entrada, en la diferencia del de frecuencia más pequeña y del segundo armónico de mayor frecuencia.

El análisis de distorsión se usa para los siguientes dispositivos no lineales: Los diodos (DIO), BJT, JFET, MOSFETs (niveles 1, 2, 3, 4/BSIM1, 5/BSIM2, y 6) y MESFETs. Todos los dispositivos lineales se les calcula automáticamente el análisis de distorsión. Si hay interruptores presentes en el circuito, el análisis continúa siendo preciso previendo que los interruptores no cambien bajo las pequeñas excitaciones usadas para cálculos de distorsión.

ANÁLISIS DE SENSIBILIDAD

Spice calculará o la sensibilidad del punto de operación en DC o la sensibilidad de la pequeña señal en AC de una variable de salida con respecto a todas las variables del circuito, incluyendo parámetros modelo. Spice calcula la diferencia en una salida variable (ya sea un voltaje de nodo o una corriente de rama) perturbando cada parámetro de cada dispositivo independientemente. Desde que el método es una aproximación numérica, los resultados pueden demostrar que los efectos de segundo orden afectan a los parámetros altamente sensitivos, puede fallar al mostrar sensibilidad muy próxima a cero. Adicionalmente, si la variable es perturbada minimamente, los parámetros apreciados en cero no son analizados (esto reduce la cantidad de datos).

ANÁLISIS DE RUIDO

La parte de análisis de ruido de Spice hace el análisis del ruido generado en un dispositivo para el circuito dado. Estando provisto de una fuente de entrada y un terminal de salida, el análisis calcula las contribuciones de ruido de cada dispositivo (y cada generador de ruido dentro del dispositivo) que generan voltaje a la salida. También calcula el ruido de entrada para el circuito, equivalente para el ruido de salida referido a la fuente especificada de entrada. Esto está hecho para cada frecuencia en un rango especificado, el valor calculado del ruido es propio de la densidad espectral del circuito vista como un proceso estocástico de Gauss estacionario.

Después de calcular las densidades espectrales, el análisis de ruido integra estos valores sobre el rango de frecuencias especificadas para llegar al voltaje/corriente total de ruido(sobre este rango de frecuencias). Este valor calculado corresponde a la variación entre lo que el circuito variable viese y un proceso de Gauss estacionario.

ANÁLISIS A DIFERENTES TEMPERATURAS

Todos los datos de entrada para Spice se meden a una temperatura nominal de 27 C, lo cual puede variarse usando el parámetro TNOM en la línea de control .OPTION. Este valor se puede incrementar

para cualquier dispositivo especificando el parámetro TNOM en su propio modelo. La simulación del circuito se realiza a una temperatura de 27 C, a menos que se modifique el parámetro TEMP en la línea de control .OPTION. Las especificaciones individuales pueden fomentar el incremento de temperatura del circuito por medio del parámetro TEMP.

La dependencia de la temperatura para resistencias, condensadores, diodos, JFETs, BJTs, nivel 1, 2, y 3 MOSFETs. BSIM (niveles 4 y 5) MOSFETs tiene un esquema diferente de dependencia con la temperatura que ajusta todos los parámetros modelo antes de la entrada para SPICE. Para los detalles del ajuste BSIM de temperatura, ver [6] y [7] .

La temperatura aparece explícitamente en los términos exponenciales del BJT y en las ecuaciones modelo del diodo. Además, las corrientes de saturación contribuyen a la dependencia de temperatura. La dependencia de temperatura de la corriente de saturación en los modelos BJT está resuelta por:

$$I_S(T_1) = I_S(T_0)\left(\frac{T_1}{T_0}\right)^{XTI} \exp\left(\frac{E_g q(T_1 T_0)}{k(T_1 - T_0)}\right)$$

Dónde k es la constante de Boltzmann, q es la carga eléctrica, Eg es el energía de la banda prohibida que es un parámetro modelo, y XTI es el exponente de temperatura de la corriente de saturación (también un parámetro modelo, y usualmente igual a 3).

La dependencia de temperatura antes y después de la beta es de acuerdo a la fórmula:

$$\beta(T_1) = \beta(T_0)\left(\frac{T_1}{T_0}\right)^{XTB}$$

Dónde T1 y T0 están en grados Kelvin, y XTB es un parámetro modelo suministrado por el usuario. Los efectos de temperatura en la beta son extraídos por el ajuste apropiado para los valores de bF, ISE, bR, e ISC (parámetros modelo Spice BF, ISE, BR, e ISC, respectivamente).

La dependencia de temperatura de la corriente de saturación en el modelo del diodo de unión está resuelta por:

$$I_S(T_1) = I_S(T_0)\left(\frac{T_1}{T_0}\right)^{\frac{XTI}{N}} \exp\left(\frac{E_g q(T_1 T_0)}{Nk(T_1 - T_0)}\right)$$

Donde la N es el coeficiente de emisión, el cual es un parámetro modelo, y los otros símbolos tienen el mismo significado citado anteriormente. Note que para diodos de barrera Schottky, el valor del exponente de temperatura de corriente de saturación, XTI, es usualmente 2.

La temperatura aparece explícitamente en el valor de potencial de unión, f (en Spice PHI), para todo los dispositivos modelo. La dependencia de temperatura viene dada por:

$$\Phi(T) = \frac{kT}{q}\log_e\left(\frac{N_a N_d}{N_i(T)^2}\right)$$

Dónde k es la constante de Boltzmann, q es la carga eléctrica, Na es la densidad de la impureza del aceptador, Nd es la densidad de impurezas del donador, Ni es la concentración intrínseca del portador, y Eg es la energía de la banda prohibida.

La temperatura aparece explícitamente en el valor de movilidad de la superficie, m0 (o UO), para el modelo MOSFET. La dependencia de temperatura viene determinada por:

$$\mu_0(T) = \frac{\mu_0(T_0)}{\left(\frac{T}{T_0}\right)^{1.5}}$$

Los efectos de la temperatura en resistencias son modelados por la fórmula:

$$R(T) = R(T_0)\left[1 + TC_1(T - T_0) + TC_2(T - T_0)^2\right]$$

Donde la T es la temperatura del circuito, T0 es la temperatura nominal, y TC1 y TC2 son los términos de primer y segundo orden de temperatura.

Manual de Spice: Capitulo 3

Descripción de los circuitos

3.1 Estructura general y convenios

El circuito analizado está descrito para el Spice como un conjunto de líneas de elemento, las cuales definan la topología del circuito y del elemento, así como un grupo de líneas de control, las cuáles definen los parámetros modelo y los controles usados. La primera línea en el archivo de entrada debe ser el título, y la última línea debe ser ".END". La orden de las líneas restantes es arbitraria (excepto, claro está, las líneas de continuación que deben seguir siendo continuadas). Una línea de elemento contiene: el nombre del elemento, los nodos del circuito en los que el elemento está conectado, y los valores de los parámetros que determinan las características eléctricas del elemento en el circuito. La primera letra del nombre del elemento especifica el tipo del elemento. El formato para los tipos de elemento SPICE viene dado más adelante. El strings XXXXXXX, YYYYYYY, y ZZZZZZZ denotan a strings alfanuméricos arbitrarios. Por ejemplo, el nombre de una resistencia debe comenzar con la R y puede contener a uno o más caracteres. Por lo tanto, R, R1, RSE, ROUT y R3AC2ZY son nombres válidos de la resistencia. Los detalles de cada tipo de dispositivo son especificados en un siguiente capítulo. Los campos en una línea son separados por uno o más espacios vacíos, una coma, un signo igual ('='), o un paréntesis izquierdo-derecho ( ); Los espacios adicionales son ignorados. Una línea puede ser continuada

introduciendo un '+' en la columna 1 de la siguiente línea; Spice continúa leyendo a partir de la columna 2.

Un campo de nombre debe comenzar a con una letra (A - Z) y no puede contener ningún delimitador.

Un campo de número puede ser un entero (12, -44), coma flotante (3.14159), ya sea un entero o número de coma flotante seguido por un exponente entero (1e-14, 2.65e3), o ya sea un entero o un número de coma flotante siguió una de los siguientes factores de escala:

$T = 10^{12}$	$G = 10^{9}$	$Meg = 10^{6}$	$K = 10^{3}$	$mil = 25.4^{-6}$
$m = 10^{-3}$	$u(o\ M) = 10^{-6}$	$N = 10^{-9}$	$p = 10^{-12}$	$f = 10^{-15}$

Las letras inmediatamente después de un número que no son factores de escala son ignoradas, y las letras inmediatamente después de un factor de escala son ignoradas. Por lo tanto, 10, 10V, 10Volts, y 10Hz todo representan el mismo número, y M, MA, MSec, y MMhos todos representan el mismo factor de escala. Note que 1000, 1000.0, 1000Hz, 1e3, 1.0e3, 1KHz, y 1K todo representan el mismo número.

Los nombres de nodos pueden ser cadenas de caracteres arbitrarias. El nodo referencia (tierra) debe ser llamado '0'. Note la diferencia en Spice donde los nodos son tratados como cadenas de caracteres y no evaluados como números, así '0' y '00' son nodos distintos en Spice pero no en SPICE2. El circuito no puede contener un lazo de fuentes de voltaje y/o los inductores y no puede contener una fuente de corriente y/o los condensadores en corte. Cada nodo en el circuito debe tener un camino DC a tierra. Cada nodo debe tener al menos dos conexiones excepto por nodos de una línea de transmisión (para permitir líneas no terminadas de transmisión) y los nodos del substrato MOSFET (que tenga dos conexiones internas).

3.2 Línea de título, línea de comentario y línea .END

3.2.1 Línea título

Ejemplos:

POWER AMPLIFIER CIRCUIT

TEST OF CAM CELL

La línea del título debe ser lo primero en el archivo de entrada. Sus contenidos son impresos literalmente como el encabezamiento para cada sección de salida.

3.2.2 Línea .END

Ejemplos:

.END

La línea "End" siempre debe ser lo último en el archivo de entrada. Note que el punto es parte del nombre.

3.2.3 Comentarios

Forma general:

*

Ejemplos:

* RF=1K Gain should be 100

* Check open-loop gain and phase margin

El asterisco en la primera columna señala que esta línea es una línea del comentario. La línea de comentario puede ser colocada en cualquier parte de la descripción del circuito. Note que el Spice también considera cualquier linea con espacio blanco delante como un comentario.

3.3 .MODEL: Modelos de dispositivos

Forma general:

.MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ...)

Ejemplos:

.MODEL MOD1 NPN (BF=50 IS=1E-13 VBF=50)

Los elementos del circuito más sencillos típicamente requieren sólo unos pocos valores de parámetros. Sin embargo, algunos dispositivos (semiconductores en particular) que son incluidos en Spice requieren muchos valores de parámetro. A menudo, muchos dispositivos en un circuito están definidos por el mismo conjunto de parámetros de modelo del dispositivo. Por estas razones, un conjunto de parámetros de modelo de dispositivo están definidos en una línea separada .MODEL y asignado uno único nombre. Las líneas de elemento de dispositivo en Spice además se refieren al nombre modelo.

Para los dispositivos más complejos, cada línea de elemento del dispositivo contiene el nombre del dispositivo, los nodos a los cuales el dispositivo está conectado, y el nombre de modelo del dispositivo. Además, otros parámetros optativos pueden estar especificados para algunos dispositivos: Los factores geométricos y una condición inicial (vea el siguiente capítulo de Transistores y Diodos para más detalles).

MNAME es el nombre modelo, y el tipo es uno de los siguientes:

R	Modelo semiconductor de la resistencia
C	Modelo semiconductor del condensador
SW VSWITCH	Voltaje controlado por interruptor
CSW ISWITCH	Corriente controlada por interruptor
URC	Distrubución uniforme del modelo RC
LTRA	Modelo de pérdidas de la línea de transmisión
D	Modelo de diodo
NPN	Modelo NPN BJT
PNP	Modelo PNP BJT
NJF	Modelo Canal-N JFET
PJF	Modelo Canal-P JFET
NMOS	Modelo Canal-N MOSFET
PMOS	Modelo Canal-P MOSFET
NMF	Modelo Canal-N MESFET
PMF	Modelo Canal-P MESFET

Los valores de parámetro son definidos añadiendo el nombre del parámetro seguido por un signo de igual y el valor de parámetro. A los parámetros del modelo que no reciben un valor se les asigna valores predeterminados para cada tipo de modelo. Modelos, parámetros modelo, y valores predeterminados están listados en el siguiente capítulo junto con la descripción de líneas del elemento del dispositivo.

3.4 Subcircuitos

Un subcircuito que consta de elementos de Spice puede estar definido y referenciado de forma parecido a los modelos del dispositivo. El subcircuito está definido en el archivo de entrada por un agrupamiento de líneas de elemento; El programa automáticamente inserta el grupo de elementos dondequiera que el subcircuito es llamado. No hay límite en el tamaño o la complejidad de subcircuitos, y los subcircuitos pueden contener otros subcircuitos. Un ejemplo de uso del subcircuito está en el Apéndice A.

3.4.1 Línea .SUBCKT

Forma general:

.SUBCKT subnam N1

Ejemplos:

.SUBCKT OPAMP 1 2 3 4

La definición del circuito comienza con una línea .SUBCKT. SUBNAM es el nombre del subcircuito, y N1, N2, son los nodos externos, el cual no puede ser cero. El grupo de líneas de elemento que inmediatamente siguen la línea .SUBCKT define el subcircuito. La última línea en una definición del subcircuito es la línea .ENDS (vea debajo). Las líneas de control no pueden aparecer dentro de una definición de subcircuito; Sin embargo, las definiciones del subcircuito pueden contener cualquier otra cosa, incluyendo otras definiciones de subcircuitos, los modelos del dispositivo y las llamadas del subcircuito (vea debajo). Note que cualquier modelo de dispositivo o cualquier definición de subcircuito incluida como parte de una definición del subcircuito son estrictamente locales (i.e. Tales modelos y definiciones no se conocen fuera del subcircuito). También, cualquier nodo del elemento no incluido en la línea .SUBCKT son estrictamente locales, a excepción de 0 (tierra) que es siempre global.

Otros nodos pueden hacerse globales usando la directiva .GLOBAL.

3.4.2 Línea .ENDS

Forma general:

.ENDS

Ejemplos:

.ENDS OPAMP

Las líneas "Ends" deben ser lo último para cualquier definición del subcircuito. El nombre del subcircuito, está incluido e indica que la definición del subcircuito ha terminado; si se omite, quiere decir que todos los subcircuitos definidos han terminado. El nombre es necesario sólo cuando las definiciones anidadas del subcircuito se están construyendo.

3.4.3 Línea .GLOBAL

Forma general:

.GLOBAL N1

Ejemplos:

.GLOBAL 1 2 3 9

Esta línea define un conjunto de nodos globales. Estos nodos no son modificados en la creación del subcircuito.

3.4.4 Xxxxx: Llamadas a subcircuitos

Forma general:

XYYYYYYY N1 SUBNAM

Ejemplos:

X1 2 4 17 3 1 MULTI

Los subcircuitos se usan en SPICE especificando pseudos-elementos empezando por la letra X, seguido por los nodos del circuito que van a usarse al crear el subcircuito.

3.5 Archivos combinados

3.5.1 Xxxxx: Llamadas a subcircuitos

Forma general:

.INCLUDE filename

.INCLUDE "filename with spaces.cir"

Ejemplos:

.INCLUDE \users\spice\common\wattmeter.cir

.INCLUDE "\users\spice files\wattmeter.cir"

Frecuentemente, partes de descripciones del circuito se volverán a usar en varios archivos de entrada, particularmente con subcircuitos y modelos comunes. En cualquier archivo de entrada SPICE, la línea .include puede usarse para copiar algún otro archivo como si ese segundo archivo apareciese en lugar de la línea ".include" en el archivo original. No hay restricción impuestas por SPICE en el nombre del archivo, más allá de las restricciones impuestas por el sistema operativo local.

Si el nombre del archivo o el camino contiene espacios, entonces hay que usar las dobles comillas.

3.5.2 Líneas .LIB

Forma general:

.LIB filename

.LIB "filenamewith spaces"

Ejemplos:

.LIB \users\spice\common\bipolar.lib

Ésta es una extensión, que no se encuentra en la versión Berkeley de SPICE3, eso otorga compatibilidad con versiones anteriores de PSPICE.

La línea .LIB es similar a la línea .INCLUDE excepto que en el archivo especificado se asume que contiene las definiciones .MODEL y .SUBCKT. Spice va en busca de cualquier subcircuito o modelo indefinido en el archivo especificado y extrae las definiciones requeridas y las pega en el circuito. La diferencia principal es que sólo extrae partes del archivo especificado, no incluye el archivo entero en su circuito, la línea .LIB usa mucha menos cantidad de memoria.

El archivo de entrada puede tener cualquier extensión, pero por convención tiene la extensión .lib.

Si el nombre del archivo o el camino contiene espacios, entonces las dobles comillas deben ser usadas.

Manual de Spice: Capitulo 4

Modelos y elementos del circuito

Los campos de dato que están entre los signos ('< >') son optativos. Todos los signos de puntuación (paréntesis, signos de igual, etc.) son optativos pero indican la presencia de delimitadores. Además, las implementaciones futuras pueden requerir esa puntuación. Un estilo coherente de puntuación simplifica la comprensión. Con relación a los voltajes y corrientes de la rama, Spice utiliza un convenio de referencia (corriente fluye en la dirección de que el voltaje desciende).

4.1 Dispositivos elementales

4.1.1 Rxxxx: Resistencias

4.1.1.1 Resistencias simples

Forma general:

RXXXXXXX N1 N2 VALUE

Ejemplos:

R1 1 2 100

RC1 12 17 1K

N1 y N2 son los nodos del elemento. VALUE es la resistencia (en ohmios) y puede ser positiva o negativa pero no cero.

4.1.1.2 Resistencias semiconductoras

Forma general:

RXXXXXXX N1 N2

Ejemplos:

RLOAD 2 10 10K

RMOD 3 7 RMODEL L=10u W=1u

Ésta es la más forma general de la resistencia presentado en el capítulo 4.1.1.1, y permite el modelaje de efectos de temperatura y para el cálculo del valor real de resistencia con información estrictamente geométrica y de las especificaciones del proceso.

Si VALUE es especificado, entonces pasa por encima de la información geométrica y define la resistencia. Si MNAME es especificado, entonces la resistencia puede calcularse de la información de proceso en el modelo MNAME con LENGTH y WIDTH dado. Si VALUE no es especificado, entonces MNAME y WIDTH deben ser especificados. Si LENGTH no es especificada, entonces es tomada de la anchura predeterminada en el modelo. El (optativo) valor TEMP es la temperatura en la cual este dispositivo debe funcionar, y pasa sobre la especificación de temperatura dada en la línea de control .OPTION.

4.1.1.3 Modelo de la resistencia semiconductora (R)

El modelo de la resistencia consta de datos relativos al proceso del dispositivo que permiten que la resistencia sea calculada con información geométrica y corregida en función de la temperatura.

Los parámetros disponibles son:

Nombre	Parámetro	Unidad	Defecto	Ejemplo
TC1	Coeficiente de primer orden de la temperatura	OHM/°C	0.0	-
TC2	Coeficiente de segundo orden de la temperatura	OHM/°C^2	0.0	-
RSH	Resistencia laminar	OHM/square	-	50
DEFW	Anchura por defecto	Metros	1e-6	2e-6
NARROW	Estrechamiento debido al grabado lateral	Metros	0.0	1e-7
TNOM	Temperatura de medida	°C	27	50

La resistencia laminar se usa con el parámetro de estrechamiento, L y W de la resistencia para determinar la resistencia nominal, por la fórmula

$$R = RSH\frac{L - NARROW}{W - NARROW}$$

DEFW se usa para suministrar un valor predeterminado a W si no está especificado para el dispositivo. Si tampoco RSH o L están especificados, entonces el valor predeterminado estándar de resistencia de 1k Z es el que se usa.

TNOM se usa para pasar por encima del valor de ancho dado en la línea de control .OPTIONS donde los parámetros de este modelo han sido medidos en una temperatura diferente. Después de que la resistencia nominal se calcula, está se ajusta para la temperatura, por la fórmula:

$$R(T) = R(T_0)[1 + TC_1(T - T_0) + TC_2(T - T_0)^2]$$

4.1.2 Cxxxx: Codensadores

4.1.2.1 Condensadores simples

Forma general:

CXXXXXXX N+ N- VALUE

Ejemplos:

CBYP 13 0 1UF

COSC 17 23 10U IC=3V

N+ y N- son los nodos positivo y negativo respectivamente. VALUE es la capacidad en Faradios. La (optativa) condición inicial es el valor inicial (tiempo cero) de voltaje del condensador (en Volts). Note que las condiciones iniciales (si cualquier) se aplican 'sólo' si la opción UIC es especificada en la línea de control .TRAN.

NOTA: A diferencia de Spice2, los condensadores no lineales que usan POLY no son directamente soportados por Spice. Sin embargo, pueden ser simulados usando fuentes de corriente o voltaje no lineales. La dependencia del voltaje y capacidad con la temperatura pueden simularse usando el condensador descrito en el capítulo 4.1.2.3.

4.1.2.2 Condensadores semiconductores

Forma general:

CXXXXXXX N1 N2

Ejemplos:

CLOAD 2 10 10P

CMOD 3 7 CMODEL L=10u W=1u

Ésta es la forma general del condensador presentado en el capítulo 4.1.2.1, y para el cálculo del valor real de la capacidad usa información estrictamente geométrica y de las especificaciones del proceso.

Si VALUE es especificado, entonces define la capacidad. Si MNAME es especificado, la capacidad se calcula de la información de proceso del modelo MNAME, LENGTH y WIDTH dado. Si VALUE no es especificado, entonces MNAME y LEGTH deben ser especificados. Si la WIDTH no se especifica, entonces se toma la anchura predeterminada dada en el modelo. Ya sea VALUE o MNAME, LENGTH, y WIDTH pueden estar especificados, pero no ambos grupos.

4.1.2.3 Modelo de capacidad semiconductora (C)

El modelo del condensador contiene información de proceso que puede usarse para computar la capacidad desde información estrictamente geométrica.

Nombre	Parámetro	Unidad	Defecto	Ejemplo
TNOM	Temperatura medida	°C	27	50
TC1	Coeficiente de primer orden de la temperatura	OHM/°C	0.0	-
TC2	Coeficiente de segundo orden de la temperatura	OHM/°C^2	0.0	-
VC1	Coeficiente de primer orden del voltaje	volt^1	0.0	-
VC2	Coeficiente de segundo orden del voltaje	volt^2	0.0	-
CJ	capacidad del final de la unión	F/metros^2	-	5e-5
CJSW	capacidad de la pared lateral de la unión	F/metros	-	2e-11
DEFW	Anchura por defecto	Metros	1e-6	2e-6
NARROW	Estrechamiento debido al grabado lateral	Metros	0.0	1e-7

La capacidad del condensador se calcula como:

CAP = CJ(LENGTH-NARROW)(WIDTH-NARROW)+2CJSW(LENGTH+WIDTH-2NARROW)

$$CAP = CJ(LENGTH - NARROW)(WIDTH - NARROW) + 2CJSW(LENGTH + WIDTH - 2NARROW)$$

TNOM se usa para pasar por encima del valor ancho del circuito dado en la línea de control .OPTIONS donde los parámetros de este modelo han sido medidos en una temperatura diferente.

Después de que la capacidad nominal se calcule, es ajustada para la no-linealidad de temperatura y voltaje, por la fórmula:-

$C_{eff} = CAP(1+VC_1 \cdot V_{cap}^2)(1+TC_1(T-T_{nom})+TC_2(T-T_{nom})^2)$

$$C_{eff} = CAP(1 + VC^1 V_{cap}^2)(1 + TC_1(T - T_{nom}) + TC_2(T - T_{nom})^2)$$

4.1.3 Lxxxx: Inductores

Forma general:

LYYYYYYY N+ N- VALUE

Ejemplos:

LLINK 42 69 1UH

LSHUNT 23 51 10U IC=15.7MA

N+ y N- son los nodos positivos y negativos respectivamente. VALUE es la inductancia n Henrios. La (optativa) condición inicial es el valor inicial (tiempo cero) de corriente del inductor (en Amps) que fluye desde N + a N-. Note que las condiciones iniciales (si cualquier) tienen aplicación sólo si la opción UIC es especificada en la línea de análisis .TRAN.

NOTA: A diferencia de Spice2, los inductores no lineales no son directamente soportados por Spice. Sin embargo, pueden ser simulados usando fuentes de corriente y voltaje no lineales.

4.1.4 Kxxxx: Inductores acoplados (Mutuamente)

Forma general:

KXXXXXXX LYYYYYYY LZZZZZZZ VALUE

Ejemplos:

K43 LAA LBB 0.999

KXFRMR L1 L2 0.87

LYYYYYYY y LZZZZZZZ son los nombres de los dos inductores acoplados, y el VALUE es el coeficiente de acoplamiento, K, el cual debe ser mayor que 0 y menor o igual que 1. Usando el convenio del 'punto', se coloca un 'punto' en el primer nodo de cada inductor.

4.1.5 Sxxxx y Wxxxx: interruptores

4.1.5.1 Sxxxx: Interruptor controlado por voltaje

Forma general:

SXXXXXXX N+ N- NC+ NC- MODEL

Ejemplos:

s1 1 2 3 4 switch1 ON

s2 5 6 3 0 sm2 off

Switch1 1 2 10 0 smodel1

Los nodos 1 y 2 son los nodos entre los cuales los terminales del interruptor están conectados. El nombre del modelo es obligatorio mientras las condiciones iniciales son optativas. Los nodos 3 y 4 son los nodos positivos y negativos que controla respectivamente.

4.1.5.2 Wxxxx: Interruptor controlado por corriente

Forma general:

WYYYYYYY N+ N- VNAM MODEL

Ejemplos:

w1 1 2 vclock switchmod1

W2 3 0 vramp sm1 ON

wreset 5 6 vclck lossyswitch OFF

Los nodos 1 y 2 son los nodos entre los cuales los terminales del interruptor están conectados. El nombre del modelo es obligatorio mientras que las condiciones iniciales son optativas. La corriente controlante es la que fluye a través de la fuente especificada de voltaje. La dirección positiva de la corriente es del nodo positivo al el nodo negativo de la fuente.

4.1.5.3 Modelo de interruptor (SW/CSW)

Forma general:

.MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ...)

Ejemplos:

.MODEL SMOD SW(RON=5M ROFF=10E9 VT=1.0 VH=0.1)

.MODEL SMOD VSWITCH(RON=5M ROFF=10E9 VON=1.1 VOFF=0.9)

.MODEL SMOD CSW(RON=5M ROFF=10E9 IT=0.5MA IH=0.5MA)

.MODEL SMOD ISWITCH(RON=5M ROFF=10E9 ION=1.0MA IOFF=0)

Las formas VSWITCH y ISWITCH de la muestra modelo son provistos para la compatibilidad con PSPICE.

El modelo del interruptor da interruptor apenas ideal descrito en Spice. El interruptor no es muy ideal, ya que la resistencia no puede cambiar de 0 a infinito, pero siempre debe tener un valor positivo finito. Por la selección correcta de la resistencia de encendido y apagado, pueden ser eficazmente cero e infinito en contraste para otros elementos del circuito.

Los parámetros disponibles son:

Nombre	Parámetro	Unidad	Defecto	Interruptor

VT	Voltaje umbral	Volts	0.0	S
VH	Voltaje de histeresis	Volts	0.0	S
VON	Voltaje umbral	Volts	0.0	S
VOFF	Voltaje umbral	Volts	0.0	S
IT	Corriente umbral	Amps	0.0	W
IH	Corriente de histeresis	Amps	0.0	W
ION	Corriente umbral	Amps	0.0	W
IOFF	Corriente umbral	Amps	0.0	W
RON	Resistencia encendido	OHM	1.0	Ambos
ROFF	Resistencia apagado	OHM	1/GMIN *	Ambos

*(Vea la línea de control .OPTIONS para una descripción de GMIN, el valor predeterminado en una resistencia de apagado es de 1.0e + 12 ohms.)

Si el voltaje controla el interruptor, el interruptor está ON si

$$V_{ctrl} > (VT + VH)$$

si VT y VH están definidos

$$V_{ctrl} > (VON)$$

Si VON está definido. Está en el estado OFF si

$$V_{ctrl} < (VT - VH)$$

si VT y VH están definidos

$$V_{ctrl} < (VOFF)$$

si VOFF esta definido.

Si la corriente controla el interruptor, el interruptor esta en ON si

$$I_{ctrl} > (IT + IH)$$

si IT y IH están definidos

$$I_{ctrl} > (ION)$$

si ION esta definido.

Y esta en OFF si

$$I_{ctrl} < (IT - IH)$$

si IT y IH están definidos

$$I_{ctrl} < (IOFF)$$

si IOFF esta definido.

El uso de un elemento ideal altamente no lineal como un interruptor puede causar discontinuidades grandes en los voltajes de nodo del circuito. Un cambio rápido puede causar roundoff numérico o problemas de tolerancia conduciendo a resultados erróneos o dificultades timestep. El usuario de interruptores puede mejorar la situación tomando los siguientes pasos:

Primero, es sabio colocar impedancias ideales al interruptor lo suficientemente altas o bajo para que sean insignificantes con relación a otros elementos del circuito. Usar impedancias del interruptor próximas a la "ideal" en todos los casos agrava el problema de discontinuidades mencionado arriba. Por supuesto, al modelar dispositivos reales como MOSFETs, la resistencia debería estar ajustada a un nivel realista dependiente del tamaño del dispositivo modelado.

Una amplia variedad de resistencias de encendido y apagado deben usarse en los interruptores (ROFF/RON > 1e + 12), luego la tolerancia en los errores permitidos durante el análisis transitorio debería ser disminuida usando la línea de control .OPTIONS y especificando a TRTOL a estar por debajo del valor predeterminado 7.0. Cuando los interruptores se sitúan alrededor de condensadores, la opción CHGTOL también debería acortarse. Los valores sugeridos para estas dos opciones son 1.0 y 1e-16 respectivamente. Estos cambios informan a Spice para ser más meticulosos alrededor de los puntos del interruptor para que no haya errores a causa del cambio rápido en el circuito.

4.2 Fuentes de voltaje y corriente

4.2.1 Ixxxx y Vxxxx: Fuentes independientes

Forma general:

VXXXXXXX N+ N- < DC/TRAN VALUE> >>

+ >> >>

IYYYYYYY N+ N- < DC/TRAN VALUE> >>

+ >> >>

Ejemplos:

VCC 10 0 DC 6

VIN 13 2 0.001 AC 1 SIN(0 1 1MEG)

ISRC 23 21 AC 0.333 45.0 SFFM(0 1 10K 5 1K)

VMEAS 12 9

VCARRIER 1 0 DISTOF1 0.1 -90.0

VMODULATOR 2 0 DISTOF2 0.01

IIN1 1 5 AC 1 DISTOF1 DISTOF2 0.001

N + y N- son los nodos positivo y negativo, respectivamente. Note que las fuentes de voltaje no necesitan ser puestas en tierra. Se da por hecho que corriente es positiva cuando fluye desde el nodo positivo, a través la fuente, hasta el nodo negativo. Una fuente de corriente de valor positivo obliga a la corriente a fluir desde el nodo N+, a través de la fuente, hacia el nodo N-. Las fuentes de voltaje, además de servir para excitar el circuito, son los ' amperímetros ' para Spice, el cero en las fuentes de voltaje pueden estar en el circuito con el objetivo de medir corriente. Por supuesto no tienen efecto en el funcionamiento del circuito.

DC/TRAN es el DC y valor transitorio de análisis de la fuente. Si el valor de la fuente es cero, entonces para DC y el análisis transitorio es cero, este valor puede omitirse. Si el valor de la fuente es invariante con el tiempo (ej. suministro de corriente), el valor puede opcionalmente ser precedido por las letras DC.

ACMAG es la magnitud AC y ACPHASE es la fase AC. La fuente toma ese valor en el análisis AC. Si ACMAG es omitido siguiendo la palabra clave AC, entonces se supone un valor unidad. Si ACPHASE es omitido, entonces se asume un valor cero. Si la fuente no es AC de pequeña señal de entrada, entonces la palabra clave AC y los valores AC se omiten.

DISTOF1 y DISTOF2 son las palabras claves que especifican que la fuente independiente tiene entradas de distorsión a frecuencia F1 y F2 respectivamente (vea la descripción de la línea de control .DISTO). Las palabras claves de magnitud (optativa) y fase pueden añadirse. Los valores predeterminados de la magnitud y la fase son 1.0 y 0.0 respectivamente.

A cualquier fuente independiente le puede ser asignado un valor dependiente del tiempo para el transitorio. Si a una fuente le es asignado un valor dependiente del tiempo, el valor de cero en el tiempo sirve para análisis DC. Hay cinco funciones independientes en la fuente: El pulso (PULSE), exponencial (EXP), sinusoidal (SIN), piece-wise linear(PWL), y frecuencia simple FM (SFFM). Si los parámetros fuera de los valores de la fuente se omiten o colocados a cero, entonces se suponen los valores predeterminados exteriormente. En las descripciones a continuación, TSTEP es el valor del incremento y TSTOP es el tiempo final (vea la línea de control .TRAN para la explicación - el capítulo 5.3.9).

Vea capítulos 10.26 y 10.13 para los parámetros que puede alterarse usando el comando 'alter' (vea a 6.9.3).

4.2.1.1 PULSE(): Pulso

Forma general:

PULSE(V1 V2 TD TR TF PW PER)

Ejemplos:

VIN 3 0 PULSE(-1 1 2NS 2NS 2NS 50NS 100NS)

Parámetro	Valor por defecto	Unidad
V1 (valor inicial)		Volts o Amps
V2 (valor del pulso)		Volts o Amps
TD (tiempo de retardo)	0.0	Segundos
TR (tiempo de subida)	TSTEP	Segundos
TF (tiempo de bajada)	TSTEP	Segundos
PW (Ancho del pulso)	TSTOP	Segundos
PER (periodo)	TSTOP	Segundos

Esta tabla define un pulso simple especificado:

Tiempo	Valor
0	V1
TD	V1
TD+TR	V2
TD+TR+PW	V2
TD+TR+PW+TF	V1
TSTOP	V1

Los puntos intermedios se determinan por interpolación lineal.

4.2.1.2 SIN(): Sinusoidal

Forma general:

SIN(VO VA FREQ TD THETA)

Ejemplos:

VIN 3 0 SIN(0 1 100MEG 1NS 1E10)

Parámetro	Valor por defecto	Unidad
VO (offset)		Volts o Amps
VA (amplitud)		Volts o Amps
FREQ (frecuencia)	1/TSTOP	Hz
TD (retardo)	0.0	Segundos
THETA (retardo)	0.0	1/segundos

La tabla siguiente describe la forma de onda:

Tiempo	Valor
0 a TD	VO
TD a TSTOP	$V0 + VAe^{-(t-TD)THETA}\sin(2piFREQ(t+TD))$

4.2.1.3 EXP(): Exponencial

Forma general:

EXP(V1 V2 TD1 TAU1 TD2 TAU2)

Ejemplos: VIN 3 0 EXP(-4 -1 2NS 30NS 60NS 40NS)

Parámetro	Valor por defecto	Unidad
V1 (valor inicial)		Volts o Amps
V2 (valor de pulso)		Volts o Amps
TD1 (tiempo de retardo de subida)	0.0	Segundos
TAU1 (constante de tiempo de subida)	TSTEP	Segundos
TD2 (tiempo de retardo de bajada)	TD1+TSTEP	Segundos

TAU2 (constante de tiempo de bajada)	TSTEP	Segundos

La tabla siguiente describe la forma de onda:

Tiempo	Valor
0 a TD1	V1
TD1 a TD2	$V1 + (V2 - V1)(1 - e^{\frac{-(t-TD1)}{TAU1}})$
TD2 a TSTOP	$V1 + (V2 - V1)(1 - e^{\frac{-(t-TD1)}{TAU1}}) + (V1 - V2)(1 - e^{\frac{-(t-TD2)}{TAU2}})$

4.2.1.4 PWL(): Piece-Wise Linear

Forma general:

PWL(T1 V1)

Ejemplos:

VCLOCK 7 5 PWL(0 -7 10NS -7 11NS -3 17NS -3 18NS -7 50NS -7)

Cada par de valores (Ti, Vi) especifica el valor de la fuente Vi (en Volts o Amps) en el de tiempo = Ti. El valor de la fuente en valores intermedios de tiempo se resuelve usando interpolación lineal en los valores de entrada.

4.2.1.5 SFFM(): Frecuencia simple FM

Forma general:

SFFM(VO VA FC MDI FS)

Ejemplos:

V1 12 0 SFFM(0 1M 20K 5 1K)

Parámetro	Valor por defecto	Unidad
VO (offset)		Volts o Amps
VA (Amplitud)		Volts o Amps
FC (frecuencia de la portadora)	1/TSTOP	Hz
MDI (coeficiente de la modulación)		
FS (frecuencia de la señal)	1/TSTOP	Hz

La forma de onda esta descrita por la ecuación:

$$V(t) = V_0 + V_A sin(2piFCt + MDIsin(2piFSt))$$

4.2.2 Fuentes lineales dependientes

SPICE permite a los circuitos tener fuentes lineales dependientes caracterizadas por cualquiera de estas cuatro ecuaciones

i = g v v = e v i = f i v = h i

donde g, e, f, y h son constantes que representan la ganancia de transconductancia, de voltaje, ganancia actual, y transresistencia, respectivamente.

4.2.2.1 Gxxxx: Fuentes lineales de voltaje controladas por corriente

Forma general:

GXXXXXXX N+ N- NC+ NC- VALUE

Ejemplos:

G1 2 0 5 0 0.1MMHO

N + y N- son los nodos positivo y negativo, respectivamente. El flujo de corriente es del nodo positivo, a través la fuente, para el nodo negativo. NC+ y NC- son los nodos positivos y negativos de control, respectivamente. VALUE es la transconductancia (en Mohs).

4.2.2.2 Exxxx: Fuentes lineales de voltaje controladas por voltaje

Forma general:

EXXXXXXX N+ N- NC+ NC- VALUE

Ejemplos:

E1 2 3 14 1 2.0

N+ es el nodo positivo, y N- el nodo negativo. NC+ y NC- son los nodos positivos y negativos de control, respectivamente. VALUE es la ganancia de voltaje.

4.2.2.3 Fxxxx: Fuente lineal de corriente controlada por corriente

Forma general:

FXXXXXXX N+ N- VNAM VALUE

Ejemplos:

F1 13 5 VSENS 5

N + y N- son los nodos positivos y negativos, respectivamente. El flujo de corriente va del nodo positivo, a través la fuente, para el nodo negativo. VNAM es el nombre de una fuente de voltaje a través de la cual la corriente controlante fluye. La dirección del flujo de corriente controlante es del nodo positivo, a través de la fuente, para el nodo negativo de VNAM. VALUE es la ganancia de corriente.

4.2.2.4 Hxxxx: Fuente lineal de corriente controlada por voltaje

Forma general:

HXXXXXXX N+ N- VNAM VALUE

Ejemplos:

HX 5 17 VZ 0.5K

N + y N- son los nodos positivos y negativos, respectivamente. VNAM es el nombre de una fuente de voltaje a través de la cual la corriente controlante fluye. La dirección del flujo de corriente controlante es del nodo positivo, a través de la fuente, para el nodo negativo de VNAM. VALUE es la transresistencia (en ohms).

4.2.3 Fuentes dependientes no lineales usando POLY()

Para la compatibilidad con SPICE2, WinSpice permite a los circuitos contener fuentes dependientes caracterizadas por cualquiera de las cuatro ecuaciones

i=f(v) v=f(v) i=f(i) v=f(i)

donde las funciones deben ser polinomios, y los argumentos pueden ser multidimensionales. Las funciones de polinomio se especifican por un conjunto de coeficientes p0, p1, ..., pn. Tanto el número de dimensiones y el número de coeficientes son arbitrarios. El significado de los coeficientes depende de la dimensión del polinomio, como se muestra en los siguientes ejemplos:

Supongo que la función es de una sola dimensión (esto es, una función de un argumento). Con lo que fv (valor de la función) se determina con la siguiente expresión en fa (el argumento de función):

fv = p0 + (p1*fa) + (p2*fa**2) + (p3*fa**3) + (p4*fa**4) + (p5*fa**5) + ...

Suponga ahora que la función es de dos dimensiones, con argumentos de fa y fb. Luego el fv (valor de función) se determina por la siguiente expresión:

fv = p0 + (p1*fa) + (p2*fb) + (p3*fa**2) + (p4*fa*fb)

+ (p5*fb**2)

+ (p6*fa**3) + (p7*fa**2*fb) + (p8*fa*fb**2)

+ (p9*fb**3) + ...

Considere ahora el caso de una función polinómica tridimensional con argumentos fa, fb, y fc. Con lo que la fv (valor de función) se determina por la siguiente expresión:

fv = p0 + (p1*fa) + (p2*fb) + (p3*fc) + (p4*fa**2)

+ (p5*fa*fb)

+ (p6*fa*fc) + (p7*fb**2) + (p8*fb*fc) + (p9*fc**2)

+ (p10*fa**3)

+ (p11*fa**2*fb) + (p12*fa**2*fc) + (p13*fa*fb**2)

+ (p14*fa*fb*fc)

+ (p15*fa*fc**2) + (p16*fb**3) + (p17*fb**2*fc)

+ (p18*fb*fc**2)

+ (p19*fc**3) + (p20*fa**4) + ...

Nota: Si el polinomio es de una sola dimensión y se especifica un coeficiente, entonces SPICE se hace cargo de él siendo p1 (y p0 = 0.0), para facilitar la entrada de fuentes controladas lineales.

Para las cuatro de fuentes dependientes descritas más adelantes, el parámetro de condición inicial es optativo. En caso de que no se especifique, WinSpice asume 0 como condición inicial para las fuentes dependientes es una 'suposición' inicial para el valor de la variable controlante. El programa usa esta condición inicial para obtener el punto de operación DC del circuito. Después de que la convergencia haya sido obtenida, el programa continúa iterando para obtener el valor exacto para la variable controlante. Por lo tanto, reduciendo el esfuerzo computacional para el punto de operación DC, o si el polinomio especifica una no linealidad fuerte, entonces debe especificarse un valor medianamente cerca de la variable controlante real para la condición inicial.

4.2.3.1 Voltage-Controlled Current Sources

Forma general:

GXXXXXXX N+ N- NC1+ NC1- ... P0

Ejemplos:

G1 1 0 5 3 0 0.1M

GR 17 3 17 3 0 1M 1.5M IC=2V

GMLT 23 17 POLY(2) 3 5 1 2 0 1M 17M 3.5U IC=2.5, 1.3

N+ and N- are the positive and negative nodes, respectively. Current flow is from the positive node, through the source, to the negative node. POLY(ND) only has to be specified if the source is multi-dimensional (one-dimensional is the default). If specified, ND is the number of dimensions, which must be positive. NC1+, NC1-, .are the positive and negative controlling nodes, respectively. One pair of nodes must be specified for each dimension. P0, P1, P2, ..., Pn are the polynomial coefficients. The (optional) initial condition is the initial guess at the value(s) of the controlling voltage(s). If not specified, 0.0 is assumed. The polynomial specifies the source current as a function of the controlling voltage(s). The second example above describes a current source with value

I = 1E-3*V(17,3) + 1.5E-3*V(17,3)**2

note that since the source nodes are the same as the controlling nodes, this source actually models a nonlinear resistor.

4.2.3.2 Fuentes de voltaje controlado por voltaje

Forma general:

EXXXXXXX N+ N- NC1+ NC1- ... P0

Ejemplos:

E1 3 4 21 17 10.5 2.1 1.75

EX 17 0 POLY(3) 13 0 15 0 17 0 0 1 1 1 IC=1.5,2.0,17.35

N + y N- son los nodos positivo y negativo, respectivamente. El flujo de corriente es del nodo positivo, a través la fuente, para el nodo negativo. POLY (ND) sólo tiene que estar especificado si la fuente es multidimensional (una dimensión por defecto). Estando ND especificado, que es el número de dimensiones, el cual debe ser positivo. NC1+, NC1-, son los nodos positivo y negativo controladores, respectivamente. Un par de nodos debe especificarse para cada dimensión.

P0, P1, P2, ..., Pn son los coeficientes polinómicos. La condición inicial (optativa) es la suposición inicial en el valor(es) del voltaje controlante(s). En caso de que no se especifiquen, se suponen 0.0.

El polinomio especifica la corriente de la fuente como una función del voltaje controlante(s). El segundo ejemplo de arriba describe una fuente de corriente con valor

V = V(13,0) + V(15,0) + V(17,0)

(En otras palabras, un voltaje ideal).

4.2.3.3 Fuentes de corriente controladas por corriente

Forma general:

FXXXXXXX N+ N- VN1 P0

Ejemplos:

F1 12 10 VCC 1MA 1.3M

FXFER 13 20 VSENS 0 1

N+ y N- son los nodos positivo y negativo, respectivamente. El flujo de corriente es del nodo positivo, a través la fuente, para el nodo negativo. POLY (ND) sólo tiene que estar especificado si la fuente es multidimensional (una dimensión por defecto). ND es el número de dimensiones, el cuál debe ser positivo. VN1, VN2, ... son los nombres de fuentes de voltaje a través de las cuales la corriente controlante fluye; Un nombre debe especificarse para cada dimensión. La dirección del flujo de corriente es del nodo positivo, a través la fuente, para el nodo negativo de cada fuente de voltaje. P0, P1, P2, ..., Pn son los coeficientes polinómicos. La condición inicial (optativa) es la suposición inicial en el valor(es) de la corriente controlante(s) (en Amps). En caso de que no lo especifiquemos, se asume un valor 0.0. El polinomio especifica la fuente actual como una función de la corriente controlante(s). El primer ejemplo de arriba describe una fuente en uso con valor

I = 1E-3 + 1.3E-3*I(VCC)

4.2.3.4 Fuentes de corriente controladas por voltaje

Forma general:

HXXXXXXX N+ N- VN1 P0

Ejemplos:

HXY 13 20 POLY(2) VIN1 VIN2 0 0 0 0 1 IC=0.5 1.3

HR 4 17 VX 0 0 1

N+ y N- son los nodos positivos y negativos, respectivamente. POLY (ND) sólo tiene que especificarse si la fuente es multidimensional (una dimensión por defecto). Estando ND especificado, que es el número de dimensiones, el cuál debe ser el positivo. VN1, VN2, ... so los nombres de fuentes de voltaje a través de las cuales la corriente controlante fluye; Un nombre debe especificarse para cada dimensión. La dirección del flujo de corriente es del nodo positivo, a través la fuente, para el nodo negativo de cada fuente de voltaje. P0, P1, P2, ..., Pn son los coeficientes polinómicos. La condición inicial (optativa) es la suposición inicial en el valor(es) de la corriente controlante(s) (en Amps). En caso de que se especifiquen, se asume 0.0. El polinomio especifica el voltaje de la fuente como una función de la corriente controlante(s). El primer ejemplo de arriba describe una fuente de voltaje con valor

V = I(VIN1)*I(VIN2)

4.2.4 Bxxxx: Fuentes no lineales dependientes

Forma general:

BXXXXXXX N+ N-

Ejemplos:

B1 0 1 I=cos(v(1))+sin(v(2))

B1 0 1 V=ln(cos(log(v(1,2)^2)))-v(3)^4+v(2)^v(1)

B1 3 4 I=17

B1 3 4 V=exp(pi^i(vdd))

N+ es el nodo positivo, y N- es el nodo negativo. Los valores de V y los parámetros I determinan los voltajes y corrientes a través del dispositivo, respectivamente. Si I es dado, además el dispositivo es una fuente de corriente, y si la V se da el dispositivo es una fuente de voltaje. Solo uno de estos parámetros debe ser dado.

El comportamiento en pequeña señal AC de la fuente no lineal es una fuente dependiente lineal (o las fuentes) con una constante de proporcionalidad igual para la derivada (o las derivadas) de la fuente en el punto de operación DC.

Las expresiones dadas para V e I pueden ser cualquier función de voltajes y corrientes a través de fuentes de voltaje en el sistema. En un análisis AC, la componente DC de voltaje o corriente de la fuente solo se usa cuando el punto de operación inicial se calcula. Las siguientes funciones de variables reales están definidas:

abs	Asinh	cosh	sin
acos	Atan	exp	sinh
acosh	Atanh	ln	sqrt
asin	Cos	log	tan

La función "u" es la unidad de paso, con un valor de uno para argumentos mayores que uno y cero para argumentos menores que cero. La función "uramp" es la integral de la unidad de paso: Para una entrada x, el valor es cero si la x es menor que cero, o si la x es mayor que el cero el valor es x.

Estas dos funciones son útiles para sintetizar funciones piece-wise no lineales, aunque la convergencia puede afectarse negativamente.

Los siguientes operadores estándar están definidos:

+ - * / ^ unitario -

Si el argumento de log, ln, o sqrt se vuelve menor que cero, entonces se usa el valor absoluto del argumento. Si un divisor se vuelve cero o la discusión de log o ln se vuelve cero, entonces un error resultará. Otros problemas pueden ocurrir cuando una función de una derivada parcial entra en una región donde esa función no está indefinida.

Para obtener tiempo en la expresión usted puede integrar la corriente de una fuente constante de corriente con un condensador y usar el voltaje resultante (no se olvide de ajustar el voltaje inicial a través del condensador). Resistencias no lineales, condensadores, e inductores pueden ser sintetizados con la fuente dependiente no lineal. Las resistencias no lineales son evidentes. Los condensadores no lineales y los inductores son implementados con sus equivalentes lineales por un cambio de variables

implementadas con la fuente dependiente no lineal. El siguiente subcircuito implementará un condensador no lineal:

```
.Subckt nlcap pos neg

* Bx: calculate f(input voltage)

Bx 1 0 v = f(v(pos,neg))

* Cx: linear capacitance

Cx 2 0 1

* Vx: Ammeter to measure current into the capacitor

Vx 2 1 DC 0Volts

* Drive the current through Cx back into the circuit

Fx pos neg Vx 1

.ends
```

Los inductores no lineales son similares.

Apéndice

Las siguientes tablas resumen los parámetros disponibles para cada uno de los dispositivos y modelos. Hay varias tablas para cada tipo de dispositivo soportado por Spice.

Los parámetros de entrada de los modelos son parámetros que pueden aparecer en una línea de definición de una dispositivo o modelo de la forma "keyword=value" donde "keyword" es el nombre del parámetro dado en las tablas. Por defecto los parámetros de entrada (como la resistencia de un resistor o la capacidad de un condensador) no necesitan una "keyword" específica.

Los parámetros de salida son parámetros que están disponibles para la salida del punto de funcionamiento y como información de depuración. Hay dos tipos de parámetros:

· Parámetros de dispositivo

Estos son parámetros que son calculados para cada dispositivo

· Parámetros de modelo

Estos son parámetros que son calculados para un modelo especifico y son compartidos por todos los dispositivos que describen ese modelo.

Los parámetros de dispositivo son especificados como "@device[keyword]" y están disponibles para el ultimo punto calculado o, si se especifica en una declaración ".save" , para una simulación completa como un vector de salida. Por tanto, para monitorizar la capacidad fuente-puerta de un MOSFET, el comando

save @m1[cgs]

antes de una simulación de régimen transitorio provoca que el valor de la capacidad sea salvado cada cierto tiempo, y escribiendo después un comando como

plot @m1[cgs]

se consigue la representación deseada (nótese que el comando "show" no sigue este formato).

Los parámetros de modelo se especifican como "@model[keyword]" y están disponibles para el ultimo punto calculado o, si se especifica en una declaración ".save" , para una simulación completa como un vector de salida normal como en el caso de los parámetros de dispositivo.

Otro ejemplo que muestra el uso de los parámetros de dispositivo y modelo para un BJT es el siguiente:

.model BC107 NPN(Is=1.527f Xti=3 Eg=1.11 Vaf=106.8 Bf=334.5 Ne=1.642 + Ise=222f Ikf=.1596 Xtb=1.5 Br=.788 Nc=2 Isc=0 Ikr=0 Re=.6 Rc=0.25 + Cjc=6.072p Mjc=.3333 Vjc=.75 Fc=.5 Cje=10.67p Mje=.3333 Vje=.75 + Tr=10n Tf=471.8p Itf=0 Vtf=0 Xtf=0) . . . * Declare two BC107 instances in a circuit Q1 22 24 25 BC107 Q2 42 44 45 BC107 . . . * Save internal base resistance (model parameter) * and the base-emitter voltage for q1 and q2. save @bc107[rb], @q1[vbe], @q2[vbe] . . . * Perform an analysis op . . . print @bc107[rb], @q1[vbe], @q2[vbe]

Algunas variables aparecen como de entrada y salida a la vez, y su salida simplemente devuelve el valor previo de la entrada, o el valor por defecto después de la simulación. Algunos parámetros son solo de entrada, porque la salida del sistema no puede manejar variables del tipo dado todavía, o por que la necesidad de ellas como variables de salida no estaba clara. Muchas de las variables de entrada están disponibles como variables de salida en un formato diferente, tales como los vectores de condiciones iniciales, que pueden ser recuperados como valores individuales de condición inicial. Por último, valores obtenidos internamente son sólo de salida y son proporcionados para depuración y salida del punto de funcionamiento.

Nótese que estas tablas no contienen una información detallada de los parámetros, que puede encontrarse en la sección dedicada a cada uno de los modelos y dispositivos, pero pueden ser usadas a modo de guía rápida.

B.1 URC: Línea R.C. Uniforme

URC - instance parameters (input-output)	
	Length of transmission line (longitud de la línea de transmisión)
n	Number of lumps (número de trozos)

URC - instance parameters (output-only)	
pos_node	Positive node of URC (nodo positivo del URC)
neg_node	Negative node of URC (nodo negativo del URC)
gnd	Ground node of URC (nodo de tierra o masa del URC)

URC - model parameters (input-only)	
urc	Uniform R.C. line model (modelo de URC)

URC - model parameters (input-output)	
k	Propagation constant of interest (constante de propagación)
fmax	Maximum frequency (frecuencia máxima)
rperl	Resistance per unit length (resistencia por unidad de longitud)
cperl	Capacitance per unit length (capacitancia por unidad de longitud)
isperl	Saturation current per length (corriente de saturación por longitud)
rsperl	Diode resistance per length (resistencia del diodo por longitud)

B.2 ASRC: Fuente arbitraria

ASRC - instance parameters (input-only)	

i	Current source (fuente de corriente)
v	Voltage source (fuente de tensión)

ASRC - instance parameters (output-only)	
i	Current through source (corriente a través de la fuente)
v	Voltage across source (tensión en la fuente)
pos_node	Positive Node (nodo positivo)
neg_node	Negative Node (nodo negativo)

B.3 BJT: Transistor bipolar de unión

BJT - instance parameters (input-only)	
ic	Initial condition vector (vbe, vce) (vector de condiciones iniciales)

BJT - instance parameters (input-output)	
off	Device initially off (dispositivo inicialmente apagado)
icvbe	Initial B-E voltage (tensión B-E inicial)
icvce	Initial C-E voltage (tensión C-E inicial)
area	Area factor (factor de área)
temp	Instance temperature (temperatura de dispositivo)

BJT - instance parameters (output-only)	
colnode	Number of collector node (número del nodo de colector)
basenode	Number of base node (número del nodo de base)
emitnode	Number of emitter node (número del nodo de emisor)
substnode	Number of substrate node (número del nodo de sustrato)
colprimenode	Internal collector node (nodo interno del colector)
baseprimenode	Internal base node (nodo interno de la base)
emitprimenode	Internal emitter node (nodo interno del emisor)
ic	Current at collector node (corriente en el nodo de colector)
ib	Current at base node (corriente en el nodo de base)
ie	Emitter current (corriente de emisor)
is	Substrate current (corriente de sustrato)
vbe	B-E voltage (tensión base-emisor)
vbc	B-C voltage (tensión base-colector)
gm	Small signal transconductance (transconductancia en pequeña señal)

gpi	Small signal input conductance -pi (conductancia de entrada en pequeña señal -pi)
gmu	Small signal conductance - mu (conductancia en pequeña señal -mu)
gx	Conductance from base to internal base (conductancia entre base y base interna)
go	Small signal output conductance (conductancia de salida en pequeña señal)
geqcb	d(Ibe)/d(Vbc)
gccs	Internal C-S cap. equiv. cond.
geqbx	Internal C-B-base cap. equiv. cond.
cpi	Internal base to emitter capacitance (capacitancia interna base-emisor)
cmu	Internal base to collector capacitance (capacitancia interna base-colector)
cbx	Base to collector capacitance (capacitancia base-colector)
ccs	Collector to substrate capacitance (capacitancia entre colector y sustrato)
cqbe	Cap. due to charge storage in B-E jct. (capacitancia debida al almacenamiento de carga en la unión B-E)
cqbc	Cap. due to charge storage in B-C jct.(capacitancia debida al almacenamiento de carga en la unión B-C)
cqcs	Cap. due to charge storage in C-S jct. (capacitancia debida al almacenamiento de carga en la unión C-S)
cqbx	Cap. due to charge storage in B-X jct. (capacitancia debida al almacenamiento de carga en la unión B-X)
cexbc	Total Capacitance in B-X junction (capacitancia total en la unión B-X)
qbe	Charge storage B-E junction (almacenamiento de carga en la unión B-E
qbc	Charge storage B-C junction (almacenamiento de carga en la unión B-E)
qcs	Charge storage C-S junction (almacenamiento de carga en la unión C-S)
qbx	Charge storage B-X junction (almacenamiento de carga en la unión B-X)
p	Power dissipation (disipación de potencia)

BJT - model parameters (input-output)	
npn	NPN type device (dispositivo de tipo NPN)
pnp	PNP type device (dispositivo de tipo PNP)
is	Saturation Current (corriente de saturación)
bf	Ideal forward beta (beta ideal en directa)
nf	Forward emission coefficient (coeficiente de emisión en directa)
vaf	Forward Early voltage (tensión Early en directa)
va	Forward Early voltage (same as vaf) (igual que la anterior)

ikf	Forward beta roll-off corner current (codo de caida de corriente para la beta en directa)
ik	Forward beta roll-off corner current (same as ikf) (igual que la anterior)
ise	B-E leakage saturation current (corriente de fuga en saturación B-E)
ne	B-E leakage emission coefficient (coeficiente de emisión de fuga B-E)
br	Ideal reverse beta (beta ideal en inversa)
nr	Reverse emission coefficient (coeficiente de emisión en inversa)
var	Reverse Early voltage (tensión Early en inversa)
vb	Reverse Early voltage (same as var) (igual que la anterior)
ikr	reverse beta roll-off corner current (codo de caida de corriente para la beta en inversa)
isc	B-C leakage saturation current (corriente de fuga en saturación B-C)
nc	B-C leakage emission coefficient (coeficiente de emisión de fuga B-C)
rb	Zero bias base resistanse (resistencia de base tendente a cero)
irb	Current for base resistance=(rb+rbm)/2 (corriente por la resistencia de base)
rbm	Minimum base resistance (resistencia minima de base)
re	Emitter resistance (resistencia de emisor)
rc	Collector resistance (resistencia de colector)
cje	Zero bias B-E depletion capacitance (capacitancia de deplexión B-E tendente a cero)
vje	B-E built in potential (potencial incorporado B-E)
pe	B-E built in potential (same as vje) (igual que la anterior)
mje	B-E junction grading coefficient (coeficiente de graduación de la unión B-E)
me	B-E junction grading coefficient (same as mje) (igual que la anterior)
tf	Ideal forward transit time (tiempo ideal de transición en directa)
xtf	Coefficient for bias dependence of TF (coeficiente de dependencia de TF con el punto de polarización)
vtf	Voltage giving VBC dependence of TF (tensión dada en VCB dependiente de TF)
ptf	Excess phase (exceso de fase)
cjc	Zero bias B-C depletion capacitance (capacitancia de deplexión en B-C tendente a cero)
vjc	B-C built in potential (potencial incorporado B-C)
mjc	B-C junction grading coefficient (coeficiente de graduación de la unión B-C)
mc	B-C junction grading coefficient (same as mjc)(igual

	que la anterior)
xcjc	Fraction of B-C cap to internal base (fracción de la capacidad BC a la base interna)
tr	Ideal reverse transit time (tiempo de tránsito ideal en inversa)
cjs	Zero bias C-S capacitance (capacitancia CS para polarización nula)
ccs	Zero bias C-S capacitance (capacitancia CS para polarización nula)
vjs	Substrate junction built in potential (potencial incorporado en la unión sustrato)
ps	Substrate junction built in potential (same as vjs)(igual que la anterior)
mjs	Substrate junction grading coefficient (coeficiente de graduación en la unión sustrato)
ms	Substrate junction grading coefficient (same as mjs)(igual que la anterior)
xtb	Forward and reverse beta temp. exp. (exponente para la dependencia térmica de las betas en directa y en inversa)
eg	Energy gap for IS temp. dependency (dependencia de la energía de huecos para IS temp..)
ms	Substrate junction grading coefficient (same as mjs)(igual que la anterior)
xti	Temp. exponent for IS
fc	Forward bias junction fit parameter (parámetro de ajuste de la unión para polarización directa)
tnom	Parameter measurement temperature (parámetro de medida de temperatura)
kf	Flicker Noise Coefficient (coeficiente de ruido Flicker)
af	Flicker Noise Exponent (exponente de ruido Flicker)

BJT - model parameters (output-only)	
type	NPN or PNP
invearlyvoltf	Inverse early voltage:forward (tensión inversa de Early: directa)
invearlyvoltr	Inverse early voltage:reverse (tensión inversa de Early: inversa)
invrollofff	Inverse roll off - forward (Caida inversa-directa)
invrolloffr	Inverse roll off - reverse (Caida inversa-directa)
collectorconduct	Collector conductance (conductancia del colector)
emitterconduct	Emitter conductance (conductancia del emisor)
transtimevbcfact	Transit time VBC factor (factor del tiempo de transición VCB)
excessphasefactor	Excess phase fact. (factor de exceso de fase)

B.4 BSIM1: Modelo de IGFET Berkeley de canal corto

BSIM1 - instance parameters (input-only)	
ic	Vector of DS,GS,BS initial voltages (vector de las tensiones iniciales de DS,GS,BS)

BSIM1 - instance parameters (input-output)	
l	Length (longitud)
w	Width (anchura)
ad	Drain area (área de drenador)
as	Source área (área de fuente)
pd	Drain perimeter (perímetro de drenador)
ps	Source perimeter (perímetro de fuente)
nrd	Number of squares in drain (número de cuadrados en el drenador)
nrs	Number of squares in source (número de cuadrados en la fuente)
off	Device is initially off (dispositivo inicialmente apagado)
vds	Initial D-S voltage (tensión inicial D-S)
vgs	Initial G-S voltage (tensión inicial G-S)
vbs	Initial B-S voltage (tensión inicial B-S)

BSIM1 - model parameters (input-only)	
nmos	Flag to indicate NMOS (bandera para indicar NMOS)
pmos	Flag to indicate PMOS (bandera para indicar PMOS)

BSIM1 - model parameters (input-output)	
vfb	Flat band voltage (tensión de banda plana)
lvfb	Length dependence of vfb (dependencia de la longitud de vfb)
wvfb	Width dependence of vfb (dependencia de la anchura de vfb)
phi	Strong inversión surface potential (potencial de superficie en fuerte inversión)
lphi	Length dependence of phi (dependencia de la longitud de phi)
wphi	Width dependence of phi (dependencia de la anchura de phi)
k1	Bulk effect coefficient 1 (coeficiente 1 del efecto Bulk)
lk1	Length dependence of k1 (dependencia de la longitud de k1)
wk1	Width dependence of k1 (dependencia de la anchura de k1)
k2	Bulk effect coefficient 2 (coeficiente 2 del efecto Bulk)

lk2	Length dependence of k2 (dependencia de la longitud de k2)
wk2	Width dependence of k2 (dependencia de la anchura de k2)
eta	VDS dependence of threshold voltage (dependencia de VDS de la tensión umbral)
leta	Length dependence of eta (dependencia de la longitud de eta)
weta	Width dependence of eta (dependencia de la anchura de eta)
x2e	VBS dependence of eta (dependencia de VBS de eta)
lx2e	Length dependence of x2e (dependencia de la longitud de x2e)
wx2e	Width dependence of x2e (dependencia de la anchura de x2e)
x3e	VDS dependence of eta (dependencia de VDS de eta)
lx3e	Length dependence of x3e (dependencia de la longitud de x3e)
wx3e	Width dependence of x3e (dependencia de la anchura de x3e)
dl	Channel length reduction in um (reducción de la longitud del canal)
dw	Channel width reduction in um (reducción de la anchura del canal)
muz	Zero field mobility at VDS=0 VGS=VTH (movilidad de campo cero)
x2mz	VBS dependence of muz (dependencia de VBS de muz)
lx2mz	Length dependence of x2mz (dependencia de la longitud de x2mz)
wx2mz	Width dependence of x2mz (dependencia de la anchura de x2mz)
mus	Mobility at VDS=VDD VGS=VTH, channel length modulation (movilidad con VDS=VDD VGS=VTH, modulacion de longitud del canal)
lmus	Length dependence of mus (dependencia de la longitud de mus)
wmus	Width dependence of mus (dependencia de la ancura de mus)
x2ms	VBS dependence of mus (dependencia de VBS de mus)
lx2ms	Length dependence of x2ms (dependencia de la longitud de x2ms)
wx2ms	Width dependence of x2ms (dependencia de la anchura de x2ms)
x3ms	VDS dependence of mus (dependencia de VDSde mus)
lx3ms	Length dependence of x3ms (dependencia de la longitud de x3ms)
wx3ms	Width dependence of x3ms (dependencia de la anchura de x3ms)

u0	VGS dependence of mobility (dependencia de VGS de la movilidad)
lu0	Length dependence of u0 (dependencia de la longitud de u0)
wu0	Width dependence of u0 (dependencia de la anchura de u0)
x2u0	VBS dependence of u0 (dependencia de la VBS de u0)
lx2u0	Length dependence of x2u0 (dependencia de la longitud de x2u0)
wx2u0	Width dependence of x2u0 (dependencia de la anchura de x2u0)
u1	VDS dependence of mobility, velocity saturation (dependencia de la VDS de la movilidad, velocidad de saturación)
lu1	Length dependence of u1 (dependencia de la longitud de u1)
wu1	Width dependence of u1 (dependencia de la anchura de u1)
x2u1	VBS dependence of u1 (dependencia de la VBS de u1)
lx2u0	Length dependence of x2u1 (dependencia de la longitud de x2u1)
wx2u1	Width dependence of x2u1 (dependencia de la anchura de x2u1)
x3u1	VDS dependence of u1 (dependencia de la VDS de u1)
lx3u1	Length dependence of x3u1 (dependencia de la longitud de x3u1)
wx3u1	Width dependence of x3u1 (dependencia de la anchura de x3u1)
n0	Sub threshold slope (pendiente sub-umbral)
ln0	Length dependence of n0 (dependencia de la longitud de n0)
wn0	Width dependence of n0 (dependencia de la anchura de n0)
nb	VBS dependence of sub threshold slope (dependencia de la VBS de la pendiente sub-umbral)
lnb	Length dependence of nb (dependencia de la longitud de nb)
wnb	Width dependence of nb (dependencia de la anchura de nb)
nd	VDS dependence of sub threshold slope (dependencia de la VDS de la pendiente sub-umbral)
lnd	Length dependence of nd (dependencia de la longitud de nd)
wnd	Width dependence of nd (dependencia de la anchura de nd)
tox	Gate oxide thickness in um (grosor del óxido de puerta en um)
temp	Temperature in degree Celsius (temperatura en grados Celsius)
vdd	Supply voltage to specify mus (tensión de

	alimentacion)
cgso	Gate source overlap capacitance per unit channel width(m) (capacidad de solapamiento puerta-fuente por unidad de anchura del canal)
cgdo	Gate drain overlap capacitance per unit channel width(m) (capacidad de solapamiento puerta-drenador por unidad de anchura del canal)
cgbo	Gate bulk overlap capacitance per unit channel length(m) (capacidad de solapamiento puerta-sustrato por unidad de anchura del canal)
xpart	Flag for channel charge partitioning (bandera para particionamiento de carga del canal)
rxs	Source drain diffusion sheet resistance in ohm per square (resistencia de difusión fuente-drenador en ohmios por cuadro)
js	Source drain junction saturation current per unit área (corriente de saturación en la unión fuente-drenador por unidad de área)
pb	Source drain junction built in potential (potencial incorporado en la unión fuente-drenador)
mj	Source drain bottom junction capacitance grading coefficient (coeficiente de gradación de capacitancia en la unión fuente-drenador)
pbsw	Source drain side junction capacitance built in potential (potencial incorporado en la cara de la unión fuente drenador)
mjsw	Source drain side junction capacitance grading coefficient (coeficiente de gradación de capacitancia en la cara de la unión fuente-drenador)
cj	Source drain bottom junction capacitance per unit area (capacitancia de la parte de abajo de la unión fuente-drenador por unidad de área)
cjsw	Source drain side junction capacitance per unit area (capacitancia en la cara de la unión fuente-drenador por unidad de área)
wdf	Default width of source drain diffusion in um (anchura de la difusión fuente-drenador por defecto en um)
dell	Length reduction of source drain diffusion (reducción de longitud de la difusión fuente-drenador)

B.5 BSIM2: Modelo de IGFET Berkeley de canal corto

BSIM2 - instance parameters (input-only)	
ic	Vector of DS,GS,BS initial voltages (vector de las tensiones iniciales de DS,GS,BS)

BSIM2 - instance parameters (input-output)	
l	Length (longitud)
w	Width (anchura)

ad	Drain area (área de drenador)
as	Source área (área de fuente)
pd	Drain perimeter (perímetro de drenador)
ps	Source perimeter (perímetro de fuente)
nrd	Number of squares in drain (número de cuadrados en el drenador)
nrs	Number of squares in source (número de cuadrados en la fuente)
off	Device is initially off (dispositivo inicialmente apagado)
vds	Initial D-S voltage (tensión inicial D-S)
vgs	Initial G-S voltage (tensión inicial G-S)
vbs	Initial B-S voltage (tensión inicial B-S)

BSIM2 - model parameters (input-only)	
nmos	Flag to indicate NMOS (bandera para indicar NMOS)
pmos	Flag to indicate PMOS (bandera para indicar PMOS)

BSIM2 - model parameters (input-output)	
vfb	Flat band voltage (tensión de banda plana)
lvfb	Length dependence of vfb (dependencia de la longitud de vfb)
wvfb	Width dependence of vfb (dependencia de la anchura de vfb)
phi	Strong inversión surface potential (potencial de superficie en fuerte inversión)
lphi	Length dependence of phi (dependencia de la longitud de phi)
wphi	Width dependence of phi (dependencia de la anchura de phi)
kl	Bulk effect coefficient 1 (coeficiente 1 del efecto Bulk)
lkl	Length dependence of k1 (dependencia de la longitud de k1)
wkl	Width dependence of k1 (dependencia de la anchura de k1)
k2	Bulk effect coefficient 2 (coeficiente 2 del efecto Bulk)
lk2	Length dependence of k2 (dependencia de la longitud de k2)
wk2	Width dependence of k2 (dependencia de la anchura de k2)
eta0	VDS dependence of threshold voltage at VDD=0 (dependencia de VDS de la tensión umbral con VDD=0)
leta0	Length dependence of eta0 (dependencia de la longitud de eta0)
weta0	Width dependence of eta0 (dependencia de la

	anchura de eta0)
etab	VBS dependence of eta (dependencia de la VBS de eta)
letab	Length dependence of etab (dependencia de la longitud de eta0)
wetab	Width dependence of etab (dependencia de la anchura de etab)
dl	Channel length reduction in um (reducción de la longitud de canal en um)
dw	Channel width reduction in um (reducción de la anchura de canal en um)
mu0	Low-field mobility, at VDS=0 VGS=VTH (movilidad de bajo campo con VDS=0 VGS=VTH)
mu0b	VBS dependence of low-field mobility (dependencia de la VBS de la movilidad de bajo campo)
lmu0b	Length dependence of mu0b (dependencia de la longitud de mu0b)
wmu0b	Width dependence of mu0b (dependencia de la anchura de mu0b)
mus0	Mobility at VDS=VDD VGS=VTH (movilidad con VDS=VDD VGS=VTH)
lmus0	Length dependence of mus0 (dependencia de la longitud de mus0)
wmus0	Width dependence of mus0 (dependencia de la anchura de mus0)
musb	VBS dependence of mus (dependencia de la VBS de mus)
lmusb	Length dependence of musb (dependencia de la longitud de musb)
wmusb	Width dependence of musb (dependencia de la anchura de musb)
mu20	VDS dependence of mu in tanh term (dependencia de la VDS de mu en terminos de tanh)
lmu20	Length dependence of mu20 (dependencia de la longitud de mu20)
wmu20	Width dependence of mu20 (dependencia de la anchura de mu20)
mu2b	VBS dependence of mu2 (dependencia de la VBS de mu2)
lmu2b	Length dependence of mu2b (dependencia de la longitud de mu2b)
wmu2b	Width dependence of mu2b (dependencia de la anchura de mu2b)
mu2g	VGS dependence of mu2 (dependencia de la VGS de mu2)
lmu2g	Length dependence of mu2g (dependencia de la longitud de mu2g)
wmu2g	Width dependence of mu2g (dependencia de la anchura de mu2g)
mu30	VDS dependence of mu in linear term (dependencia de la VDS de mu en terminos lineales)

lmu30	Length dependence of mu30 (dependencia de la longitud de mu30)
wmu30	Width dependence of mu30 (dependencia de la anchura de mu30)
mu3b	VBS dependence of mu3 (dependencia de la VBS de mu3)
lmu3b	Length dependence of mu3b (dependencia de la longitud de mu3b)
wmu3b	Width dependence of mu3b (dependencia de la anchura de mu3b)
mu3g	VGS dependence of mu3 (dependencia de la VGS de mu3)
lmu3g	Length dependence of mu3g (dependencia de la longitud de mu3g)
wmu3g	Width dependence of mu3g (dependencia de la anchura de mu3g)
mu40	VDS dependence of mu in linear term (dependencia de la VDS de mu en terminos lineales)
lmu40	Length dependence of mu40 (dependencia de la longitud de mu40)
wmu40	Width dependence of mu40 (dependencia de la anchura de mu40)
mu4b	VBS dependence of mu4 (dependencia de la VBS de mu4)
lmu4b	Length dependence of mu4b (dependencia de la longitud de mu4b)
wmu4b	Width dependence of mu4b (dependencia de la anchura de mu4b)
mu4g	VGS dependence of mu4 (dependencia de la VGS de mu4)
lmu4g	Length dependence of mu4g (dependencia de la longitud de mu4g)
wmu4g	Width dependence of mu4g (dependencia de la anchura de mu4g)
ua0	Linear VGS dependence of mobility (dependencia de la VGS lineal de la movilidad)
lua0	Length dependence of ua0 (dependencia de la longitud de ua0)
wua0	Width dependence of ua0 (dependencia de la anchura de ua0)
uab	VBS dependence of ua (dependencia de la VBS de ua)
luab	Length dependence of uab (dependencia de la longitud de uab)
wuab	Width dependence of uab (dependencia de la anchura de uab)
ub0	Quadratic VGS dependence of mobility (dependencia de la VGS cuadratica de la movilidad)
lub0	Length dependence of ub0 (dependencia de la longitud de ub0)
wub0	Width dependence of ub0 (dependencia de la anchura de ub0)

ubb	VBS dependence of ub (dependencia de la VBS de ub)
lubb	Length dependence of ubb (dependencia de la longitud de ubb)
wubb	Width dependence of ubb (dependencia de la anchura de ubb)
u10	VDS dependence of mobility (dependencia de la VDS de la movilidad)
lu10	Length dependence of u10 (dependencia de la longitud de u10)
wu10	Width dependence of u10 (dependencia de la anchura de u10)
u1b	VBS dependence of u1 (dependencia de la VBS de u1)
lu1b	Length dependence of u1b (dependencia de la longitud de u1b)
wu1b	Width dependence of u1b (dependencia de la anchura de u1b)
u1d	VDS dependence of u1 (dependencia de la VDS de u1)
lu1d	Length dependence of u1d (dependencia de la longitud de u1d)
wu1d	Width dependence of u1d (dependencia de la anchura de u1d)
n0	Sub threshold slope at VDS=0 VBS=0 (pendiente sub-umbral con VDS=0 VBS=0)
ln0	Length dependence of n0 (dependencia de la longitud de n0)
wn0	Width dependence of n0 (dependencia de la anchura de n0)
nb	VBS dependence of nb (dependencia de la VBS de nb)
lnb	Length dependence of nb (dependencia de la longitud de nb)
wnb	Width dependence of nb (dependencia de la anchura de nb)
nd	VDS dependence of n (dependencia de la VDS de n)
lnd	Length dependence of nd (dependencia de la longitud de nd)
wnd	Width dependence of nd (dependencia de la anchura de nd)
vof0	Threshold voltage offset AT VDS=0 VBS=0 (offset de la tensión umbral con VDS=0 VBS=0)
lvof0	Length dependence of vof0 (dependencia de la longitud de vof0)
wvof0	Width dependence of vof0 (dependencia de la anchura de vof0)
vofb	VBS dependence of vof (dependencia de la VBS de vof)
lvofb	Length dependence of vofb (dependencia de la longitud de vofb)
wvofb	Width dependence of vofb (dependencia de la anchura de vofb)

vofd	VDS dependence of vof (dependencia de la VDS de vof)
lvofd	Length dependence of vofd (dependencia de la longitud de vofd)
wvofd	Width dependence of vofd (dependencia de la anchura de vofd)
ai0	Pre-factor of hot-electron effect (Pre-factor del efecto hot-electron)
lai0	Length dependence of ai0 (dependencia de la longitud de ai0)
wai0	Width dependence of ai0 (dependencia de la anchura de ai0)
aib	VBS dependence of ai (dependencia de la VBS de ai)
laib	Length dependence of aib (dependencia de la longitud de aib)
waib	Width dependence of aib (dependencia de la anchura de aib)
bi0	Exponential factor of hot-electron effect (factor exponencial del efecto hot-electron)
lbi0	Length dependence of bi0 (dependencia de la longitud de bi0)
wbi0	Width dependence of bi0 (dependencia de la anchura de bi0)
bib	VBS dependence of bi (dependencia de la VBS de bi)
lbib	Length dependence of bib (dependencia de la longitud de bib)
wbib	Width dependence of bib (dependencia de la anchura de bib)
vghigh	Upper bound of the cubic spline function (limite superior de la función cubic spline)
lvghigh	Length dependence of vghigh (dependencia de la longitud de vghigh)
wvghigh	Width dependence of vghigh (dependencia de la anchura de vghigh)
vglow	Lower bound of the cubic spline function (limite inferior de la función cubic spline)
lvglow	Length dependence of vglow (dependencia de la longitud de vglow)
wvglow	Width dependence of vglow (dependencia de la anchura de vglow)
tox	Gate oxide thickness in um (grosor del óxido de puerta en um)
temp	Temperature in degree Celsius (temperatura en grados Celsius)
vdd	Maximum Vds (maxima Vds)
vgg	Maximum Vgs (maxima Vgs)
vbb	Maximum Vbs (maxima Vbs)
cgso	Gate source overlap capacitance per unit channel width(m) (capacidad de solapamiento puerta-fuente por unidad de anchura del canal)

cgdo	Gate drain overlap capacitance per unit channel width(m) (capacidad de solapamiento puerta-drenador por unidad de anchura del canal)
cgbo	Gate bulk overlap capacitance per unit channel length(m) (capacidad de solapamiento puerta-sustrato por unidad de anchura del canal)
xpart	Flag for channel charge partitioning (bandera para particionamiento de carga del canal)
rxs	Source drain diffusion sheet resistance in ohm per square (resistencia de difusión fuente-drenador en ohmios por cuadro)
js	Source drain junction saturation current per unit área (corriente de saturación en la unión fuente-drenador por unidad de área)
pb	Source drain junction built in potential (potencial incorporado en la unión fuente-drenador)
mj	Source drain bottom junction capacitance grading coefficient (coeficiente de gradación de capacitancia en la unión fuente-drenador)
pbsw	Source drain side junction capacitance built in potential (potencial incorporado en la cara de la unión fuente drenador)
mjsw	Source drain side junction capacitance grading coefficient (coeficiente de gradación de capacitancia en la cara de la unión fuente-drenador)
cj	Source drain bottom junction capacitance per unit area (capacitancia de la parte de abajo de la unión fuente-drenador por unidad de área)
cjsw	Source drain side junction capacitance per unit area (capacitancia en la cara de la unión fuente-drenador por unidad de área)
wdf	Default width of source drain diffusion in um (anchura de la difusión fuente-drenador por defecto en um)
dell	Length reduction of source drain diffusion (reducción de longitud de la difusión fuente-drenador)

B.6 Capacitor: condensador fijo

Capacitor - instance parameters (input-output)	
capacitance	Device capacitance (capacitancia de dispositivo)
ic	Initial capacitor voltage (tensión inicial en el condensador)
w	Device width (anchura de dispositivo)
l	Device length (longitud de dispositivo)

Capacitor - instance parameters (output-only)	
i	Device current (corriente de dispositivo)
p	Instantaneous device power (potencia instantanea de

	dispositivo)

Capacitor - model parameters (input-only)	
c	Capacitor model (modelo de condensador)

Capacitor - model parameters (input-output)	
cj	Bottom Capacitance per area (capacitancia de la unión inferior por área)
cjsw	Sidewall capacitance per meter (capacitancia de la pared lateral por metro)
defw	Default width (anchura por defecto)
tc1	First order temp. coefficient (coeficiente de temperatura de primer orden)
tc2	Second order temp. coefficient (coeficiente de temperatura de segundo orden)
vc1	First order voltage coefficient (coeficiente de tensión de primer orden)
vc2	Second order voltage coefficient (coeficiente de tensión de segundo orden)
narrow	Width correction factor (factor de corrección de anchura)

B.7 CCCS: Fuente de corriente controlada por corriente

CCCS - instance parameters (input-output)	
gain	Gain of source (ganancia de fuente)
control	Name of controlling source (nombre de la fuente de control)

CCCS - instance parameters (output-only)	
neg_node	Negative node of source (nodo negativo de fuente)
pos_node	Positive node of source (nodo positivo de fuente)
i	CCCS output current (corriente de salida CCCS)
v	CCCS voltage at output (tensión de salida CCCS)
p	CCCS power (potencia CCCS)

B.8 CCVS: Fuente de corriente controlada por corriente lineal

CCVS - instance parameters (input-output)	
gain	Transresistance (gain) (ganancia
control	Controlling voltage source (fuente controlada por tensión)

CCVS - instance parameters (output-only)	
pos_node	Positive node of source (nodo positivo de fuente)
neg_node	Negative node of source (nodo negativo de fuente)
i	CCVS output current (corriente de salida CCVS)
v	CCVS output voltage (tensión de salida CCVS)
p	CCVS power (potencia CCVS)

B.9 CSwitch: conmutador ideal controlado por corriente

CSwitch - instance parameters (input-only)	
on	Initially closed (inicialmente cerrado)
off	Initially open (inicialmente abierto)

CSwitch - instance parameters (input-output)	
control	Name of controlling source (nombre de la fuente de control)

CSwitch - instance parameters (output-only)	
pos_node	Positive node of switch (nodo positivo del conmutador)
neg_node	Negative node of switch (nodo negativo del conmutador)
i	Switch current (corriente por el conmutador)
p	Instantaneous power (potencia instantanea)

CSwitch - model parameters (input-output)	
csw	Current controlled switch model (modelo de conmutador controlado por corriente)
it	Threshold current (corriente de umbral)
ih	Hysteresis current (corriente de histeresis)
ron	Closed resistance (resistencia cerrada)
roff	Open resistance (resistencia en abierto)
ion	Control current to switch on (corriente de control para conmutador en on)
ioff	Control current to switch off (corriente de control para conmutador en off)

CSwitch - model parameters (output-only)	
gon	Closed conductance (conductancia cerrada)
goff	Open conductance (conducatancia en abierto)

B.10 Diode: modelo del diodo de unión

Diode - instance parameters (input-output)	
off	Initially off (inicialmente en off)
temp	Instance temperature (temperatura de dispositivo)
ic	Initial device voltage (tensión inicial del dispositivo)
area	Area factor (factor de área)

Diode - instance parameters (output-only)	
vd	Diode voltage (tensión del diodo)
id	Diode current(corriente del diodo)
ic	Diode current (corriente del diodo)
gd	Diode conductance (conductancia del diodo)
cd	Diode capacitance (capacitancia del diodo)
charge	Diode capacitor charge (carga del condensador del diodo)
capcur	Diode capacitor current (corriente del condensador del diodo)
p	Diode power (potencia del diodo)

Diode - model parameters (input-only)	
d	Diode model (modelo del diodo)

Diode - model parameters (input-output)	
is	Saturation current (corriente de saturación)
tnom	Parameter measurement temperature (parámetro de la medida de temperatura)
rs	Ohmic resistance (resistencia ohmica)
n	Emission Coefficient (coeficiente de emisión)
tt	Transit Time (tiempo de tránsito)
cjo	Junction capacitance (capacitancia de la unión)
cj0	Junction capacitance (capacitancia de la unión)
vj	Junction potential (potencial de la unión)
m	Grading coefficient (coeficiente de gradación)
eg	Activation energy (energia de activacion)
xti	Saturation current temperature exp. (exp de temperatura de la corriente de saturación)
kf	flicker noise coefficient (coeficiente de ruido flicker)
af	flicker noise exponent (exponente de ruido flicker)
fc	Forward bias junction fit parameter (parámetro de ajuste de la unión para polarización directa)
bv	Reverse breakdown voltage (tensión de ruptura en inversa)

ibv	Current at reverse breakdown voltage (corriente a la tensión de ruptura en inverso)

Diode - model parameters (output-only)	
cond	Ohmic conductance (conductancia ohmica)

B.11 Inductor: Inductores

Inductor - instance parameters (input-output)	
inductance	Inductance of inductor (inductancia del inductor)
ic	Initial current through inductor (corriente inicial a traves del inductor)

Inductor - instance parameters (output-only)	
flux	Flux through inductor (flujo a traves del inductor)
v	Terminal voltage of inductor (tensión en terminal del inductor)
Volt	Terminal voltage of inductor. Same a 'v'. (igual que 'v')
i	Current through the inductor (corriente a traves del inductor)
current	Current through the inductor. Same as 'i'. (igual que 'i')
p	Instantaneous power dissipated by the inductor (potencia instantanea disipada por el inductor)

B.12 mutual: Inductores mutuos

mutual - instance parameters (input-output)	
k	Mutual inductance (inductancia mutua)
coefficient	(null)
inductor1	First coupled inductor (primario del inductor acoplado)
inductor2	Second coupled inductor(secundario del inductor acoplado)

B.13 Isource: Fuente de corriente independiente

Isource - instance parameters (input-only)	
pulse	Pulse description (descripción del pulso)
sine	Sinusoidal source description (descripción de la

	fuente seno)
sin	Sinusoidal source description (igual que la anterior)
exp	Exponential source description (descrip. Fuente exponencial)
pwl	Piecewise linear description(descrip. Lineal a trozos)
sffm	single freq. FM description (descrip. Unica frecuencia FM)
ac	AC magnitude,phase vector (magnitud AC, vector de fase)
c	Current through current source(corriente a traves de la fuente)
distof1	f1 input for distortion (entrada f1 para distorsión)
distof2	f2 input for distortion (entrada f2 para distorsión)

Isource - instance parameters (input-output)	
dc	DC value of source (valor en continua de la fuente)
acmag	AC magnitude (magnitud en alterna) . .
acphase	AC phase (fase en alterna)

Isource - instance parameters (output-only)	
neg_node	Negative node of source (nodo negativo de fuente)
pos_node	Positive node of source (nodo positivo de fuente)
acreal	AC real part (parte real de alterna)
acimag	AC imaginary part (parte imaginaria de alterna)
function	Function of the source (función de la fuente)
order	Order of the source function (orden de la función de fuente)
coeffs	Coefficients of the source (coeficientes de la fuente)
v	Voltage across the supply (tensión a traves de alimentacion)
p	Power supplied by the source(potencia aplicada por fuente)

B.14 JFET: Transistor de efecto de campo

JFET - instance parameters (input-output)	
off	Device initially off (dispositivo inicialmente en off)
ic	Initial VDS,VGS vector(VDS inicial, vector VGS)
area	Area factor (factor de área)
ic-vds	Initial D-S voltage (tensión inicial DS)
ic-vgs	Initial G-S voltage (tensión inicial GS)
temp	Instance temperature (temperatura de dispositivo)

JFET - instance parameters (output-only)	
drain-node	Number of drain node (número de nodo drenador)
gate-node	Number of gate node (número de nodo de puerta)
source-node	Number of source node (número de nodo de fuente)
drain-prime-node	Internal drain node (nodo interno de drenador)
source-prime-node	Internal source node (nodo interno de fuente)
vgs	Voltage G-S (tensión GS)
vgd	Voltage G-D (tensión GD)
ig	Current at gate node (corriente en nodo de puerta)
id	Current at drain node (corriente en nodo drenador)
is	Source current (corriente de fuente)
igd	Current G-D (corriente GD)
gm	Transconductance (transconductancia)
gds	Conductance D-S (conductancia DS)
ggs	Conductance G-S (conductancia GS)
ggd	Conductance G-D (conductancia GD)
qgs	Charge storage G-S junction (almacen. de carga en GS)
qgd	Charge storage G-D junction (almacen. de carga en GD)
cqgs	Capacitance due to charge storage G-S junction (capacitancia debida al almacenamiento de carga en GS)
cqgd	Capacitance due to charge storage G-D junction (capacitancia debida al almacenamiento de carga en GD)
p	Power dissipated by the JFET (potencia disipada por el JFET)

JFET - model parameters (input-output)	
njf	N type JFET model (modelo de JFET tipo N)
pjf	P type JFET model(modelo de JFET tipo P)
vt0	Threshold voltage (tensión umbral)
vto	Threshold voltage (tensión umbral)
beta	Transconductance parameter (parámetro transconductancia)
lambda	Channel length modulation param. (para. Modulacion de canal)
rd	Drain ohmic resistance (resistencia ohmica de drenador)
rs	Source ohmic resistance(resistencia ohmica de fuente)
cgs	G-S junction capacitance (capacitancia unión GS)
cgd	G-D junction cap (capacitancia unión GD)
pb	Gate junction potential(potencial de la unión de puerta)

is	Gate junction saturation current (corriente de saturación de la unión de puerta)
fc	Forward bias junction fit parm.(parámetro de ajuste de la unión para polarización directa)
b	Doping tail parameter(param. De la cola de dopado)
tnom	Parameter measurement temperature (parámetro de la temperatura de medida)
kf	Flicker Noise Coefficient (coeficiente de ruido flicker)
af	Flicker Noise Exponent (exponente de ruido flicker)

JFET - model parameters (output-only)	
type	N-type or P-type JFET model (modelo JFET tipo P o N)
gd	Drain conductance (conductancia de drenador)
gs	Source conductance (conductancia de fuente)

B.15 LTRA: Linea de transmision con perdidas

LTRA - instance parameters (input-only)	
ic	Initial condition vector:v1,i1,v2,i2 (vector de condiciones iniciales)

LTRA - instance parameters (input-only)	
v1	Initial voltage at end 1 (tensión inicial en el extremo 1)
v2	Initial voltage at end 2 (tensión inicial en el extremo 2)
i1	Initial current at end 1 (corriente inicial en el extremo 1)
i2	Initial current at end 1 (corriente inicial en el extremo 1)

LTRA - instance parameters (output-only)	
pos_node1	Positive node of end 1 of t-line (nodo positivo en el extremo 1 de la linea de transmision)
neg_node1	Negative node of end 1 of t-line (nodo negativo en el extremo 1 de la linea de transmision)
pos_node2	Positive node of end 2 of t-line (nodo positivo en el extremo 2 de la linea de transmision)
neg_node2	Negative node of end 2 of t-line (nodo negativo en el extremo 2 de la linea de transmision)

LTRA - model parameters (input-output)	
ltra	LTRA model (modelo de LTRA)

r	Resistance per metre (resistencia por metro)
l	Inductance per metre (inductancia por metro)
g	(null)
c	Capacitance per metre (capacitancia por metro)
len	Length of line (longitud de linea)
nocontrol	No timestep control (no controlado por paso de tiempo)
steplimit	Always limit timestep to 0.8*(delay of line) (limita siempre el paso de tiempo a 0.8, retardo de linea)
nosteplimit	Don't always limit timestep to 0.8*(delay of line) (no limita siempre el paso de tiempo a 0.8, retardo de linea)
lininterp	Use linear interpolation (usa interpolacion lineal)
quadinterp	Use quadratic interpolation (usa interpolacion cuadratica)
mixedinterp	Use linear interpolation if quadratic results look unacceptable (usa interpolacion lineal si la cuadratica no ofrece resultados aceptables)
truncnr	Use N-R iterations for step calculation in LTRAtrunc (usa N-R iteraciones por paso de calculo en LTRAtrunc)
truncdontcut	Don't limit timestep to keep impulse response calculation errors low (no limita los pasos de tiempo para mantener el calculo de la respuesta al impulso en valores de error bajos)
compactrel	Special reltol for straight line checking (tolerancia relativa especial para la comprobación de línea recta)
compactabs	Special abstol for straight line checking (tolerancia absoluta especial para la comprobación de línea recta)

LTRA - model parameters (output-only)	
rel	Rel. rate of change of deriv. for bkpt (ratio relativo de cambio de la derivada por bkpt)
abs	Abs. rate of change of deriv. for bkpt (ratio absoluto de cambio de la derivada por bkpt)

B.16 MES: Modelo de MESFET GaAs

MES - instance parameters (input-output)	
area	Area factor (factor de área)
icvds	Initial D-S voltage (tensión inicial DS)
icvgs	Initial G-S voltage (tensión inicial GS)

MES - instance parameters (output-only)	
off	Device initially off (dispositivo inicialmente apagado)

dnode	Number of drain node (número de nodo de drenador)
gnode	Number of gate node (número de nodo de puerta)
snode	Number of source node (número de nodo de fuente)
dprimenode	Number of internal drain node (num. nodo interno drenador)
sprimenode	Number of internal source node(num. nodo interno fuente)
vgs	Gate-Source voltage (tensión puerta-fuente)
vgd	Gate-Drain voltage (tensión puerta-drenador)
cg	Gate capacitance (capacitancia de puerta)
cd	Drain capacitance (capacitancia de drenador)
cgd	Gate-Drain capacitance(capacitancia puerta-drenador)
gm	Transconductance (transconductancia)
gds	Drain-Source conductance (conductancia drenador-fuente)
ggs	Gate-Source conductance (conductancia puerta-fuente)
ggd	Gate-Drain conductance (conductancia puerta-drenador)
cqgs	Capacitance due to gate-source charge storage(capacitancia debido al almacenamiento de carga en puerta-fuente)
cqgd	Capacitance due to gate-drain charge storage(capacitancia debido al almacenamiento de carga en puerta-drenador)
qgs	Gate-Source charge storage (almacen. carga puerta-fuente)
qgd	Gate-Drain charge storage (almacen. carga puerta-drenador)
is	Source current (corriente de fuente)
p	Power dissipated by the mesfet (potencia disipado por el mesfet)

MES - model parameters (input-only)	
nmf	N type MESfet model (modelo MESfet tipo N)
pmf	P type MESfet model (modelo MESfet tipo P)

MES - model parameters (input-output)	
vt0	Pinch-off voltage (tensión de Pinch-off)
vto	Pinch-off voltage (tensión de Pinch-off)
alpha	Saturation voltage parameter (param. Tensión de saturación)
beta	Transconductance parameter (param. Transconductancia)
lambda	Channel length modulation parm. (param. Modulacion de longitud de canal)

b	Doping tail extending parameter(parámetro de extension de la cola de dopado)
rd	Drain ohmic resistance (resistencia ohmica de drenador)
rs	Source ohmic resistance (resistencia ohmica de fuente)
cgs	G-S junction capacitance (capacitancia unión GS)
cgd	G-D junction capacitance(capacitancia unión GD)
pb	Gate junction potential (potencial de la unión de puerta)
is	Junction saturation curren (corriente de saturación en la unión)
fc	Forward bias junction fit parm.(parámetro de ajuste de la union para polarización directa)
kf	Flicker noise coefficient (coeficiente ruido flicker)
af	Flicker noise exponent (exponente ruido flicker)

MES - model parameters (output-only)	
type	N-type or P-type MESfet model (modelo MESfet tipo P o N)
gd	Drain conductance (conductancia drenador)
gs	Source conductance (conductancia de fuente)
depl_cap	Depletion capacitance (capacitancia de deplexión)
vcrit	Critical voltage (tensión critica)

B.17 Mos1: Modelo MOSFET nivel 1 con modelo de capacitancia de Meyer

Mos1 - instance parameters (input-only)	
off	Device initially off (dispositivo inicialmente apagado)
ic	Vector of D-S, G-S, B-S voltages (vector de tensiones)

Mos1 - instance parameters (input-output)	
l	Length (longitud)
w	Width (anchura)
ad	Drain area (área de drenador)
as	Source area (área de fuente)
pd	Drain perimeter (perímetro de drenador)
ps	Source perimeter (perímetro de fuente)
nrd	Drain squares (cuadros de drenador)
nrs	Source squares (cuadros de fuente)
icvds	Initial D-S voltage (tensión inicial DS)
icvgs	Initial G-S voltage (tensión inicial GS)
icvbs	Initial B-S voltage (tensión inicial BS)

temp	Instance temperature (temperatura de dispositivo)

Mos1 - Instance parameters (output only)	
id	Drain current (corriente de drenador)
is	Source current (corriente de fuente)
ig	Gate current (corriente de puerta)
ib	Bulk current (corriente de sustrato)
ibd	B-D junction current (corriente de la unión BD)
ibs	B-S junction current (corriente de launio BS)
vgs	Gate-Source voltage (tensión puerta fuente)
vds	Drain-Source voltage (tensión drenador fuente)
vbs	Bulk-Source voltage (tensión sustrato fuente)
vbd	Bulk-Drain voltage (tensión sustrato-drenador)
dnode	Number of the drain node (número de nodo de drenador)
gnode	Number of the gate node (número de nodo de puerta)
snode	Number of the source node (número de nodo de fuente)
bnode	Number of the node (número de nodo)
dnodeprime	Number of int. drain node (num. Nodo interno de drenador)
snodeprime	Number of int. source node (num. Nodo interno de fuente)
von	Turn-on voltage (tensión de encendido)
vdsat	Saturation drain voltage (tensión de saturación en drenador)
sourcevcrit	Critical source voltage (tensión critica de fuente)
drainvcrit	Critical drain voltage(tensión critica de drenador)
rs	Source resistance (resistencia de fuente)
sourceconductance	Conductance of source (conductancia de fuente)
rd	Drain conductance (conductancia de drenador)
drainconductance	Conductance of drain (conductancia de drenador)
gm	Transconductance (transconductancia)
gds	Drain-Source conductance (conductancia drenador-fuente)
gmb	Bulk-Source transconductance (transconductancia sustrato-fuente)
gmbs	Bulk-Source transconductance(transconductancia sustrato-fuente)
gbd	Bulk-Drain conductance (conductancia sustrato-drenador)
gbs	Bulk-Source conductance (conductancia sustrato-fuente)
cbd	Bulk-Drain capacitance (capacitancia sustrato-drenador)

cbs	Bulk-Source capacitance (capacitancia sustrato-fuente)
cgs	Gate-Source capacitance (capacitancia puerta-fuente)
cgd	Gate-Drain capacitance (capacitancia puerta-drenador)
cgb	Gate-Bulk capacitance (capacitancia puerta-sustrato)
cqgs	Capacitance due to gate-source charge storage (capacitancia debida al almacenamiento de carga en la unión puerta-fuente)
cqgd	Capacitance due to gate-drain charge storage (capacitancia debida al almacenamiento de carga en la unión puerta-drenador)
cqgb	Capacitance due to gate-bulk charge storage (capacitancia debida al almacenamiento de carga en la unión puerta-sustrato)
cqbd	Capacitance due to bulk-drain charge storage (capacitancia debida al almacenamiento de carga en la unión sustrato-drenador)
cqbs	Capacitance due to bulk-source charge storage (capacitancia debida al almacenamiento de carga en la unión sustrato-fuente)
cbd0	Zero-Bias B-D junction capacitance (capacitancia de la union BD a polarización nula)
cbdsw0	
cbs0	Zero-Bias B-S junction capacitance (capacitancia de la union BS a polarización nula)
cbssw0	
qgs	Gate-Source charge storage (almacen. carga puerta-fuente)
qgd	Gate-Drain charge storage (almacen. carga puerta-drenador)
qgb	Gate-Bulk charge storage (almacen. carga puerta-sustrato)
qbd	Bulk-Drain charge storage (almacen. carga sustrato-drenador)
qbs	Bulk-Source charge storage (almacen. carga sustrato-fuente)
p	Instantaneous power (potencia instantanea)

Mos1 - model parameters (input-only)	
nmos	N type MOSFET model (modelo MOSFET tipo N)
pmos	P type MOSFET model (modelo MOSFET tipo P)

Mos1 - model parameters (input-output)	
vto	Threshold voltage (tensión umbral)
vt0	Threshold voltage (tensión umbral)
kp	Transconductance parameter (parámetro de transconductancia)

gamma	Bulk threshold parameter (parámetro de umbral de sustrato)
phi	Surface potential (potencial de superficie)
lambda	Channel length modulation (modulacion de longitud de canal)
rd	Drain ohmic resistance (resistencia ohmica de drenador)
rs	Source ohmic resistance (resistencia ohmica de fuente)
cbd	B-D junction capacitance (capacitancia de la unión BD)
cbs	B-S junction capacitance (capacitancia de la unión BS)
is	Bulk junction sat.current (corriente de sat. unión sustrato)
pb	Bulk junction potential (potencial de unión sustrato)
cgso	Gate-source overlap cap. (cap. solapamiento puerta-fuente)
cgdo	Gate-drain overlap cap. (cap. solapamiento puerta-drenador)
cgbo	Gate-bulk overlap cap. (cap. solapamiento puerta-sustrato)
rsh	Sheet resistance (resistencia laminar)
cj	Bottom junction cap per area (cap. unión inferior por área)
mj	Bottom grading coefficient (coef. Gradación unión inferior)
cjsw	Side junction cap per area (cap. Unión superior por área)
mjsw	Side grading coefficient (coef. Gradación unión superior)
js	Bulk jct. sat. current density (densidad de corriente en saturación en la unión de sustrato)
tox	Oxide thickness (grosor del óxido)
ld	Lateral diffusion (difusión lateral)
u0	Surface mobility (movilidad en superficie)
uo	Surface mobility (movilidad en superficie)
fc	Forward bias junction fit parm.(parámetro de ajuste de la unión para polarización directa)
nsub	Substrate doping (dopado de sustrado)
tpg	Gate type (tipo de puerta)
nss	Surface state density (desidad de estados superficiales)
tnom	Parameter measurement temperature (param. Medida temp.)
kf	Flicker noise coefficient (coeficiente ruido flicker)
af	Flicker noise exponent (exponente ruido flicker)

Mos1 - model parameters (output-only)	
type	N-channel or P-channel MOS (MOS de canal P o N)

B.18 Mos2: Modelo MOSFET nivel 2 con modelo de capacitancia Meyer

Mos2 - instance parameters (input-only)	
off	Device initially off (dispositivo inicialmente en off)
ic	Vector of D-S, G-S, B-S voltages (vector de tensiones)

Mos2 - instance parameters (input-output)	
l	Length (longitud)
w	Width (anchura)
ad	Drain area (área de drenador)
as	Source area (área de fuente)
pd	Drain perimeter (perímetro de drenador)
ps	Source perimeter (perímetro de fuente)
nrd	Drain squares (cuadros del drenador)
nrs	Source squares (cuadros de la fuente)
icvds	Initial D-S voltage (tensión inicial DS)
icvgs	Initial G-S voltage (tensión inicial GS)
icvbs	Initial B-S voltage (tensión inicial BS)
temp	Instance operating temperature (Temp. de operación del dispositivo)

Bibliografia

[1] A. Vladimirescu and S. Liu, The Simulation of MOS Integrated Circuits Using SPICE2 ERL Memo No. ERL M80/7, Electronics Research Laboratory University of California, Berkeley, October 1980

[2] T. Sakurai and A. R. Newton, A Simple MOSFET Model for Circuit Analysis and its application to CMOS gate delay analysis and series-connected MOSFET Structure ERL Memo No. ERL M90/19, Electronics Research Laboratory, University of California, Berkeley, March 1990

[3] B. J. Sheu, D. L. Scharfetter, and P. K. Ko, SPICE2 Implementation of BSIM ERL Memo No. ERL M85/42, Electronics Research Laboratory University of California, Berkeley, May 1985

[4] J. R. Pierret, A MOS Parameter Extraction Program for the BSIM Model ERL Memo Nos. ERL M84/99 and M84/100, Electronics Research Laboratory University of California, Berkeley, November 1984

[5] Min-Chie Jeng, Design and Modeling of Deep-Submicrometer MOSFETSs ERL Memo Nos. ERL M90/90, Electronics Research Laboratory University of California, Berkeley, October 1990

[6] Soyeon Park, Analysis and SPICE implementation of High Temperature Effects on MOSFET, Master's thesis, University of California, Berkeley, December 1986.

[7] Clement Szeto, Simulator of Temperature Effects in MOSFETs (STEIM), Master's thesis, University of California, Berkeley, May 1988.

[8] J.S. Roychowdhury and D.O. Pederson, Efficient Transient Simulation of Lossy Interconnect, Proc. of the 28th ACM/IEEE Design Automation Conference, June 17-21 1991, San Francisco

[9] A. E. Parker and D. J. Skellern, An Improved FET Model for Computer Simulators, IEEE Trans CAD, vol. 9, no. 5, pp. 551-553, May 1990.

[10] R. Saleh and A. Yang, Editors, Simulation and Modeling, IEEE Circuits and Devices, vol. 8, no. 3, pp. 7-8 and 49, May 1992

[11] H.Statz et al., GaAs FET Device and Circuit Simulation in SPICE, IEEE Transactions on Electron Devices, V34, Number 2, February, 1987 pp160-169.

Traducido por Pablo Antonio Álvarez Benavides & Andrés María Roldán Aranda del manual original de SPICE3 para el proyecto

Object Pascal Source Files

/

- CLXAbout.pas
- CLXChildWin.pas
- CLXMain.pas
- CLXMDIApp.dpr
- CLXSetup.pas

Lib/

- CadBasic.pas
- CadDevices.pas
- CadGrip.pas
- CadLayer.pas
- CadMisc.pas
- CadObjects.pas
- CadObjects3D.pas
- CadObjLink.pas
- CadOsnap.pas
- CadProperty.pas
- CadRubber.pas
- CadSelect.pas
- CadTree.pas
- CadUndo.pas
- CommonCAD.pas

```
unit CLXAbout;

interface

uses
  SysUtils, Types, Classes, QGraphics, QControls, QForms, QDialogs,
  QStdCtrls, QExtCtrls;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    OKButton: TButton;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

{$R *.xfm}

end.
```

```
unit CLXChildWin;

interface

uses
  SysUtils, Types, Classes, QGraphics, QControls, QForms, QDialogs,
  QStdCtrls, QExtCtrls, CommonCAD;

type
  TMDIChild = class(TForm)
    Memo1: TMemo;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Drawing: TDraw;
  end;

var
  MDIChild: TMDIChild;

implementation

{$R *.xfm}

uses CLXMain;

procedure TMDIChild.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if Memo1.Enabled and (Caption='Modelos') then
     if Application.MessageBox('Desea guardar los cambios a los modelos?', 'Guardar',
[smbOK, smbCancel])=smbOK then
        Memo1.Lines.SaveToFile('models.cir');
  if Memo1.Enabled and (Caption='Netlist') then
     if Application.MessageBox('Desea guardar los cambios a la Netlist?', 'Guardar',
[smbOK, smbCancel])=smbOK then
        Memo1.Lines.SaveToFile(CommonCad.Drawing.Caption+'.cir');
  Action := caFree;
  inherited;
end;

procedure TMDIChild.FormActivate(Sender: TObject);
begin
     if (Self.Drawing<>NIL) then
        CommonCad.Drawing:= Self.Drawing;
end;

end.
```

```
unit CLXMain;

interface

uses
 {$IFDEF LINUX}
  libc,
 {$ENDIF}
 {$IFDEF MSWINDOWS}
  ShellAPI,
 {$ENDIF}
  SysUtils, Classes, QForms, QImgList, QStdActns, QActnList, QDialogs,
  QMenus, QTypes, QComCtrls, QControls, QExtCtrls, QStdCtrls,
  QGraphics, Types;

type
  TMainForm = class(TForm)
    OpenDialog: TOpenDialog;
    StatusBar: TStatusBar;
    ActionList1: TActionList;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    FileNew1: TAction;
    FileSave1: TAction;
    FileExit1: TAction;
    FileOpen1: TAction;
    FileSaveAs1: TAction;
    WindowCascade1: TWindowCascade;
    WindowMinimizeAll1: TWindowMinimizeAll;
    HelpAbout1: TAction;
    FileClose1: TWindowClose;
    WindowClose1: TWindowClose;
    WindowTile1: TWindowTile;
    ImageList1: TImageList;
    ControlBar1: TControlBar;
    ToolBar2: TToolBar;
    ToolButton9: TToolButton;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ToolButton7: TToolButton;
    ToolButton8: TToolButton;
    ToolButton10: TToolButton;
    ToolButton12: TToolButton;
    ToolBar1: TToolBar;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    FileNewItem: TMenuItem;
    FileOpenItem: TMenuItem;
    FileCloseItem: TMenuItem;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    Edit1: TMenuItem;
```

```
    CutItem: TMenuItem;
    CopyItem: TMenuItem;
    PasteItem: TMenuItem;
    Window1: TMenuItem;
    TreeView1: TMenuItem;
    WindowTileItem: TMenuItem;
    WindowMinimizeItem: TMenuItem;
    Layers1: TMenuItem;
    Help1: TMenuItem;
    HelpAboutItem: TMenuItem;
    ToolButton11: TToolButton;
    ToolButton13: TToolButton;
    ToolButton14: TToolButton;
    ToolButton15: TToolButton;
    ToolButton16: TToolButton;
    ToolButton17: TToolButton;
    ToolButton18: TToolButton;
    ToolButton19: TToolButton;
    ToolButton20: TToolButton;
    ToolButton21: TToolButton;
    ToolButton22: TToolButton;
    ToolButton23: TToolButton;
    ToolButton24: TToolButton;
    ToolButton25: TToolButton;
    ToolButton26: TToolButton;
    ToolButton27: TToolButton;
    ToolButton28: TToolButton;
    ToolButton29: TToolButton;
    DeviceList1: TImageList;
    ComboBox1: TComboBox;
    N2: TMenuItem;
    Deshacer1: TMenuItem;
    Repetir1: TMenuItem;
    N3: TMenuItem;
    Borrar1: TMenuItem;
    ToolButton30: TToolButton;
    ToolButton31: TToolButton;
    ToolButton32: TToolButton;
    N4: TMenuItem;
    Modelos1: TMenuItem;
    NetList1: TMenuItem;
    ConfigurarSimulacin1: TMenuItem;
    procedure FileNew1Execute(Sender: TObject);
    procedure FileOpen1Execute(Sender: TObject);
    procedure HelpAbout1Execute(Sender: TObject);
    procedure FileExit1Execute(Sender: TObject);
    procedure ToolButton13Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ToolButton14Click(Sender: TObject);
    procedure ToolButton16Click(Sender: TObject);
    procedure ToolButton17Click(Sender: TObject);
    procedure ToolButton24Click(Sender: TObject);
    procedure ToolButton22Click(Sender: TObject);
    procedure File1Click(Sender: TObject);
    procedure FileSaveItemClick(Sender: TObject);
    procedure ControlBar1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure Deshacer1Click(Sender: TObject);
    procedure Repetir1Click(Sender: TObject);
    procedure CopyItemClick(Sender: TObject);
    procedure CutItemClick(Sender: TObject);
```

```
    procedure PasteItemClick(Sender: TObject);
    procedure Borrar1Click(Sender: TObject);
    procedure Edit1Click(Sender: TObject);
    procedure TreeView1Click(Sender: TObject);
    procedure Layers1Click(Sender: TObject);
    procedure ToolButton21Click(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure ComboBox1Change(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ToolButton30Click(Sender: TObject);
    procedure ToolButton15Click(Sender: TObject);
    procedure ToolButton23Click(Sender: TObject);
    procedure ToolButton26Click(Sender: TObject);
    procedure ToolButton25Click(Sender: TObject);
    procedure ToolButton20Click(Sender: TObject);
    procedure ToolButton27Click(Sender: TObject);
    procedure ToolButton28Click(Sender: TObject);
    procedure Modelos1Click(Sender: TObject);
    procedure ConfigurarSimulacin1Click(Sender: TObject);
    procedure NetList1Click(Sender: TObject);
  private
    { Private declarations }
    procedure CreateMDIChild(const Name: string);
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.xfm}

uses CLXAbout, CommonCad, QClipbrd, CLXSetup, CLXChildWin;

procedure TMainForm.CreateMDIChild(const Name: string);
var
    Child: TMDIChild;
begin
{//}  if not Assigned(ActiveMDIChild) then
  begin
  // create a new MDI child window
  Child := TMDIChild.Create(Application);
  Child.KeyPreview := True;
  CommonCad.Drawing:= tDraw.Create(Child);
  CommonCad.Drawing.StatusPanel:= StatusBar.Panels[0];
  CommonCad.Drawing.GetName('SinTitulo');
  Child.Drawing:= CommonCad.Drawing;
  Child.Drawing.Changed:= True;
//{
  end
  else
  begin
  Child:=TMDIChild(ActiveMDIChild);
  Child.Drawing.GripList.
Clear;                                                    //if there is a
selection
  Child.Drawing.DestroyComponents;
```

```
    Child.Drawing.ChildNames.Clear;
    Child.Drawing.Invalidate;
    Child.Drawing.Changed:= True;
    SetLength(Analisis.Puertos, 0);
  //}
    end;
    if FileExists(Name) then
    begin
       Child.Drawing.LoadFromFile(Name);
       Child.Caption:=ExtractFilename(Name);
    end
    else
    begin
       Child.Drawing.Caption:= Name;
       Child.Caption:= Name;
    end;
    Child.WindowState:=wsMaximized;
    Child.Drawing.NameChanged;
end;

procedure TMainForm.FileNewlExecute(Sender: TObject);
begin
  CreateMDIChild('SinTitulo');
end;

procedure TMainForm.FileOpenlExecute(Sender: TObject);
begin
  with OpenDialog do
  begin
    InitialDir:= ExtractFilePath(Application.ExeName)+'circuitfiles';
    DefaultExt:= '*.ccad';
    Filter:= 'CommonCAD file (*.ccad)|All files (*.*)|';
    FilterIndex:= 0;
    if Execute then
      CreateMDIChild(OpenDialog.FileName);
  end;
end;

procedure TMainForm.HelpAboutlExecute(Sender: TObject);
begin
  AboutBox.ShowModal;
end;

procedure TMainForm.FileExitlExecute(Sender: TObject);
begin
  Close;
end;

procedure TMainForm.ToolButtonl3Click(Sender: TObject);
begin
  //Resistencia
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Resistance');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.FormCreate(Sender: TObject);
```

```
var
  i: Integer;

begin
  FileNew1Execute(Application);
  for i := 1 to ParamCount do
  begin
       if FileExists(ParamStr(i)) then
          CreateMDIChild(ParamStr(i));
  end;

  with Analisis do
  begin
{
      n:=3;
      s:=1;
      k:=20;
      freq:=1000;
}
      timestep:=1E-5;
      endtime:=1E-3;
      Tipo:=True;
      Netlist:=True;
  end;
end;

procedure TMainForm.ToolButton14Click(Sender: TObject);
begin
  //Capacitor
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Capacitance');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton16Click(Sender: TObject);
begin
  //Diodo
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Diode');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton17Click(Sender: TObject);
begin
  //Fuente DC
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Vsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
```

```
end;

procedure TMainForm.ToolButton24Click(Sender: TObject);
begin
  //Tierra
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Anground');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton22Click(Sender: TObject);
begin
  //BJT NPN
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Bjt');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton15Click(Sender: TObject);
begin
//Inductance
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Inductance');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;


procedure TMainForm.ComboBox1Change(Sender: TObject);
begin
    with Drawing do
       begin
          if ComboBox1.Text='25%'  then Zoom.Factor:=0.0025;
          if ComboBox1.Text='50%'  then Zoom.Factor:=0.0050;
          if ComboBox1.Text='75%'  then Zoom.Factor:=0.0075;
          if ComboBox1.Text='100%' then Zoom.Factor:=0.0100;
          if ComboBox1.Text='150%' then Zoom.Factor:=0.0150;
          if ComboBox1.Text='200%' then Zoom.Factor:=0.0200;
          Invalidate;
       end;
end;

procedure TMainForm.File1Click(Sender: TObject);
begin
  FileSaveItem.Enabled:= Drawing.Changed;  //grayed the menu item
end;

procedure TMainForm.FileSaveItemClick(Sender: TObject);
begin
```

```
  with tSaveDialog.Create(Application)do
  begin
    FileName:= Drawing.Caption;
    InitialDir:= ExtractFilePath(Application.ExeName);
    DefaultExt:= '*.ccad';
    Filter:= 'CommonCAD file (*.ccad)|All files (*.*)|';
    FilterIndex:= 0;
    if Execute then
      Drawing.SaveToFile(FileName);
    Free;
  end;
end;

procedure TMainForm.ControlBar1MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
begin
     ToolButton2.Enabled:= Drawing.Changed;  //grayed the menu item
     ToolButton5.Enabled:= Drawing.SelectList.Count> 0;
     ToolButton4.Enabled:= Drawing.SelectList.Count> 0;
     ToolButton6.Enabled:= Clipboard.Provides('text/plain');
end;

procedure TMainForm.Deshacer1Click(Sender: TObject);
begin
     Drawing.Undo;
end;

procedure TMainForm.Repetir1Click(Sender: TObject);
begin
     Drawing.Redo;
end;

procedure TMainForm.CopyItemClick(Sender: TObject);
begin
     Drawing.CopyToClipboard;
end;

procedure TMainForm.CutItemClick(Sender: TObject);
begin
     Drawing.CopyToClipboard;
     Drawing.Delete;
end;

procedure TMainForm.PasteItemClick(Sender: TObject);
begin
     Drawing.PasteFromClipboard;
end;

procedure TMainForm.Borrar1Click(Sender: TObject);
begin
     Drawing.Delete;
end;

procedure TMainForm.Edit1Click(Sender: TObject);
begin
  Repetir1.Enabled:= Drawing.RedoList.Count> 0;
  Deshacer1.Enabled:= Drawing.UndoList.Count> 0;
  CopyItem.Enabled:= Drawing.SelectList.Count> 0;
  CutItem.Enabled:= Drawing.SelectList.Count> 0;
  Borrar1.Enabled:= Drawing.SelectList.Count> 0;
  PasteItem.Enabled:= Clipboard.Provides('text/plain');
```

```
end;

procedure TMainForm.TreeView1Click(Sender: TObject);
begin
  Drawing.DialogTree.MenuItem:= Treeview1;
  if Treeview1.Checked
    then Drawing.DialogTree.Close
    else Drawing.DialogTree.Show;
end;

procedure TMainForm.Layers1Click(Sender: TObject);
begin
  Drawing.DialogLayers.MenuItem:= Layers1;
  if Layers1.Checked
    then Drawing.DialogLayers.Close
    else Drawing.DialogLayers.Show;
end;

procedure TMainForm.ToolButton21Click(Sender: TObject);
begin
  with Drawing.Command do
  begin
    Status:= csCreateAddpoint;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Polyline');
  end;
end;

procedure TMainForm.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if Assigned(ActiveMDIChild) then
  begin
     TMDIChild(ActiveMDIChild).Drawing.DoKeyDown(Sender, Key, Shift);
  end;
end;

procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
     if Drawing.PopupInspector.Visible then
        Drawing.PopupInspector.FormDeactivate(Self);
{
     Drawing.GripList.Clear;         //if there is a selection
     Drawing.DestroyComponents;    //deleting of the drawing elements
     Drawing.FreeOnRelease;
//}
end;

procedure TMainForm.ToolButton30Click(Sender: TObject);
var
   Handle: Integer;
begin
    if (Analisis.n=0) OR (Length(Analisis.Puertos)=0) then
       FrmSetup.ShowModal;
    if Analisis.NetList then
       Drawing.SaveNetList;
 {$IFDEF LINUX}
    if libc.system('/root/projects/hbdemo/hbdemo')=-1 then
       ShowMessage('Can''''t start the program');
 {$ENDIF}
 {$IFDEF MSWINDOWS}
```

```
    Handle:=0;
    ShellExecute(Handle, 'open', PChar(ExtractFileDir(Application.ExeName)+'\spice.bat'),
PChar(Drawing.Caption+'.cir'), NIL, 1);
 {$ENDIF}
end;

procedure TMainForm.ToolButton23Click(Sender: TObject);
begin
  //Fuente DC
  with Drawing.Command do
  begin
    Status:= csCreate;
    Drawing.ChildNames.Add('Grounded Source');
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Vsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton26Click(Sender: TObject);
begin
  //Fuente AC
  with Drawing.Command do
  begin
    Status:= csCreate;
    Drawing.ChildNames.Add('AC Source');
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Vsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton25Click(Sender: TObject);
begin
  //Fuente de Corriente
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Isource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton20Click(Sender: TObject);
begin
  //Unknown
  with Drawing.Command do
  begin
    Status:= csCreate;
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Bsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton27Click(Sender: TObject);
begin
  //Fuente Variable
  with Drawing.Command do
```

```
  begin
    Status:= csCreate;
    Drawing.ChildNames.Add('Voltage Source');
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Dsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.ToolButton28Click(Sender: TObject);
begin
  //Fuente Variable
  with Drawing.Command do
  begin
    Status:= csCreate;
    Drawing.ChildNames.Add('Current Source');
    ObjectType:= Drawing.DrawCompItemList.GetClassType('Dsource');
    Element:= Drawing.CreateObject(Drawing.Mouse.Pos);
    Drawing.Invalidate;
  end;
end;

procedure TMainForm.Modelos1Click(Sender: TObject);
var
    Child: TMDIChild;
begin
     // create a new MDI child window
     if FileExists('models.cir') then
     begin
        Child := TMDIChild.Create(Application);
        Child.Caption:='Modelos';
        Child.Memo1.ScrollBars:=ssAutoBoth;
        Child.Memo1.Visible:=true;
        Child.Memo1.Enabled:=true;
        Child.Memo1.Lines.LoadFromFile('models.cir');
     end
     else
        Application.MessageBox('El archivo models.cir no existe por favor reinstale la
aplicación', 'Advertencia', [smbOK])
end;

procedure TMainForm.ConfigurarSimulacin1Click(Sender: TObject);
begin
     FrmSetup.Show;
end;

procedure TMainForm.NetList1Click(Sender: TObject);
var
    Child: TMDIChild;
begin
     if FileExists(Drawing.Caption+'.cir') then
     begin
        Child := TMDIChild.Create(Application);
        Child.Caption:='Netlist';
        Child.Memo1.ScrollBars:=ssAutoBoth;
        Child.Memo1.Visible:=true;
        Child.Memo1.Enabled:=true;
        Child.Memo1.Lines.LoadFromFile(Drawing.Caption+'.cir');
     end
     else
        Application.MessageBox('El archivo Netlist no existe'+Chr(13)+Chr(10)+'configure
```

```
program CLXMDIApp;

uses
  QForms,
  CLXMain in 'CLXMain.pas' {MainForm},
  CLXChildWin in 'CLXChildWin.pas' {MDIChild},
  CLXAbout in 'CLXAbout.pas' {AboutBox},
  CLXSetup in 'CLXSetup.pas' {FrmSetup};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TAboutBox, AboutBox);
  Application.CreateForm(TFrmSetup, FrmSetup);
  Application.Run;
end.
```

```
unit CLXSetup;

interface

uses
  SysUtils, Types, Classes, QGraphics, QControls, QForms, QDialogs,
  QStdCtrls, QExtCtrls;

type
  TFrmSetup = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    RadioGroup1: TRadioGroup;
    CheckBox1: TCheckBox;
    Button1: TButton;
    Button2: TButton;
    GroupBox2: TGroupBox;
    Label7: TLabel;
    Label9: TLabel;
    Edit6: TEdit;
    Edit8: TEdit;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure RadioGroup1Click(Sender: TObject);
  private
    { Private declarations }
    procedure SaveAnalisis;
  public
    { Public declarations }
  end;

var
  FrmSetup: TFrmSetup;

implementation

{$R *.xfm}

uses CommonCad, CadDevices;

procedure TFrmSetup.Button2Click(Sender: TObject);
begin
    Close;
end;

procedure TFrmSetup.Button1Click(Sender: TObject);
```

```
var
   i , number: integer;
   values, srcvalues: TStrings;

begin
    values := TStringList.Create; { construct the list object }
    srcvalues := TStringList.Create;

    try
    { use the string list }

    with Analisis do
    begin
        n:= StrToInt(Edit1.Text);
        s:= StrToInt(Edit2.Text);
        k:= StrToInt(Edit3.Text);
        freq:= StrToFloat(Edit4.Text);
        timestep:= StrToFloat(Edit6.Text);
        endtime:= StrToFloat(Edit8.Text);
        Tipo:= (RadioGroup1.ItemIndex=1);
        Netlist:= CheckBox1.Checked;
        number:= ExtractStrings([',',';'],[' '],PChar(AnsiString(Edit5.Text)),values);
        if number>0 then
        begin
           SetLength(Puertos, number);
           for i:= 0 to number-1 do
               Puertos[i]:=StrToInt(values[i]);
        end;
     end;
    Close;

    finally
        values.Free;     { destroy the list object }
        srcvalues.Free;
        SaveAnalisis;
    end;
end;

procedure TFrmSetup.FormShow(Sender: TObject);
var
   i , no: integer;
begin
     with Analisis do
     begin
        Edit1.Text:= IntToStr(n);
        Edit2.Text:= IntToStr(s);
        Edit3.Text:= IntToStr(k);
        Edit4.Text:= FloatToStr(freq);
        Edit6.Text:= FloatToStr(timestep);
        Edit8.Text:= FloatToStr(endtime);
        RadioGroup1.ItemIndex:=Ord(Tipo);
        CheckBox1.Checked:=Netlist;
        Edit5.Text:='';
        no:=Length(Puertos);
        if no>0 then
           for i:=0 to no-1 do
           begin
               Edit5.Text:=Edit5.Text+IntToStr(Puertos[i]);
```

```
                if i<>no-1 then Edit5.Text:=Edit5.Text+',';
             end
          else
             Edit5.Text:=CommonCad.Puertos;
      end;
end;

procedure TFrmSetup.RadioGroup1Click(Sender: TObject);
begin
      if (RadioGroup1.ItemIndex=1) then
      begin
         GroupBox1.Visible:=true;
         GroupBox2.Visible:=false;
      end
      else
      begin
         GroupBox1.Visible:=false;
         GroupBox2.Visible:=true;
      end
end;

procedure TFrmSetup.SaveAnalisis;
var aFileName: String;
    AnalisisFile, srcvalues: TStrings;
    no, i, j, number: Integer;

begin
      AnalisisFile:= TStringList.Create;

      Screen.Cursor:= crHourGlass;
      aFileName:='analisis.opt';
      AnalisisFile.Clear;
      with Analisis do
      begin
         AnalisisFile.Add(Drawing.Caption);
         AnalisisFile.Add(IntToStr(n)+ #09 + 'n');
         AnalisisFile.Add(IntToStr(s)+ #09 + 's');
         AnalisisFile.Add(IntToStr(k)+ #09 + 'k');
         AnalisisFile.Add(FloatToStr(freq)+ #09 + 'freq');
         AnalisisFile.Add(FloatToStr(timestep)+ #09 + 'timestep');
         AnalisisFile.Add(FloatToStr(endtime)+ #09 + 'endtime');
         AnalisisFile.Add(IntToStr(Ord(Tipo))+ #09 + 'Tipo');
         no:=Length(Puertos);
         AnalisisFile.Add(IntToStr(no)+ #09 + 'Puertos');
         if no>0 then
            for i:=0 to no-1 do
            begin
                AnalisisFile.Add(IntToStr(Puertos[i]));
            end;

         srcvalues := TStringList.Create;
         try
         with Drawing do
         for i:= 0 to ComponentCount- 1 do
         begin
           if Components[i].ClassNameIs('tVSource') then
           begin
              srcvalues.Clear;
```

```
            number:= ExtractStrings([',',';'],[' '],PChar(tVSource(Components[i]).
FreqElms), srcvalues);
            if number>0 then
            begin
               AnalisisFile.Add(IntToStr(number)+ #09 + 'Sources');
               for j:= 0 to number-1 do
               begin
                  AnalisisFile.Add(srcvalues[j]);
               end;
            end;
         end;
      end;
      finally
         srcvalues.Free;
      end;
   end;

   AnalisisFile.SaveToFile(ExtractFileDir(Application.ExeName)+'\circuitfiles
\'+aFileName);
   Screen.Cursor:= crDefault;
end;

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadBasic;

interface

uses Classes, QGraphics, Types, QForms, QControls, QT, QComCtrls,
     CadGrip, CadMisc, CadProperty;

type
  tChildNames = class(tStringList)
  public
    function GetNumber(S: String): Integer;
    function GetName(Basic: String): String;
  end;

  tDrawComponent = class;

  tDrawPoint = class;
  tPointType = (ptFree, ptRect);

  tPointList = class(tList)                                          // contains
drawpoints
  private
    fOwner: tComponent;
    Temp: tPointList;
    function OnLineRegion(PP1, PP2: t3dPoint; PP9: tPoint): Boolean;
  public
    PointArray: array of tPoint;
    PointType: tPointType;
    constructor CreateBy(aOwner: tComponent);
    destructor Destroy; override;
    procedure AssignTo(aPointList: tPointList; aComponent: tComponent);
    procedure DrawPolygon(aCanvas: tCanvas);
    procedure DrawPolyline(aCanvas: tCanvas);
    function GetRgn(aClosed, aFilled: Boolean): QRegionH;
    procedure Mirror(P1, P2: tPoint);
    procedure MoveTo(X, Y, Z: Integer);
    function OnConnect(aPoint: tPoint): Boolean;
    procedure PutGrips;
    procedure ReadFromString(aString: String);
    procedure RectGripDrag(aPoint: tDrawPoint; NewCoord: t3dPoint);
    procedure RemoveLast;
    procedure Reverse;
    procedure RetrieveValue;
    procedure Rotate(aRP: t3dPoint; aRad: Double);
    procedure Rotate3d(MM: tMatrix);
    procedure RotateByRubberLine;
    procedure ScaleByRubberLine;
    procedure SaveValue;
    procedure SetPointType;
    procedure Shift;
    procedure Update;
    function WriteToString: String;
```

```
  end;

  tDrawPoint = class
  private
  public
    Value: t3dPoint;                                                    //
coordinates
    Component: tComponent;                                              // owner
(tDrawComponent)
    PointType: tPointType;                                              // which
type of grip will use
    constructor CreateBy(aPoint: t3DPoint; aComponent: tComponent);
    function Clone(aComponent: tComponent): tDrawPoint;
    function Get2dPoint: tPoint;
    function GetPoint: t3dPoint;
    procedure MoveTo(X, Y, Z: Integer);
    procedure PutGrips;
    procedure Rotate3d(MM: tMatrix);
  end;

  tBasic = class(tComponent)
  protected
    procedure Clip(var aStr: String);
    function GetCloseString(aString: String): String;
    function Indent(Num: Integer): String;
  public
    Color, LinkID: Integer;
    Caption: String;
    ChildNames: tChildNames;
    Node: tTreeNode;
    PropList: tPropList;
    function GetName(aName: String): String;
    procedure AddTree(aTree: tTreeView; aParent: tTreeNode); virtual;
    procedure ChangeOwner(aNew: tBasic);
    procedure ChangePosition(aNew: tBasic);
    procedure NameChanged; virtual;
    function OnConnect(aPoint: tPoint): tDrawComponent; virtual;
    function OnItem(aPoint: tPoint): tComponent; virtual;
    procedure Paint(aCanvas :tCanvas); virtual;
    procedure ReadXml(aString: String); virtual;
    procedure SetCaption(aValue: String);
    procedure SetPropList; virtual;
    procedure WriteXml(aIndent: Integer); virtual;
    procedure WriteCir(aIndent: Integer); virtual;
  end;

  tDrawComponent = class(tBasic)
  private
  public
    CreatingByAddPoint, Closed, Filled: Boolean;
    Layer: String;
    LineColor, LineWidth: Integer;
    PointList: tPointList;
    constructor Create(aOwner: tComponent); override;
    constructor CreateBy(aOwner: tComponent); virtual;
    constructor CreateByPoint(aOwner: tComponent; aPoint: t3DPoint); virtual;
    procedure AddTree(aTree: tTreeView; aParent: tTreeNode); override;
    procedure Draw(aCanvas :tCanvas); virtual;
    function Clone(aOwner: tComponent): tDrawComponent;
    function GetLinkPoint: t3dPoint; virtual;
    function GetRgn: QRegionH; virtual;
```

```
    procedure Mirror(P1, P2: tPoint); virtual;
    procedure MoveTo(X, Y, Z: Integer); virtual;
    function OnConnect(aPoint: tPoint): tDrawComponent; override;
    function OnItem(aPoint: tPoint): tComponent; override;
    procedure OnSelectingRect(aHR: QRegionH; Crossing: Boolean);
    procedure Paint(aCanvas :tCanvas); override;
    procedure PaintTo(aX, aY, aZ: Integer; aCanvas :tCanvas); virtual;
    procedure PutGrips; virtual;
    procedure Rotate3d(MM: tMatrix);
    procedure RotateByRubberLine; virtual;
    procedure ScaleByRubberLine; virtual;
    procedure SetPropList; override;
    procedure SetRubberLine;
    procedure SizeTo(aPoint: tPoint); virtual;
    procedure SetupCanvas(aCanvas :tCanvas);
    procedure Update;
  end;

  tDrawingClass = class of tDrawComponent;

  tDrawCompItem = class
  public
    Name: String;
    ClassType: tDrawingClass;
    constructor Create(aName: String; aClassType: tDrawingClass);
  end;

  tDrawCompItemList = class(tList)
  public
    function GetName(aClassType: tDrawingClass): String;
    function GetClassType(aName: String): tDrawingClass;
  end;

var
  DrawCompItemList: tDrawCompItemList;

implementation

uses QDialogs, SysUtils,
     CadLayer, CadOsnap, CadSelect, CadUndo, CommonCAD, CadDevices;

//tChildNames

const NumChar: Set of Char = ['0','1','2','3','4','5','6','7','8','9'];

function tChildNames.GetNumber(S: String): Integer;
var i, Code, Num: Integer;
begin
  for i:= Length(S) downto 1 do
    if not (S[i] in NumChar) then Break;
  Val(System.Copy(S, i+1, Length(S)-i), Num, Code);
  if Code <> 0 then
  begin
    ShowMessage('Problem with objectname: '+ S);
    Application.Terminate;
  end;
  Result:= Num;
end;

function tChildNames.GetName(Basic: String): String;
var i, Num: Integer;
    S: String;
```

```
begin
  if (Pos(Basic, Text) > 0) then
  begin
    for i:= 0 to Count-1 do
    if (Pos(Basic, Strings[i]) > 0) then
    begin
      Num:= GetNumber(Strings[i]);
      Str(Num+ 1, S);
      S:= Basic+ S;
      Delete(i);
      Insert(0, S);
      Break;
    end;
  end else
  begin
    S:= Basic+ '1';
    Add(S);
  end;
  Result:= S;
end;

//tPointList

constructor tPointList.CreateBy(aOwner: tComponent);
begin
  inherited Create;
  fOwner:= aOwner;
end;

destructor tPointList.Destroy;
var i: Integer;
begin
  for i:= Count- 1 downto 0 do
    tComponent(Items[i]).Free;
  inherited;
end;

procedure tPointList.AssignTo(aPointList: tPointList; aComponent: tComponent);
var i: Integer;
begin
  for i:= 0 to aPointList.Count- 1 do
    Add(tDrawPoint(aPointList.Items[i]).Clone(aComponent));
end;

procedure tPointList.DrawPolygon(aCanvas: tCanvas);
begin
  Update;
  aCanvas.Polygon(PointArray, False, 0, Count);
end;

procedure tPointList.DrawPolyline(aCanvas: tCanvas);
begin
  Update;
  aCanvas.Polyline(PointArray, 0, Count);
end;

function tPointList.GetRgn(aClosed, aFilled: Boolean): QRegionH;
var i: Integer;
    P1, P2: tPoint;
    R1, R2: QRegionH;
    MM: Array of tPoint;
begin
```

```
  Result:= nil;
  if Count= 0 then Exit;
  if (aClosed and aFilled) then
  begin
    //Result:= CreatePolygonRgn(PointArray[0], Count, WINDING);
    Result:= QRegion_create(@PointArray[0], false);
    Exit;
  end;
  try
    SetLength(MM, 4);
    R1:=QRegion_create();
    for i:= 1 to Count- 1 do
    begin
      P1:= PointArray[i- 1];
      P2:= PointArray[i];
      MM[0]:=Point(P1.X-2, P1.Y-2);
      MM[1]:=Point(P1.X+2, P1.Y+2);
      MM[2]:=Point(P2.X+2, P2.Y+2);
      MM[3]:=Point(P2.X-2, P2.Y-2);
      if i= 1 then
      begin
          QRegion_destroy(R1);
          R1:= QRegion_create(@MM[0], false);
      end else
      begin
          R2:= QRegion_create(@MM[0], false);
          QRegion_unite(R1, R1, R2);
          QRegion_destroy(R2);
      end;
    end;
    if aClosed then
    begin
      P1:= PointArray[Count- 1];
      P2:= PointArray[0];
      MM[0]:=Point(P1.X-2, P1.Y-2);
      MM[1]:=Point(P1.X+2, P1.Y+2);
      MM[2]:=Point(P2.X+2, P2.Y+2);
      MM[3]:=Point(P2.X-2, P2.Y-2);
      R2:= QRegion_create(@MM[0], false);
      QRegion_unite(R1, R1, R2);
      QRegion_destroy(R2);
    end;
    Result:= R1;
//    R1.Free;
//    MM.Free;
  except
    Result:= nil;
  end;
end;

{
var i, DD: Integer;
    P1, P2: tPoint;
    P1F: TGPPointF;
    R1, R2: TGPRegion;
    AA: Single;
    MM: TGPMatrix;
    myGraphic: TGPGraphics;

begin
  //agregar lineas aqui
```

```
    try
      R1:= TGPRegion.Create;
      for i:= 1 to Count- 1 do
      begin
        P1:= PointArray[i- 1];
        P1F:= MakePoint(P1.X*1.0, P1.Y*1.0);
        P2:= PointArray[i];
        DD:= Round(Dist(P1, P2));
        AA:= RadToAng(Ang(P1, P2));
        MM:= TGPMatrix.Create;
        MM.Translate(P1F.X, P1F.Y, MatrixOrderAppend);
        MM.RotateAt(AA, P1F, MatrixOrderAppend);
        if i= 1 then
        begin
          R1.Free;
          R1:= TGPRegion.Create(MakeRect(0, -2, DD, 4));
          R1.Transform(MM);
        end else
        begin
          R2:= TGPRegion.Create(MakeRect(0, -2, DD, 4));
          R2.Transform(MM);
          R1.Union(R2);
          R2.Free;
        end;
        MM.Free;
      end;
      if aClosed then
      begin
        P1:= PointArray[Count- 1];
        P1F:= MakePoint(P1.X*1.0, P1.Y*1.0);
        P2:= PointArray[0];
        DD:= Round(Dist(P1, P2));
        AA:= RadToAng(Ang(P1, P2));
        MM:= TGPMatrix.Create;
        MM.Translate(P1F.X, P1F.Y, MatrixOrderAppend);
        MM.RotateAt(AA, P1F, MatrixOrderAppend);
        R2:= TGPRegion.Create(MakeRect(0, -2, DD, 4));
        R2.Transform(MM);
        R1.Union(R2);
        R2.Free;
        MM.Free;
      end;
      myGraphic := TGPGraphics.Create(aCanvas.Handle);
      Result:= R1.GetQRegionH(myGraphic);
      myGraphic.Free;
      R1.Free;
    except
      Result:= nil;
    end;
  end;
  }

  procedure tPointList.Mirror(P1, P2: tPoint);
  var PP1, PP2, PPP: tPoint;
      i: Integer;
  begin
    PP1:= Drawing.Inverse(P1);
    PP2:= Drawing.Inverse(P2);
    for i:= 0 to Count- 1 do
    begin
      PPP:= Perp(tDrawPoint(Items[i]).Get2dPoint, PP1, PP2);
      PPP:= Pol(PPP,
```

```
              Ang(tDrawPoint(Items[i]).Get2dPoint, PPP),
              Dist(tDrawPoint(Items[i]).Get2dPoint, PPP));
    tDrawPoint(Items[i]).Value.X:= PPP.X;
    tDrawPoint(Items[i]).Value.Y:= PPP.Y;
  end;
  Reverse;
end;

procedure tPointList.MoveTo(X, Y, Z: Integer);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawPoint(Items[i]).MoveTo(X, Y, Z);
end;

function tPointList.OnLineRegion(PP1, PP2: t3dPoint; PP9: tPoint): Boolean;
var PA, PB: tPoint;
    RR: QRegionH;
    MM: array of tPoint;

begin
  PA:= Drawing.Adjust(PP1);
  PB:= Drawing.Adjust(PP2);
  SetLength(MM, 4);
  MM[0]:=Point(PA.X-2, PA.Y-2);
  MM[1]:=Point(PA.X+2, PA.Y+2);
  MM[2]:=Point(PB.X+2, PB.Y+2);
  MM[3]:=Point(PB.X-2, PB.Y-2);
  RR:= QRegion_create(@MM[0], false);
  Result:= QRegion_contains(RR, PPoint(@PP9));
  QRegion_destroy(RR);
{
var DD: Integer;
    PA, PB: tPoint;
    P1F: TGPPointF;
    RR: TGPRegion;
    AA: Single;
    MM: TGPMatrix;
    //myGraphic : TGPGraphics;                                    //for
testing regions
    //SolidBrush: TGPSolidBrush;                                  //
begin
  PA:= Drawing.Adjust(PP1);
  P1F:= MakePoint(PA.X*1.0, PA.Y*1.0);
  PB:= Drawing.Adjust(PP2);
  DD:= Round(Dist(PA, PB));
  AA:= RadToAng(Ang(PA, PB));
  MM:= TGPMatrix.Create;
  MM.Translate(P1F.X, P1F.Y, MatrixOrderAppend);
  MM.RotateAt(AA, P1F, MatrixOrderAppend);
  RR:= TGPRegion.Create(MakeRect(0, -2, DD, 4));
  RR.Transform(MM);
  Result:= RR.IsVisible(MakePoint(PP9.X, PP9.Y));
  //myGraphic := TGPGraphics.Create(aCanvas.Handle);                //
  //SolidBrush:= TGPSolidBrush.Create(MakeColor(2, 0, 144, 144));   //
  //myGraphic.FillRegion(SolidBrush, RR);                           //
  //SolidBrush.Free;                                                //
}
end;

procedure tPointList.Reverse;
var i: Integer;
```

```
    Temp: tList;
begin
  Temp:= tList.Create;
  for i:= Count- 1 downto 0 do Temp.Add(Items[i]);
  Clear;
  for i:= 0 to Temp.Count- 1 do Add(Temp.Items[i]);
  Temp.Free;
end;

function tPointList.OnConnect(aPoint: tPoint): Boolean;
var i: Integer;
begin
  Result:= False;
  for i:= 0 to Count- 2 do
  begin
    Result:= OnLineRegion(tDrawPoint(Items[i]).Value,
                          tDrawPoint(Items[i+1]).Value, aPoint);
    if Result then
    begin
      Drawing.Mouse.ConPoints:= Point(i, i+1);
      Break;
    end
  end;
  if (not Result)and(tDrawComponent(fOwner).Closed) then
  begin
    Result:= OnLineRegion(tDrawPoint(Items[0]).Value,
                          tDrawPoint(Items[Count-1]).Value, aPoint);
    if Result then
      Drawing.Mouse.ConPoints:= Point(0, Count-1);
  end;
end;

procedure tPointList.PutGrips;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawPoint(Items[i]).PutGrips;
end;

procedure tPointList.ReadFromString(aString: String);
var A1, A2: tAwkLine;
    i: Integer;
begin
  if Count<> 0 then
  begin
    for i:= 0 to Count- 1 do tDrawPoint(Items[i]).Free;
    Clear;
  end;
  A1:= tAwkLine.Create(aString, '/');
  for i:= 0 to A1.Count- 1 do
  begin
    A2:= tAwkLine.Create(A1[i], '*');
    if A2.Count< 2 then
    begin
      A2.Free;
      Continue;
    end;
    Add(tDrawPoint.CreateBy(Make3DPoint(StrToFloat(A2[0]), StrToFloat(A2[1]), StrToFloat(A2
[2])), fOwner));
    A2.Free;
  end;
  A1.Free;
```

```
end;

procedure tPointList.RectGripDrag(aPoint: tDrawPoint; NewCoord: t3dPoint);
var P0, P1, P2, P3: Integer; //indices
    PPP: t3dPoint;
begin
  P0:= IndexOf(aPoint);
  if P0< 3 then P1:= P0+ 1 else P1:= 0;
  if P1< 3 then P2:= P1+ 1 else P2:= 0;
  if P2< 3 then P3:= P2+ 1 else P3:= 0;
  if not Drawing.Command.vFloatLive then                                    //it's a
flag that vFloat is setted
  begin
    Drawing.Command.vFloat:= Angle(tDrawPoint(Items[P0]).Value,             //storing
angle in a global variable
                                   tDrawPoint(Items[P1]).Value);
    Drawing.Command.vFloatLive:= True;
  end;
  tDrawPoint(Items[P0]).Value:= NewCoord;
  Inter(tDrawPoint(Items[P2]).Value,
        Polar(tDrawPoint(Items[P2]).Value,
              Drawing.Command.vFloat, 10000),
        tDrawPoint(Items[P0]).Value,
        Polar(tDrawPoint(Items[P0]).Value,
              Drawing.Command.vFloat- Pi/2, 10000),
        False,
        PPP);
  tDrawPoint(Items[P3]).Value:= PPP;
  Inter(tDrawPoint(Items[P2]).Value,
        Polar(tDrawPoint(Items[P2]).Value,
              Drawing.Command.vFloat- Pi/2, 10000),
        tDrawPoint(Items[P0]).Value,
        Polar(tDrawPoint(Items[P0]).Value,
              Drawing.Command.vFloat, 10000),
        False,
        PPP);
  tDrawPoint(Items[P1]).Value:= PPP
end;

procedure tPointList.RemoveLast;
begin
  tDrawComponent(Items[Count- 1]).Free;
  Delete(Count-1);
  if Count>2 then ;
end;

procedure tPointList.RetrieveValue;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawPoint(Items[i]).Value:= tDrawPoint(Temp.Items[i]).Value;
  Temp.Free;
end;

procedure tPointList.Rotate(aRP: t3dPoint; aRad: Double);
var i: Integer;
    AP, PP1, PP2: tPoint;
begin
  AP:= Point(Round(aRP.X), Round(aRP.Y));
  for i:= 0 to Count- 1 do
  begin
    PP1:= tDrawPoint(Items[i]).Get2dPoint;
```

```
    PP2:= Pol(AP, Ang(AP, PP1)+ aRad, Dist(AP,PP1));
    tDrawPoint(Items[i]).Value.X:= PP2.X;
    tDrawPoint(Items[i]).Value.Y:= PP2.Y;
  end;
end;

procedure tPointList.Rotate3d(MM: tMatrix);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawPoint(Items[i]).Rotate3D(MM);
end;

procedure tPointList.RotateByRubberLine;
var A1, A2: Double;
    FP, LP, SP, RP: tPoint;
begin
  FP:= Point(Round(Drawing.RubberLine.First.X), Round(Drawing.RubberLine.First.Y));
  LP:= Point(Round(Drawing.RubberLine.Last.X), Round(Drawing.RubberLine.Last.Y));
  SP:= Point(Round(Drawing.RubberLine.Second.X), Round(Drawing.RubberLine.Second.Y));
  RP:= Drawing.Inverse(FP);
  A1:= Ang(FP, LP);
  A2:= Ang(FP, SP);
  Rotate(Make3dPoint(RP.X, RP.Y, 0), A2- A1);
end;

procedure tPointList.ScaleByRubberLine;
var D1, D2, SF: Double;
    i: Integer;
    FP, LP, SP, PP1, PP2, RP: tPoint;
begin
  FP:= Point(Round(Drawing.RubberLine.First.X), Round(Drawing.RubberLine.First.Y));
  LP:= Point(Round(Drawing.RubberLine.Last.X), Round(Drawing.RubberLine.Last.Y));
  SP:= Point(Round(Drawing.RubberLine.Second.X), Round(Drawing.RubberLine.Second.Y));
  RP:= Drawing.Inverse(FP);
  D1:= Dist(FP, LP);
  D2:= Dist(FP, SP);
  if D1= 0 then Exit;
  SF:= D2/ D1;
  if SF< 0.01 then Exit;
  for i:= 0 to Count- 1 do
  begin
    PP1:= tDrawPoint(Items[i]).Get2dPoint;
    PP2:= Pol(RP, Ang(RP, PP1), Dist(RP, PP1)* SF);
    tDrawPoint(Items[i]).Value.X:= PP2.X;
    tDrawPoint(Items[i]).Value.Y:= PP2.Y;
  end;
end;

procedure tPointList.SaveValue;
var i: Integer;
begin
  Temp:= tPointList.Create;
  for i:= 0 to Count- 1 do
    Temp.Add(tDrawPoint.CreateBy(tDrawPoint(Items[i]).Value,
                                 tDrawPoint(Items[i]).Component));
end;

procedure tPointList.SetPointType;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
```

```
    tDrawPoint(Items[i]).PointType:= PointType;
end;

procedure tPointList.Shift;
begin
  Add(Items[0]);
  Delete(0);
end;

procedure tPointList.Update;
var i: Integer;
    PP: t3dPoint;
begin
  SetLength(PointArray, Count);
  for i:= 0 to Count- 1 do
  begin
    PP:= tDrawPoint(Items[i]).GetPoint;
    PointArray[i]:= Point(Round(PP.X* Drawing.Zoom.Factor)- Drawing.Mouse.Shift.X,
                          Round(PP.Y* Drawing.Zoom.Factor)- Drawing.Mouse.Shift.Y);
  end;
end;

function tPointList.WriteToString: String;
var i: Integer;
    DP: tDrawPoint;
begin
  if Count= 0 then Exit;
  DP:= tDrawPoint(Items[0]);
  Result:= FloatToStr(DP.Value.X)+ '*'+ FloatToStr(DP.Value.Y)+ '*'+ FloatToStr(DP.Value.Z);
  if Count= 1 then Exit;
  for i:= 1 to Count- 1 do
  begin
    DP:= tDrawPoint(Items[i]);
    Result:= Result+ '/';
    Result:= Result+ FloatToStr(DP.Value.X)+ '*'+ FloatToStr(DP.Value.Y)+ '*'+ FloatToStr
(DP.Value.Z);
  end;
end;

//tDrawPoint

constructor tDrawPoint.CreateBy(aPoint: t3DPoint; aComponent: tComponent);
begin
  Value:= aPoint;
  Component:= aComponent;
  PointType:= ptFree;
end;

function tDrawPoint.Clone(aComponent: tComponent): tDrawPoint;
begin
  Result:= tDrawPoint.CreateBy(Value, aComponent);
  Result.PointType:= PointType;
end;

function tDrawPoint.Get2dPoint: tPoint;
begin
  Result.X:= Round(Value.X);
  Result.Y:= Round(Value.Y);
end;

function tDrawPoint.GetPoint: t3dPoint;
begin
```

```
  Result.X:= Value.X;
  Result.Y:= Value.Y;
  Result.Z:= Value.Z;
end;

procedure tDrawPoint.MoveTo(X, Y, Z: Integer);
begin
  Value.X:= Value.X+ X;
  Value.Y:= Value.Y+ Y;
  Value.Z:= Value.Z+ Z;
  tDrawComponent(Component).Update;
end;

procedure tDrawPoint.PutGrips;
begin
  case PointType of
    ptFree: Drawing.GripList.Add(tFreeGrip.CreateBy(Self));
    ptRect: Drawing.GripList.Add(tRectGrip.CreateBy(Self));
  end;
end;

procedure tDrawPoint.Rotate3D(MM: tMatrix);
var XX, YY, ZZ: Double;
begin
  XX:= Value.X;
  YY:= Value.Y;
  ZZ:= Value.Z;
  Value.X:=MM[1,1] * XX + MM[1,2] * YY + MM[1,3] * ZZ;
  Value.Y:=MM[2,1] * XX + MM[2,2] * YY + MM[2,3] * ZZ;
  Value.Z:=MM[3,1] * XX + MM[3,2] * YY + MM[3,3] * ZZ;
end;

//tBasic

procedure tBasic.Clip(var aStr: String);
begin
  while aStr[1]=' ' do Delete(aStr, 1, 1);
end;

function tBasic.GetCloseString(aString: String): String;
var BB, LL: Integer;
begin
  BB:= Pos('<', aString)+ 1;
  Result:=Copy(aString, BB, Length(aString));
  LL:= Pos(' ', Result);
  Result:= '/'+ Copy(Result, 1, LL-1);
end;

function tBasic.GetName(aName: String): String;
begin
  if (not Assigned(ChildNames)) then
    ChildNames:= tChildNames.Create;
  Result:= ChildNames.GetName(aName);
end;

function tBasic.Indent(Num: Integer): String;
var i: Integer;
begin
  Result:= '';
  for i:= 0 to Num- 1 do
    Result:= Result+ ' ';
end;
```

```
procedure tBasic.AddTree(aTree: tTreeView; aParent: tTreeNode);
var i: Integer;
begin
  if (Assigned(Drawing.Tree) and
      Drawing.Tree.Visible) then
  begin
    Node:= aTree.Items.AddChildObject(aParent, Caption, Self);
    for i:= 0 to ComponentCount- 1 do
      tBasic(Components[i]).AddTree(aTree, Node);
    end;
end;

procedure tBasic.ChangeOwner(aNew: tBasic);
begin
  Owner.RemoveComponent(Self);
  aNew.InsertComponent(Self);
end;

procedure tBasic.ChangePosition(aNew: tBasic);                    //inserting
before aNew
var aList: tList;
    i: Integer;
    aNO: tComponent;
begin
  Owner.RemoveComponent(Self);
  aNO:= aNew.Owner;                                               //save the
owner object
  aList:= tList.Create;                                           //temporary
container for the ownerless objects
  for i:= aNew.ComponentIndex to aNO.ComponentCount-1 do
  begin
    aList.Add(aNO.Components[aNO.ComponentCount-1]);              //add to
list
    aNO.RemoveComponent(aNO.Components[aNO.ComponentCount-1]);    //remove
from owner
  end;
  aNO.InsertComponent(Self);
  for i:= 0 to aList.Count-1 do
    aNO.InsertComponent(tComponent(aList.Items[i]));
  aList.Free;
end;

procedure tBasic.NameChanged;                                     //setting
up the treenode
begin
  try
    if (Assigned(Drawing.Tree) and Drawing.Tree.Visible) then
      Node.Text:= Caption;
  except
    ShowMessage('Problem with tBasic.NameChanged...');
  end;
end;

function tBasic.OnConnect(aPoint: tPoint): tDrawComponent;
var i: Integer;
begin
  Result:= nil;
  for i:= ComponentCount- 1 downto 0 do
  begin
    Result:= tDrawComponent(Components[i]).OnConnect(aPoint);
```

```
      if Assigned(Result) then Break;
    end;
  end;

  function tBasic.OnItem(aPoint: tPoint): tComponent;
  var i: Integer;
  begin
    Result:= nil;
    for i:= ComponentCount- 1 downto 0 do
    begin
      Result:= tDrawComponent(Components[i]).OnItem(aPoint);
      if Assigned(Result) then Break;
    end;
  end;

  procedure tBasic.Paint(aCanvas: tCanvas);
  begin
      inherited;
  {
    for i:= 0 to ComponentCount- 1 do
      tDrawComponent(Components[i]).Paint(aCanvas);
  }
  end;

  procedure tBasic.ReadXml(aString: String);
  var CloseString, S: String;
      AW: tAwkLine;
  begin
    PropList.ReadXml(aString);
    if (className<> 'tlink') and (LinkID> -1) then
      Drawing.LinkList.AddIndex(LinkId, Self);                          //register
  the linksource
    if (Pos('/>', aString)<> 0) then Exit;                              //hasn't
  children
    CloseString:= GetCloseString(aString);
    repeat
      S:= Drawing.DrawingFile.ReadLn;
      if S='' then Break;
      Clip(S);
      AW:= tAwkLine.Create(S, ' ');
      if (Pos('<', AW.Strings[0])<> 0)and
         (Pos('/', AW.Strings[0])= 0) then
      try
        with DrawCompItemList.GetClassType(AW.Strings[0]).CreateBy(Self) do      //creating
  a child object
        begin
          ReadXml(S);
          PointList.SetPointType;
          PointList.Update;
        end;
      except
        ShowMessage('The '+ AW.Strings[0]+ ' didn''t registered for the system');
      end;
      AW.Free;
    until(Pos(CloseString, S)<>0);
  end;

  procedure tBasic.SetCaption(aValue: String);
  begin
    if (Pos('#', aValue) = Length(aValue))
      then Caption:= tBasic(Owner).GetName(Copy(aValue, 1, Length(aValue)-1))
```

```
    else Caption:= aValue;
end;

procedure tBasic.SetPropList;
var LP: tCustomProp;
begin
  with PropList do
  begin
    Add(tNameProp.CreateProp(@Caption, 'Nombre'));
    Add(tColorProp.CreateProp(@Color, 'Color'));
    LP:= tIntegerProp.CreateProp(@LinkID, 'LinkID');
    LP.InternalUse:= True;
    Add(LP);
  end;
end;

procedure tBasic.WriteXML(aIndent: Integer);
var i: Integer;
begin
  Drawing.DrawingFile.Write(Indent(aIndent)+ '<'+ Copy(LowerCase(ClassName), 2, Length
(ClassName)-1));
  if ComponentCount = 0 then
  begin
    PropList.WriteXML(True);
    Exit;
  end;
  PropList.WriteXML(False);
  for i:= 0 to ComponentCount- 1 do
    tBasic(Components[i]).WriteXML(aIndent+ 2);
  Drawing.DrawingFile.WriteLn(Indent(aIndent)+ '</'+ Copy(LowerCase(ClassName), 2, Length
(ClassName)-1)+ '>');
end;

procedure tBasic.WriteCIR(aIndent: Integer);
var
   i : integer;
begin
  if ComponentCount = 0 then
  begin
    Exit;
  end;
  for i:= 0 to ComponentCount- 1 do
    tBasic(Components[i]).WriteCIR(aIndent+ 2);
end;

//tDrawComponent

constructor tDrawComponent.Create(aOwner: tComponent);
begin
  inherited Create(aOwner);
  PointList:= tPointList.CreateBy(Self);
  PropList:= tPropList.CreateBy(Self);                                    //need to
know what is the owner
  LinkID:= -1;
  Layer:= DefaultLayer;                                                    //-1
means not used yet
  SetPropList;
  if Assigned(Owner) then
    Drawing.UndoList.AddBy(tCreateCommand.CreateBy(aOwner, Self));
end;
```



```
constructor tDrawComponent.CreateBy(aOwner: tComponent);
```

```
begin                                                                    // Items
[0]           Items[1]
  Create(aOwner);                                                        //
*------------------*
end;                                                                     // Items
[3]           Items[2]

constructor tDrawComponent.CreateByPoint(aOwner: tComponent; aPoint: t3DPoint);
var i: Integer;                                                          //the
generic layout for the objects
begin                                                                    // Items
[0]           Items[1]
  Create(aOwner);                                                        //
*------------------*
  for i:= 0 to 3
do                                                          //|                |
    PointList.Add(tDrawPoint.CreateBy(aPoint, Self));                    //
*------------------*
  PointList.SetPointType;                                                // Items
[3]           Items[2]
end;

procedure tDrawComponent.AddTree(aTree: tTreeView; aParent: tTreeNode);
var i: Integer;
begin
  if (Assigned(Drawing.Tree) and
      Drawing.Tree.Visible and
      Drawing.CombinationList.LayerEnabled(Layer)) then
  begin
    Node:= aTree.Items.AddChildObject(aParent, Caption, Self);
    for i:= 0 to ComponentCount- 1 do
      tBasic(Components[i]).AddTree(aTree, Node);
    end;
end;

function tDrawComponent.Clone(aOwner: tComponent): tDrawComponent;
var i: Integer;
begin
  Result:= DrawCompItemList.GetClassType(Copy(ClassName, 2, 255)).CreateBy(aOwner);
  PropList.CopyTo(Result.PropList);
  Result.PointList.AssignTo(PointList, Result);
  Result.PointList.Update;
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).Clone(Result);
end;

procedure tDrawComponent.Draw;
begin
  ShowMessage('Do not call this methode: tDrawComponent.Draw');
end;

function tDrawComponent.GetLinkPoint: t3dPoint;
begin
  if PointList.Count> 0
    then Result:= tDrawPoint(PointList.Items[0]).Value
    else Result:= tDrawComponent(Components[0]).GetLinkPoint;
end;

function tDrawComponent.GetRgn: QRegionH;
begin
  Result:= PointList.GetRgn(Closed, Filled);
```

```
end;

procedure tDrawComponent.Mirror(P1, P2: tPoint);
var i: Integer;
begin
  PointList.Mirror(P1, P2);
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).Mirror(P1, P2);
end;

procedure tDrawComponent.MoveTo(X, Y, Z: Integer);
var i: Integer;
begin
  PointList.MoveTo(X, Y, Z);
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).MoveTo(X, Y, Z);
end;

function tDrawComponent.OnConnect(aPoint: tPoint): tDrawComponent;
var aP: tPoint;
    P3, P4, PP: t3dPoint;
    HD: Double;
begin
  if Drawing.CombinationList.LayerEnabled(Layer) then
  begin
    Result:= inherited OnConnect(aPoint);
    if Assigned(Result) then
      Exit;
    if PointList.OnConnect(aPoint)
      then Result:= Self
      else Result:= nil;
    if Assigned(Result) then
    begin
      aP:= Drawing.Inverse(aPoint);
      P3:= tDrawPoint(PointList.Items[Drawing.Mouse.ConPoints.X]).Value;
      P4:= tDrawPoint(PointList.Items[Drawing.Mouse.ConPoints.Y]).Value;
      if (P3.X= P4.X)and (P3.Y= P4.Y) then
         Exit;
      HD:= OsnapSize/ 2/ Drawing.Zoom.Factor;
      if Dist(Point(Round(P3.X), Round(P3.Y)), aP) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= P3;
        Drawing.OsnapShape.OsnapType:= ctEnd;
        Exit;
      end;
      if Dist(aP, Point(Round(P4.X), Round(P4.Y))) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= P4;
        Drawing.OsnapShape.OsnapType:= ctEnd;
        Exit;
      end;
      PP:= Half(P3, P4);
      if Dist(Point(Round(PP.X), Round(PP.Y)), aP) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= PP;
        Drawing.OsnapShape.OsnapType:= ctMid;
        Exit;
      end;
      Drawing.OsnapShape.OsnapPoint:= NearPoint(aP, P3, P4);
      Drawing.OsnapShape.OsnapType:= ctNear;
    end;
  end else OnConnect:= nil;
```

```
end;

function tDrawComponent.OnItem(aPoint: tPoint): tComponent;
var HR: QRegionH;
begin
  if Drawing.CombinationList.LayerEnabled(Layer) then
  begin
    Result:= inherited OnItem(aPoint);
    if Assigned(Result) then Exit;
    HR:= nil;
    try
      HR:= GetRgn;
      //if PtInRegion(HR, aPoint.X, aPoint.Y) then
      if QRegion_contains(HR, PPoint(@aPoint)) then
        Result:= Self;
    finally
      //DeleteObject(HR);
      QRegion_Destroy(HR);
    end;
  end else Result:= nil;
end;

procedure tDrawComponent.OnSelectingRect(aHR: QRegionH; Crossing: Boolean);
var HR: QRegionH;
    i: Integer;
    Box: tRect;
    IsIt: Boolean;
begin
  HR:= nil;
  IsIt:= False;
  if not Crossing then
  begin
    try
      HR:= GetRgn;
      QRegion_boundingRect(HR, @Box);
      if QRegion_contains(aHR, PRect(@Box)) and
         QRegion_contains(aHR, PPoint(@Box.TopLeft)) and
         QRegion_contains(aHR, PPoint(@Box.BottomRight)) then
      begin
        Drawing.SelectList.Add(Self);
        IsIt:= True;
      end;
      {
      if (GetRgnBox(HR, Box)<> 0) and
          PtInRegion(aHR, Box.Left, Box.Top) and
          PtInRegion(aHR, Box.Left, Box.Bottom) and
          PtInRegion(aHR, Box.Right, Box.Top) and
          PtInRegion(aHR, Box.Right, Box.Bottom) then
      begin
        Drawing.SelectList.Add(Self);
        IsIt:= True;
      end;
      }
    finally
      //DeleteObject(HR);
      QRegion_Destroy(HR);
    end;
    if not IsIt then for i:= ComponentCount- 1 downto 0 do
      tDrawComponent(Components[i]).OnSelectingRect(aHR, Crossing);
    Exit;
  end;
```

```
    try
      HR:= GetRgn;
      QRegion_boundingRect(HR, @Box);
      if QRegion_contains(aHR, PRect(@Box)) then
      begin
        Drawing.SelectList.Add(Self);
        IsIt:= True;
      end;
      {
      if (GetRgnBox(HR, Box)<> 0) and RectInRegion(aHR, Box) then
      begin
        Drawing.SelectList.Add(Self);
        IsIt:= True;
      end;
      }
    finally
      //DeleteObject(HR);
      QRegion_Destroy(HR);
    end;
    if IsIt then Exit;
    for i:= ComponentCount- 1 downto 0 do
      tDrawComponent(Components[i]).OnSelectingRect(aHR, Crossing);
  end;

  procedure tDrawComponent.Paint(aCanvas :tCanvas);
  begin
    if not Drawing.CombinationList.LayerEnabled(Layer) then Exit;
    Draw(aCanvas);
    inherited Paint(aCanvas);                                                    //
  painting of children
  end;

  procedure tDrawComponent.PaintTo(aX, aY, aZ: Integer; aCanvas :tCanvas);
  var i: Integer;
  begin
    PointList.SaveValue;
    PointList.MoveTo(aX, aY, aZ);
    PointList.Update;
    Draw(aCanvas);
    PointList.RetrieveValue;
    PointList.Update;
    for i:= 0 to ComponentCount- 1 do
      tDrawComponent(Components[i]).PaintTo(aX, aY, aZ, aCanvas);
  end;

  procedure tDrawComponent.PutGrips;
  begin
    PointList.PutGrips;
  end;

  procedure tDrawComponent.Rotate3d(MM: tMatrix);
  var i: Integer;
  begin
    PointList.Rotate3d(MM);
    for i:= 0 to ComponentCount- 1 do
      tDrawComponent(Components[i]).Rotate3d(MM);
  end;

  procedure tDrawComponent.RotateByRubberLine;
  var i: Integer;
  begin
    PointList.RotateByRubberLine;
```

```
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).RotateByRubberLine;
end;

procedure tDrawComponent.ScaleByRubberLine;
var i: Integer;
begin
  PointList.ScaleByRubberLine;
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).ScaleByRubberLine;
end;

procedure tDrawComponent.SetPropList;
begin
  inherited;
  with PropList do
  begin
    Add(tBooleanProp.CreateProp(@Closed, 'Closed'));
    Add(tBooleanProp.CreateProp(@Filled, 'Filled'));
    Add(tColorProp.CreateProp(@Linecolor, 'Line.Color'));
    Add(tIntegerProp.CreateProp(@Linewidth, 'Line.Width'));
    Add(tPointsProp.CreateProp(@PointList, 'Points'));
    Add(tLayerProp.CreateProp(@Layer, 'Layer'));

    if Assigned(Drawing) then
    begin
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripCopy,
'Copy'));                              //grip menu for grip functions
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripMirror, 'Mirror'));
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripMove, 'Move'));
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripRotate, 'Rotate'));
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripRotate3D_X,
'Rotate3D_X'));                                //grip menu for grip functions
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripRotate3D_Y,
'Rotate3D_Y'));                                //grip menu for grip functions
      Add(tGripMenuProp.CreateProp(Drawing.SelectList.GripScale, 'Scale'));
    end;
  end;
end;

procedure tDrawComponent.SetRubberLine;
var P0, P1: tPoint;
    Ind, Ind2: Integer;
begin
  if not Assigned(Drawing.Mouse.GripHolder) then Exit;
  Ind:= PointList.IndexOf(tDrawPoint(tGrip(Drawing.Mouse.GripHolder).DrawPoint));
  if Ind< (PointList.Count- 1) then Ind2:= Ind+ 1 else Ind2:= 0;
  P0:= Drawing.Adjust(tDrawPoint(PointList.Items[Ind]).Value);
  P1:= Drawing.Adjust(tDrawPoint(PointList.Items[Ind2]).Value);
  Drawing.RubberLine.Start(P0);
  Drawing.RubberLine.ExpandTo(P1);
end;

procedure tDrawComponent.SetupCanvas(aCanvas :tCanvas);
begin
  with aCanvas do
  begin
    Pen.Mode:= pmCOPY;
    Pen.Color:= LineColor;
    Pen.Width:= Round(LineWidth* Drawing.Zoom.Factor);
    if Pen.Width< 1 then Pen.Width:= 1;
```

```
      if (Closed and Filled) then
      begin
        Brush.Style:= bsSolid;
        Brush.Color:= Color;
      end else Brush.Style:= bsClear;
    end;
  end;

  procedure tDrawComponent.SizeTo(aPoint: tPoint);
  begin
    if Drawing.OsnapShape.Visible then
    begin
      tDrawPoint(PointList.Items[1]).Value.X:= Drawing.OsnapShape.OsnapPoint.X;
      tDrawPoint(PointList.Items[2]).Value.X:= Drawing.OsnapShape.OsnapPoint.X;
      tDrawPoint(PointList.Items[2]).Value.Y:= Drawing.OsnapShape.OsnapPoint.Y;
      tDrawPoint(PointList.Items[3]).Value.Y:= Drawing.OsnapShape.OsnapPoint.Y;
    end else
    begin
      tDrawPoint(PointList.Items[1]).Value.X:= aPoint.X;
      tDrawPoint(PointList.Items[2]).Value.X:= aPoint.X;
      tDrawPoint(PointList.Items[2]).Value.Y:= aPoint.Y;
      tDrawPoint(PointList.Items[3]).Value.Y:= aPoint.Y;
    end;
  end;

  procedure tDrawComponent.Update;
  begin
    PointList.Update;
  end;

  //tDrawCompItem

  constructor tDrawCompItem.Create(aName: String; aClassType: tDrawingClass);
  begin
    Name:= aName;
    ClassType:= aClassType;
  end;

  //tDrawCompItemList

  function tDrawCompItemList.GetName(aClassType: tDrawingClass): String;
  var i: Integer;
  begin
    Result:= 'None';
    for i:= 0 to Count- 1 do
      if tDrawCompItem(Items[i]).ClassType = aClassType then
      begin
        Result:= tDrawCompItem(Items[i]).Name;
        Break;
      end;
  end;

  function tDrawCompItemList.GetClassType(aName: String): tDrawingClass;
  var i: Integer;
  begin
    Result:= nil;
    if aName[1]='<' then                                                    //it is a
  name of XML tag
    begin
      System.Delete(aName, 1, 1);
      aName[1]:= UpCase(aName[1]);
    end;
```

```
    try
      for i:= 0 to Count- 1 do
        if tDrawCompItem(Items[i]).Name = aName then
        begin
          Result:= tDrawCompItem(Items[i]).ClassType;
          Break;
        end;
    except
    end;
    if Result = nil then
      ShowMessage('Unknown classname: '+ aName);
end;

end.
```

```
unit CadDevices;

interface

uses Classes, Types, Sysutils, QForms, QGraphics, CadBasic, CommonCAD, CadProperty, CadMisc;

type
    tDevice = class(tDrawComponent)
        procedure DrawShape(X, Y: Integer; aCanvas:TCanvas);
    public
        Algn90 :Integer;
        Algn360:Integer;
        Mirror :Integer;
        Shape  :Integer;
        Zoom   :Real;
        Value: Real;
        NodeList: tPointList;
        constructor Create(aOwner: tComponent); override;
        procedure Draw(aCanvas :tCanvas); override;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tResistance = class(tDevice)
    private
    public
        procedure SetPropList; override;
    end;

    tCapacitance = class(tDevice)
    private
    public
        procedure SetPropList; override;
    end;

    tInductance = class(tDevice)
    private
    public
        procedure SetPropList; override;
    end;

    tDiode = class(tDevice)
    private
    public
        Model : string;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tVSource = class(tDevice)
    private
    public
        Model : string;
        FreqElms : string;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tISource = class(tDevice)
    private
```

```
    public
        Model : string;
        Tipo : string;
        prccedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tBSource = class(tDevice)
    private
    public
        Model : string;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tDSource = class(tDevice)
    private
    public
        Model : string;
        Tipo : string;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tAnGround = class(tDevice)
    private
    public
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tBJT = class(tDevice)
    private
    public
        Model : string;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

    tSubcircuit = class(tDevice)
    private
    public
        Model : string;
        PortList: tPointList;
        procedure SetPropList; override;
        procedure WriteCIR(aIndent: Integer); override;
    end;

var Shapes: TStrings;

implementation

constructor tDevice.Create(aOwner: tComponent);
begin
  with Drawing.ChildNames do
  begin
    if (Pos('Voltage Source', Text) > 0) then
    begin
      Delete(Count-1);
      Shape:=10;
    end;
```

```
    if (Pos('Current Source', Text) > 0) then
    begin
      Delete(Count-1);
      Shape:=11;
    end;
    if (Pos('Grounded Source', Text) > 0) then
    begin
      Delete(Count-1);
      Shape:=12;
    end;
    if (Pos('AC Source', Text) > 0) then
    begin
      Delete(Count-1);
      Shape:=8;
      Algn90:=0;
    end;
  end;
  inherited Create(aOwner);
  NodeList:= tPointList.CreateBy(Self);
  if ClassNameIs('tAnground') then
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(0, -1, -1), aOwner))
  else if ClassNameIs('tVsource') and (Shape=12) then
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(0, -1, -1), aOwner))
  else if ClassNameIs('tDsource') or ClassNameIs('tSubcircuit') then
  begin
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(-1, -1, -1), aOwner));
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(-1, -1, -1), aOwner));
  end
  else if ClassNameIs('tPolyline') then
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(ChildNames.GetNumber(Caption), -1, -1),
aOwner))
  else
     NodeList.Add(tDrawPoint.CreateBy(Make3DPoint(-1, -1, -1), aOwner));
end;

procedure tDevice.DrawShape(X, Y: Integer; aCanvas:TCanvas);
var X1 , Y1 , X2 , Y2 , I: Integer;

    Datos: PChar;


begin

     if (Shapes=nil) then

     begin

        Shapes:= TStringList.Create();

        if FileExists(ExtractFileDir(Application.ExeName)+'\circuitfiles\devices.dat') then

           Shapes.LoadFromFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\devices.
dat')

        else

        begin

           Application.MessageBox('El archivo devices.dat no existe por favor reinstale la
aplicación', 'Advertencia', [smbOK]);

           Halt(1);
```

```
        end;
      end;

      Datos:= PChar(Shapes[Shape-1]);
      I := 0; X2:= X; Y2:= Y;
      aCanvas.MoveTo(X2,Y2);
      While (Datos[I]<>Chr(0)) do
      begin
         X1:=X2; Y1:=Y2;
         X2:=X2 + Round(Zoom*Algn360*Abs(Algn90-1)*(Ord(Datos[I+1])-128));
         Y2:=Y2 + Round(Zoom*Algn360*Abs(Algn90  )*(Ord(Datos[I+1])-128));
         X2:=X2 + Round(Zoom*Algn360*Abs(Algn90  )*(Ord(Datos[I+2])-128));
         Y2:=Y2 + Round(Zoom*Algn360*Abs(Algn90-1)*(Ord(Datos[I+2])-128));

         if Ord(Datos[I]) = 128 Then aCanvas.MoveTo(X2,Y2);
         if Ord(Datos[I]) = 129 Then aCanvas.LineTo(X2,Y2);
       //if Ord(Datos[I]) = 130 Then aCanvas.Arc(X1-Y2+Y1,Y1-X2+X1,X2+Y2-Y1,Y2+X2-X1,X1,Y1,
X2,Y2);
         if Ord(Datos[I]) = 130 Then aCanvas.Arc(X1-Y2+Y1,Y1-X2+X1,(X2-X1+Y2-Y1)*(1+Abs
(Algn90)),(X2-X1+Y2-Y1)*(1+Abs(Algn90-1)),-8*180*Abs(Algn90),16*180*Round((X2-X1+Y2-Y1)/Abs
(X2-X1+Y2-Y1)));
         if Ord(Datos[I]) = 131 Then aCanvas.Ellipse(Round(X1-(Y2-Y1)/2),Round(Y1-(X2-X1)/2),
Round(X2+(Y2-Y1)/2),Round(Y2+(X2-X1)/2));
         I:=I+3;
      end;
end;

procedure tDevice.Draw(aCanvas: tCanvas);
var
    OriginX, OriginY: Integer;
    L, R: tPoint;
begin
  L:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).GetPoint);
  Zoom:=Drawing.Zoom.Factor/0.01*0.5;
  OriginX:=L.X + Round(048*Algn90*Zoom+(Algn360-1)*Zoom*63*(Algn90-1));
  OriginY:=L.Y + Round(-048*(Algn90-1)*Zoom+(Algn360-1)*Zoom*-63*Algn90);
  Filled:=False;
```

```
  SetupCanvas(aCanvas);

  DrawShape(OriginX, OriginY, aCanvas);

  Filled:=True;

  PointList.Update;

  With Drawing do
  if (Command.Status=csCreate) and (Command.Element = Self) then
  begin
      if Algn90=0 Then R.X:= L.X + Round(126*Self.Zoom) Else R.X:= L.X + Round(96*Self.
Zoom);
      if Algn90=0 Then R.Y:= L.Y + Round(96*Self.Zoom) Else R.Y:= L.Y + Round(126*Self.
Zoom);
      Command.Element.SizeTo(Inverse(R));
      Mouse.Holder:=Self;
      SelectList.AddBy(Mouse.Holder, false);
      Command.Status:= csObjectMove;
      Mouse.ActionPoint:= Mouse.Pos;
  end;

end;

procedure tDevice.SetPropList;
var LP: tCustomProp;
begin
  Closed:= True;
  Filled:= True;
  Algn360:=1;
  inherited;
  with PropList do
  begin
    Kill('Color');
    Kill('Line.color');
    Kill('Line.width');
    Kill('Filled');
    Kill('Closed');
    Kill('Layer');
    Add(tPointsProp.CreateProp(@NodeList, 'Nodos'));
    LP:= tCustomProp(Find('Points'));
    LP.InternalUse:= True;
    LP:= tIntegerProp.CreateProp(@Algn90, 'Algn90');
    LP.InternalUse:= True;
    Add(LP);
    LP:= tIntegerProp.CreateProp(@Algn360, 'Algn360');
    LP.InternalUse:= True;
    Add(LP);
    LP:= tIntegerProp.CreateProp(@Mirror, 'Mirror');
    LP.InternalUse:= True;
    Add(LP);
  end;
  PointList.PointType:= ptRect;
end;

procedure tDevice.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc, Text : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
```

```
    AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
    for i:= 0 to AW1.Count- 1 do
    begin
        if (AW1[i]<>'-1') then
            Desc:= Desc + ' ' +AW1[i];
    end;
    AW1.Free;
    Str(Value, Text);
    Desc := Desc + ' ' + Text;
    Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tResistance.SetPropList;
begin
  SetCaption('R#');
  Shape:=1;
  Algn90:=0;
  inherited;
  with PropList do
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
end;

procedure tCapacitance.SetPropList;
begin
  SetCaption('C#');
  Shape:=2;
  Algn90:=0;
  inherited;
  with PropList do
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
end;

procedure tInductance.SetPropList;
begin
  SetCaption('L#');
  Shape:=7;
  Algn90:=0;
  inherited;
  with PropList do
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
end;

procedure tDiode.SetPropList;
begin
  SetCaption('D#');
  Shape:=3;
  Algn90:=0;
  Model:='diode1';
  inherited;
  with PropList do
      Add(tStringProp.CreateProp(@Model, 'Modelo'));
end;

procedure tDiode.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   for i:= 0 to AW1.Count- 1 do
```

```
    begin
        if (AW1[i]<>'-1') then
            Desc:= Desc + ' ' +AW1[i];
    end;
    AW1.Free;
    if Analisis.Tipo then
       Drawing.DrawingFile.WriteLn('R'+Desc+' 1E+16');
    Desc := Desc + ' ' + Model;
    Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tVSource.SetPropList;
var LP: tCustomProp;
begin
  SetCaption('V#');
  if (Shape<>8) and (Shape<>12) then Shape:=4;
  if (Shape<>8) then Algn90:=1;
  inherited;
  with PropList do
  begin
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
      LP:=tStringProp.CreateProp(@Model, 'Función');
      if (Shape<>8) then
      LP.InternalUse:=True;
      Add(LP);
      LP:=tStringProp.CreateProp(@FreqElms, 'Freq'+#09+'Elements');
      Add(LP);
      LP:=tIntegerProp.CreateProp(@Shape, 'Simbolo');
      LP.InternalUse:=True;
      Add(LP);
  end;
end;

procedure tVsource.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc, Text : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   for i:= 0 to AW1.Count- 1 do
   begin
       if (AW1[i]<>'-1') then
           Desc:= Desc + ' ' +AW1[i];
   end;
   AW1.Free;
   Str(Value, Text);
   if (Shape=8) then
      Desc := Desc + ' AC ' + Text + ' ' + Model
   else
      Desc := Desc + ' DC ' + Text;
   Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tISource.SetPropList;
var
    TipoFuente :tStringList;

begin
  SetCaption('I#');
  Shape:=9;
```

```
  Algn90:=1;
  inherited;
  Algn360:=-1;
  TipoFuente:= tStringList.Create;
  with TipoFuente do
  begin
    Add('DC');
    Add('AC');
  end;
  Tipo:=TipoFuente.Strings[0];
  with PropList do
  begin
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
      Add(tStringProp.CreateProp(@Model, 'Función'));
      Add(tListProp.CreateListProp(@Tipo, 'Tipo', TipoFuente));
  end;
end;

procedure tIsource.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc, Text : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   for i:= 0 to AW1.Count- 1 do
   begin
       if (AW1[i]<>'-1') then
           Desc:= Desc + ' ' +AW1[i];
   end;
   AW1.Free;
   Str(Value, Text);
   Desc := Desc + ' ' + Tipo + ' ' + Text + ' ' + Model;
   Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tBSource.SetPropList;
begin
  SetCaption('B#');
  Shape:=13;
  Algn90:=1;
  inherited;
  with PropList do
  begin
      Add(tStringProp.CreateProp(@Model, 'Función'));
  end;
end;

procedure tBsource.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   for i:= 0 to AW1.Count- 1 do
   begin
       if (AW1[i]<>'-1') then
           Desc:= Desc + ' ' +AW1[i];
   end;
```

```
    AW1.Free;
    if Analisis.Tipo then
       Drawing.DrawingFile.WriteLn('R'+Desc+' 1E+16');
    Desc := Desc + ' ' + Model;
    Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tDSource.SetPropList;
var
    TipoFuente :tStringList;
    LP: tCustomProp;
begin
  if (Shape<>10) then Shape:=11;
  if (Shape<>10) then SetCaption('G#')
                 else SetCaption('H#');
  Algn90:=1;
  inherited;
  TipoFuente:= tStringList.Create;
  with TipoFuente do
  begin
    Add('V');
    Add('I');
  end;
  Tipo:=TipoFuente.Strings[0];
  with PropList do
  begin
      Add(tFloatProp.CreateProp(@Value, 'Valor'));
      Add(tStringProp.CreateProp(@Model, 'Amperimetro'));
      Add(tListProp.CreateListProp(@Tipo, 'Controlada_por', TipoFuente));
      LP:=tIntegerProp.CreateProp(@Shape, 'Simbolo');
      LP.InternalUse:=True;
      Add(LP);
  end;
end;

procedure tDsource.WriteCIR(aIndent: Integer);
var
   AW1, AW2 : tAwkLine;
   Desc, Text : string;
   i, j : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW2:= tAwkLine.Create(NodeList.WriteToString, '/');
   for j:= 0 to AW2.Count- 1 do
   begin
      AW1:= tAwkLine.Create(AW2[j] , '*');
      for i:= 0 to AW1.Count- 1 do
      begin
          if (AW1[i]<>'-1') then
              Desc:= Desc + ' ' +AW1[i];
      end;
      AW1.Free;
   end;
   AW2.Free;
   Str(Value, Text);
   if Shape=10 then
      if Tipo='V' then
        Desc := 'E'+ Desc + Text
      else
        Desc := 'H'+ Desc + ' ' + Model + ' ' + Text;
   if Shape=11 then
      if Tipo='V' then
```

```
        Desc := 'G'+ Desc + Text
      else
        Desc := 'F'+ Desc + ' ' + Model + ' ' + Text;
   Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tAnGround.SetPropList;
begin
  SetCaption('Gnd#');
  Shape:=5;
  Algn90:=1;
  inherited;
end;

procedure tAnGround.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc: string;
   i : integer;
begin
   Desc := Indent(aIndent)+ 'V' + Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   for i:= 0 to AW1.Count- 1 do
   begin
       if (AW1[i]<>'-1') then
           Desc:= Desc + ' ' +AW1[i];
   end;
   AW1.Free;
   Desc := Desc + ' 0';
   Drawing.DrawingFile.WriteLn(Desc);
end;

procedure tBJT.SetPropList;
begin
  SetCaption('Q#');
  Shape:=6;
  Algn90:=0;
  Model:='bjt1';
  inherited;
  with PropList do
      Add(tStringProp.CreateProp(@Model, 'Modelo'));
end;

procedure tBJT.WriteCIR(aIndent: Integer);
var
   AW1 : tAwkLine;
   Desc : string;
   i : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW1:= tAwkLine.Create(NodeList.WriteToString, '*');
   if Analisis.Tipo then
      Drawing.DrawingFile.WriteLn('R'+Desc+' '+AW1[0]+' '+AW1[2]+' 1E+16');
   for i:= 0 to AW1.Count- 1 do
   begin
       if (AW1[i]<>'-1') then
           Desc:= Desc + ' ' +AW1[i];
   end;
   AW1.Free;
   Desc := Desc + ' ' + Model;
   Drawing.DrawingFile.WriteLn(Desc);
end;
```

```
procedure tSubcircuit.SetPropList;
begin
  SetCaption('X#');
  Shape:=15;
  Algn90:=0;
  Model:='subcircuit';
  PortList:= tPointList.CreateBy(Self);
  PortList.Add(tDrawPoint.CreateBy(Make3DPoint(-1, -1, -1), Self));
  PortList.Add(tDrawPoint.CreateBy(Make3DPoint(-1, -1, -1), Self));
  inherited;
  with PropList do
  begin
      Add(tStringProp.CreateProp(@Model, 'Archivo Netlist'));
      Add(tPointsProp.CreateProp(@PortList, 'Puertos'));
  end;
end;

procedure tSubcircuit.WriteCIR(aIndent: Integer);
var
   AW1, AW2 : tAwkLine;
   Models, Subckt : TStrings;
   Desc : string;
   i,j : integer;
begin
   Desc := Indent(aIndent)+ Caption;
   AW2:= tAwkLine.Create(NodeList.WriteToString, '/');
   for j:= 0 to AW2.Count- 1 do
   begin
      AW1:= tAwkLine.Create(AW2[j] , '*');
      for i:= 0 to AW1.Count- 1 do
      begin
          if (AW1[i]<>'-1') then
              Desc:= Desc + ' ' +AW1[i];
      end;
      AW1.Free;
   end;
   AW2.Free;
   Desc := Desc + ' ' + Model;
   Drawing.DrawingFile.WriteLn(Desc);

   Models:= TStringList.Create;
   Subckt:= TStringList.Create;
   try
   Models.LoadFromFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\models.cir');
   if (Pos(Model, Models.Text) <= 0) then
   begin
   Desc := Indent(aIndent)+ '.SUBCKT';
   Desc := Desc + ' ' + Model;
   AW2:= tAwkLine.Create(PortList.WriteToString, '/');
   for j:= 0 to AW2.Count- 1 do
   begin
      AW1:= tAwkLine.Create(AW2[j] , '*');
      for i:= 0 to AW1.Count- 1 do
      begin
          if (AW1[i]<>'-1') then
              Desc:= Desc + ' ' +AW1[i];
      end;
      AW1.Free;
   end;
   AW2.Free;
   Models.Add(Desc);
```

```
    if FileExists(Model+'.cir') then
    begin
       Subckt.LoadFromFile(Model+'.cir');
       for i:=0 to Subckt.Count-1 do
           Models.Add(Subckt[i]);
    end
    else
       Models.Add('.INCLUDE '+Model+'.cir');
    Models.Add('.ENDS ' +Model);
    Models.Add('');
    Models.SaveToFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\models.cir');
    end;
    finally
       Models.Free;
       Subckt.Free;
    end;
end;

initialization

  if (not Assigned(DrawCompItemList)) then
    DrawCompItemList:= tDrawCompItemList.Create;                              //
DrawCompItemList have to be assigned
  DrawCompItemList.Add(tDrawCompItem.Create('Resistance', tResistance));      //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Capacitance', tCapacitance));    //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Inductance', tInductance));      //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Diode', tDiode));                //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Vsource', tVSource));          //registering
of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Isource', tISource));          //registering
of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Bsource', tBSource));          //registering
of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Dsource', tDSource));          //registering
of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Anground', tAnGround));          //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Subcircuit', tSubcircuit));          //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Bjt', tBJT));                //registering of
drawing classes

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadGrip;

interface

uses Classes, QGraphics, QT, Types,
     CadMisc;

type
  tGripList = class(tList)                                          //
collection of grips
  private
  public
    procedure AddBy(aComp: tComponent);
    procedure Clear; override;
    function OnItem(aPoint: tPoint): tObject;
    procedure Paint(aCanvas: tCanvas);
    procedure Synchronize;
  end;

  tGrip = class                                                     //base grip
class
  private
    function GetRgn: QRegionH;
  public
    DrawPoint: tObject;
    constructor CreateBy(aDrawPoint: tObject);
    procedure DragTo(aPoint: tPoint); virtual; abstract;
    function OnItem(aPoint: tPoint): tObject;
    procedure Paint(aCanvas: tCanvas);
  end;

  tRectGrip = class(tGrip)                                          //inherited
grip class for "rect style" object
  private
  public
    procedure DragTo(aPoint: tPoint); override;
  end;

  tFreeGrip = class(tGrip)                                          //inherited
grip class for "polyline style" object
  private
  public
    procedure DragTo(aPoint: tPoint); override;
  end;

var
  HalfSize: Integer;

implementation

uses CadBasic, CadSelect, CommonCAD, QForms;
```

```
//tGripList

procedure tGripList.AddBy(aComp: tComponent);
begin
  tDrawComponent(aComp).PutGrips;
end;

procedure tGripList.Clear;
begin
  while Count> 0 do
  begin
    tGrip(Items[0]).Free;
    Delete(0);
  end;
end;

function tGripList.OnItem(aPoint: tPoint): tObject;
var i: Integer;
begin
  Result:= nil;
  for i:= 0 to Count- 1 do
  begin
    Result:= tGrip(Items[i]).OnItem(aPoint);
    if Assigned(Result) then Break;
  end;
end;

procedure tGripList.Paint(aCanvas: tCanvas);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tGrip(Items[i]).Paint(aCanvas);
end;

procedure tGripList.Synchronize;
var i: Integer;
begin
  Clear;
  for i:= 0 to Drawing.SelectList.Count- 1 do
    AddBy(tDrawComponent(Drawing.SelectList.Items[i]));
  Drawing.Invalidate;
end;

//tGrip

function tGrip.GetRgn: QRegionH;
var PP: tPoint;
begin
  PP:= Drawing.Adjust(tDrawPoint(DrawPoint).GetPoint);
  Result:= Qregion_create(PP.X- HalfSize, PP.Y- HalfSize, HalfSize*2, HalfSize*2,
QRegionRegionType_Rectangle);
end;

constructor tGrip.CreateBy(aDrawPoint: tObject);
begin
  DrawPoint:= aDrawPoint;
end;

function tGrip.OnItem(aPoint: tPoint): tObject;
var HR: QRegionH;
begin
  Result:= nil;
```

```
    HR:= nil;
    try
      HR:= GetRgn;
      if QRegion_contains(HR, PPoint(@aPoint)) then
        Result:= Self;
    finally
      //DeleteObject(HR);
      QRegion_Destroy(HR);
    end;
  end;

  procedure tGrip.Paint(aCanvas :tCanvas);
  var PP: tPoint;
  begin
    PP:= Drawing.Adjust(tDrawPoint(DrawPoint).GetPoint);
    with aCanvas do
    begin
      Pen.Color:= clWhite;
      Pen.Width:= 1;
      Brush.Style:= bsSolid;
      Brush.Color:= clBlack;
      Rectangle(PP.X- HalfSize, PP.Y- HalfSize, PP.X+ HalfSize, PP.Y+ HalfSize);
    end;
  end;

  //tRectGrip

  procedure tRectGrip.DragTo(aPoint: tPoint);
  begin
      tDrawComponent(tDrawPoint(DrawPoint).Component).PointList.RectGripDrag(
        tDrawPoint(DrawPoint), Drawing.Inverse3D(Drawing.Mouse.Pos));
  end;

  //tFreeGrip

  procedure tFreeGrip.DragTo(aPoint: tPoint);
  begin
    if Drawing.OsnapShape.Visible
      then tDrawPoint(DrawPoint).Value:= Drawing.OsnapShape.OsnapPoint
      else tDrawPoint(DrawPoint).MoveTo(aPoint.X, aPoint.Y, 0);
  end;

  initialization

    HalfSize:= 3;

  end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadLayer;

interface

uses
  QT, SysUtils, Classes, QGraphics, QControls, QForms, Types,
  QDialogs, QComCtrls, QCheckLst, QExtCtrls, QStdCtrls, QMenus, QTypes;

type
  tLayer = class
  public
    Caption: String;
    constructor CreateByName(aName: String);
    procedure ReadFromString(aString: String);
    function WriteToString: String;
  end;

  tCombination = class(TList)
  public
    Caption: String;
    constructor CreateByName(aName: String);
    procedure AddLayer(aLayer: tLayer);
    procedure ChangeCaption(Old, New: String); virtual;
    procedure DeleteAll; virtual;
    procedure DelLayer(aLayerName: String); virtual;
    procedure DeployLayers(aListBox: tListBox);
    function LayerEnabled(aLayerName: String): Boolean;
    procedure ReadFromString(aString: String); virtual;
    function WriteToString: String;
  end;

  tMainCombination = class(tCombination)                    //the main
combination is the _All_
  public                                                    //it is
contains the all of layers
    function AddByLayerName(aLayerName: String): tLayer;
    procedure DeleteAll; override;
    procedure DelLayer(aLayerName: String); override;
    function GetLayer(aLayerName: String): tLayer;
    procedure ReadFromString(aString: String); override;
  end;

  tCombinationList = class(TList)                           //contains
the combinations
    fSelected: tCombination;
    function GetSelected: tCombination;
    procedure SetSelected(aValue: tCombination);
  public
    ItemIndex: Integer;
    MainCombination: tMainCombination;                      //this
contains all used layers
    procedure AddLayer(aCombinationName, aLayerName: String);
```

```
    procedure ChangeCaption(Old, New: String);
    procedure Deploy(aListBox1, aListBox2: tListBox);
    function GetCombination(aCaption: String): tCombination;
    function LayerEnabled(aLayerName: String): Boolean;
    procedure ReadFromString(aString: String);
    function WriteToString: String;
    property Selected: tCombination read GetSelected write SetSelected;
  end;

  TLayersDialog = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    ListBox1: TListBox;
    Splitter1: TSplitter;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    PopupMenu1: TPopupMenu;
    Delete1: TMenuItem;
    New1: TMenuItem;
    PopupMenu2: TPopupMenu;
    Delete2: TMenuItem;
    New2: TMenuItem;
    Edit1: TMenuItem;
    Edit2: TMenuItem;
    ListBox2: TListBox;
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Delete1Click(Sender: TObject);
    procedure Edit1Click(Sender: TObject);
    procedure New1Click(Sender: TObject);
    procedure ListBox1Click(Sender: TObject);
    procedure Delete2Click(Sender: TObject);
    procedure Edit2Click(Sender: TObject);
    procedure New2Click(Sender: TObject);
    procedure ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
    MenuItem: tMenuItem;
    procedure Refresh;
  end;

const
  DefaultLayer = '_Default_';
  DefaultCombination = '_All_';                                    //contains
all layers

var
  LayersDialog: TLayersDialog;
  CombinationList: tCombinationList;

implementation

{$R *.xfm}

B-58  uses CadMisc, CadProperty, CommonCAD;
```

```
//tLayer

constructor tLayer.CreateByName(aName: String);
begin
  Create;
  Caption:= aName;
end;

procedure tLayer.ReadFromString(aString: String);
begin
  Caption:= aString;
end;

function tLayer.WriteToString: String;
begin
  Result:= Caption;
end;

//tCombination

constructor tCombination.CreateByName(aName: String);
begin
  Create;
  Caption:= aName;
end;

procedure tCombination.AddLayer(aLayer: tLayer);
var i: Integer;
    isit: Boolean;
begin
  isit:= False;
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]) = aLayer then
    begin
      isit:= True;
      Break;
    end;
  if isit then Exit;
  Add(aLayer);
end;

                                                                    //warning!
procedure tCombination.ChangeCaption(Old, New: String);             //changing
of a layername caused a breaking
var i: Integer;                                                     //the
contact of the layer contained elements
begin
  if Old = DefaultLayer then Exit;                                  //should
not change default
  if LayerList.IndexOf(New)> -1 then Exit;                          //existing
layername
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption = Old then
    begin
      tLayer(Items[i]).Caption:= New;
      Break;
    end;
  LayerList.Clear;
  for i:= 0 to Count- 1 do
    LayerList.Add(tLayer(Items[i]).Caption);
end;
```

```
procedure tCombination.DeleteAll;
begin
  Clear;
end;

procedure tCombination.DelLayer(aLayerName: String);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption = aLayerName then
      Break;
  Delete(i);
end;

procedure tCombination.DeployLayers(aListBox: tListBox);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    aListBox.Items.Add(tLayer(Items[i]).Caption);
end;

function tCombination.LayerEnabled(aLayerName: String): Boolean;
var i: Integer;
begin
  Result:= False;
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption= aLayerName then
    begin
      Result:= True;
      Break;
    end;
end;

procedure tCombination.ReadFromString(aString: String);
var AW: tAwkLine;
    i: Integer;
begin
  AW:= tAwkLine.Create(aString, '*');
  Caption:= AW.Strings[0];
  for i:= 1 to AW.Count- 1 do
    Add(CombinationList.MainCombination.GetLayer(AW.Strings[i]));
  AW.Free;
end;

function tCombination.WriteToString: String;
var i: Integer;
begin
  Result:= Caption;
  for i:= 0 to Count- 1 do
    Result:= Result+ '*'+ tLayer(Items[i]).WriteToString;
end;

//tMainCombination

function tMainCombination.AddByLayerName(aLayerName: String): tLayer;
var i: Integer;
    isit: Boolean;
begin
  isit:= False;
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption = aLayerName then
```

```
    begin
      isit:= True;
      Break;
    end;
  Result:= nil;
  if isit then Exit;
  Result:= tLayer.CreateByName(aLayerName);
  Add(Result);
  CadProperty.LayerList.Add(aLayerName)
end;

procedure tMainCombination.DeleteAll;
var i: Integer;
    Layer: tLayer;
begin
  for i:= 0 to Count- 1 do
  begin
    Layer:= tLayer(Items[0]);
    Delete(0);
    Layer.Free;
  end;
end;

                                                                    //warning!
procedure tMainCombination.DelLayer(aLayerName: String);            //deleting
a layer, the components on its
var i: Integer;                                                     //awailable
only by the treeview to setting
    Layer: tLayer;                                                  //up the
layer property
begin
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption = aLayerName then
      Break;
  Layer:= tLayer(Items[i]);
  Delete(i);
  Layer.Free;
end;

function tMainCombination.GetLayer(aLayerName: String): tLayer;
var i: Integer;
begin
  Result:= nil;
  for i:= 0 to Count- 1 do
    if tLayer(Items[i]).Caption= aLayerName then
    begin
      Result:= tLayer(Items[i]);
      Break;
    end;
end;

procedure tMainCombination.ReadFromString(aString: String);
var AW: tAwkLine;
    i: Integer;
    LA: tLayer;
begin
  AW:= tAwkLine.Create(aString, '*');
  Caption:= AW.Strings[0];
  for i:= 1 to AW.Count- 1 do
  begin
    LA:= AddByLayerName(AW.Strings[i]);;
    LA.ReadFromString(AW.Strings[i]);
```

```
    end;
    AW.Free;
  end;

  //tCombinationList

  procedure tCombinationList.AddLayer(aCombinationName, aLayerName: String);
  var Comb: tCombination;
      i: Integer;
  begin
    if aCombinationName = DefaultCombination then
    begin
      MainCombination.AddByLayerName(aLayerName);
      Exit;
    end;
    Comb:= nil;
    for i:= 0 to Count- 1 do
      if tCombination(Items[i]).Caption = aCombinationName then
      begin
        Comb:= tCombination(Items[i]);
        Break;
      end;
    if not Assigned(Comb) then
    begin
      ShowMessage('Combination.Name is missed');
      Exit;
    end;
    Comb.AddLayer(MainCombination.GetLayer(aLayerName));
  end;

  function tCombinationList.GetSelected: tCombination;
  begin
    Result:= fSelected;
  end;

  procedure tCombinationList.SetSelected(aValue: tCombination);
  begin
    fSelected:= aValue;
    ItemIndex:= IndexOf(aValue);
    Drawing.Invalidate;
  end;

  procedure tCombinationList.ChangeCaption(Old, New: String);
  var i: Integer;
      isit: Boolean;
  begin
    if Old = DefaultCombination then Exit;                          //shouldn't
  change it
    if Old = New then Exit;                                         //nothing
  to do
    isit:= False;
    for i:= 0 to Count- 1 do
      if tCombination(Items[i]).Caption = New then
      begin
        ShowMessage('Existing name');
        isit:= True;
        Break;
      end;
    if isit then Exit;                                              //existing
  name
    for i:= 0 to Count- 1 do
```

```
    if tCombination(Items[i]).Caption = Old then
    begin
      tCombination(Items[i]).Caption:= New;
      Break;
    end;
end;

procedure tCombinationList.Deploy(aListBox1, aListBox2: tListBox);
var i: Integer;
begin
  aListBox1.Clear;
  aListBox2.Clear;
  for i:= 0 to Count- 1 do
    aListBox1.Items.Add(tCombination(Items[i]).Caption);
  aListBox1.ItemIndex:= ItemIndex;
  if not Assigned(Selected) then
     Selected:= tCombination(Items[0]);
  Selected.DeployLayers(aListBox2);
end;

function tCombinationList.GetCombination(aCaption: String): tCombination;
var i: Integer;
begin
  Result:= nil;
  for i:= 0 to Count- 1 do
    if tCombination(Items[i]).Caption = aCaption then
    begin
      Result:= tCombination(Items[i]);
      Break;
    end;
end;

function tCombinationList.LayerEnabled(aLayerName: String): Boolean;
begin
  if Assigned(Selected)
    then Result:= Selected.LayerEnabled(aLayerName)
    else Result:= False;
end;

procedure tCombinationList.ReadFromString(aString: String);
var AW: tAwkLine;
    i: Integer;
    Co: tCombination;
begin
  for i:= 0 to Count- 1 do
  begin
    Co:= tCombination(Items[0]);
    Co.DeleteAll;
    Co.Free;
    Delete(0);
  end;
  Clear;
  AW:= tAwkLine.Create(aString, '/');
  Co:= tMainCombination.Create;
  Co.ReadFromString(AW.Strings[0]);                                    //the first
textitem is the maincombination
  Add(Co);
  MainCombination:= tMainCombination(Co);
  for i:= 1 to AW.Count- 2 do
  begin
    Co:= tCombination.Create;
    Add(Co);
```

```
    Co.ReadFromString(AW.Strings[i]);
  end;
  Selected:= GetCombination(AW.Strings[AW.Count- 1]);                    //the last
textitem is the name of selected combination
  AW.Free;
end;

function tCombinationList.WriteToString: String;
var i: Integer;
begin
  Result:= MainCombination.WriteToString;
  for i:= 0 to Count- 1 do
  begin
    if tCombination(Items[i])is tMainCombination then Continue;
    Result:= Result+ '/'+ tCombination(Items[i]).WriteToString;
  end;
  Result:= Result+ '/'+ Selected.Caption;                               //the last
textitem is the name of selected combination
end;

//TLayerDialog

procedure TLayersDialog.FormShow(Sender: TObject);
begin
  Refresh;
  if Assigned(MenuItem) then MenuItem.Checked:= True;                   //setting
up the menu
{
  try
    with tRegistry.Create do
    begin
      OpenKey('\Software\'+ Application.Title+ '\LayerDialog', False);  //
HKEY_CURRENT_USER
      Left:= ReadInteger('Left');
      Top:= ReadInteger('Top');
      Width:= ReadInteger('Width');
      Height:= ReadInteger('Height');
      Panel1.Width:= ReadInteger('Splitter');
      Free;
    end;
  except
    ShowMessage('The showing of this will generate a compiler exeption '+#10#13+
                'at the first time in the Delphi IDE.');
  end;
}
end;

procedure TLayersDialog.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if Assigned(MenuItem) then MenuItem.Checked:= False;                  //setting
up the menu
{
  try
    with tRegistry.Create do
    begin
      OpenKey('\Software\'+ Application.Title+ '\LayerDialog', True);   //
HKEY_CURRENT_USER
      WriteInteger('Left', Left);
      WriteInteger('Top', Top);
      WriteInteger('Width', Width);
      WriteInteger('Height', Height);
      WriteInteger('Splitter', Panel1.Width);
```

```
      Free;
    end;
  except
  end;
}
end;

procedure TLayersDialog.Delete1Click(Sender: TObject);
begin
  if ListBox1.Items.Count> 0 then
    CombinationList.Delete(ListBox1.ItemIndex);
  Refresh;
end;

procedure TLayersDialog.Refresh;
begin
  CombinationList.Deploy(ListBox1, ListBox2);
  Drawing.SelectList.Clear;
end;

procedure TLayersDialog.Edit1Click(Sender: TObject);
var New: String;
begin
  New:= InputBox('Name change', 'Please define a new name:', ListBox1.Items[ListBox1.
ItemIndex]);
  CombinationList.ChangeCaption(ListBox1.Items[ListBox1.ItemIndex], New);
  Refresh;
  ListBox1.ItemIndex:= ListBox1.Items.IndexOf(New);
end;

procedure TLayersDialog.New1Click(Sender: TObject);
begin
  CombinationList.Add(tCombination.CreateByName('Untitled'));
  Refresh;
end;

procedure TLayersDialog.ListBox1Click(Sender: TObject);
begin
  CombinationList.Selected:= Drawing.CombinationList.GetCombination(ListBox1.Items[ListBox1.
ItemIndex]);
  Refresh;
  if Drawing.DialogTree.Visible then Drawing.DialogTree.Refresh;
end;

procedure TLayersDialog.Delete2Click(Sender: TObject);
begin
  CombinationList.GetCombination(ListBox1.Items[ListBox1.ItemIndex]).DelLayer(ListBox2.Items
[ListBox2.ItemIndex]);
  ListBox1Click(Self);
end;

procedure TLayersDialog.Edit2Click(Sender: TObject);
var New: String;
begin
  New:= InputBox('Name change', 'Please define a new name:', ListBox2.Items[ListBox2.
ItemIndex]);
  CombinationList.MainCombination.ChangeCaption(ListBox2.Items[ListBox2.ItemIndex], New);
  Refresh;
  ListBox2.ItemIndex:= ListBox2.Items.IndexOf(New);
end;

procedure TLayersDialog.New2Click(Sender: TObject);
```

```
var Lay: tLayer;
begin
  Lay:= CombinationList.MainCombination.AddByLayerName('Untitled');
  if (CombinationList.GetCombination(ListBox1.Items[ListBox1.ItemIndex]) <> CombinationList.
MainCombination) then
     CombinationList.GetCombination(ListBox1.Items[ListBox1.ItemIndex]).AddLayer(Lay);
  Refresh;
end;

procedure TLayersDialog.ListBox1DragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  Accept:= Sender is tListBox;
end;

procedure TLayersDialog.ListBox1DragDrop(Sender, Source: TObject; X,
  Y: Integer);
var Index: Integer;
    APoint: TPoint;
begin
  APoint.X := X;
  APoint.Y := Y;
  Index := ListBox1.ItemAtPos(APoint, True);
  CombinationList.AddLayer(ListBox1.Items[Index], ListBox2.Items[ListBox2.ItemIndex]);
end;

initialization

  CombinationList:= tCombinationList.Create;
  CombinationList.MainCombination:= tMainCombination.CreateByName(DefaultCombination);
  CombinationList.Add(CombinationList.MainCombination);
  CombinationList.fSelected:= CombinationList.MainCombination;                    //irregular
(direct) value setting
  CombinationList.MainCombination.AddLayer(tLayer.CreateByName(DefaultLayer));
  CadProperty.LayerList:= tStringList.Create;
  with CadProperty.LayerList do
  begin
    Sorted:= True;
    Duplicates:= dupIgnore;
  end;
  CadProperty.LayerList.Add(DefaultLayer);

  LayersDialog:= tLayersDialog.Create(Application.MainForm);

end.
```

```
{*****************************************************}
{                                                     }
{              CommonCAD Component Library            }
{              www.sitkei.com/commoncad               }
{              pal@sitkei.com                         }
{                                                     }
{*****************************************************}

unit CadMisc;

interface

uses Classes, QT, Types;

type
  tDrawingFile = class(tStringList)
  protected
  public
    Index: Integer;
    function ReadLn: String;
    procedure Write(aStr: String);
    procedure WriteLn(aStr: String);
  end;

  tAwkLine = class(tStringList)                                    //cutting
off the input string by the defined separator
  protected                                                        //the parts
of the string are collected in a stringlist
  public
    constructor Create(Line, Sep: String);
  end;

//Geometry

  tMatrix = array[1..3, 1..3] of Double;   //transformation matrix

  t3dPoint = record X, Y, Z: Double; end;
  IntsType= (ints_Normal, ints_Paralell, ints_Equal, ints_OnFirstLine,
             ints_OnSecondLine);
function Ang(P1, P2: tPoint): Double;
function Angle(P1, P2: t3dPoint): Double;
function Dist(P1, P2: tPoint): Double;
function Distance3D(const A, B: t3dPoint): Double;
function Half(P1, P2: t3dPoint): t3dPoint;
function Inter(P1, P2, P3, P4: t3dPoint; TestOn: Boolean; var Res: t3dPoint): IntsType;
function NearPoint(P1: tPoint; PP1, PP2: t3dPoint): t3dPoint;      //from P1 a
perp
function Perp(P1, PP1, PP2: tPoint): tPoint;                       //from P1 a
perp for a line
function Pol(P1: tPoint; Ang, Dist: Double): tPoint;
function Polar(P1: t3dPoint; Ang, Dist: Double): t3dPoint;
function RadToAng(Rad: Double): Double;

//Other

function Indent(Num: Integer): String;
function Make3dPoint(aX, aY, aZ: Double): t3dPoint;
function CreateMatrix(Alfa, Beta, Gamma: double): tMatrix;

implementation
```

```
uses QDialogs, SysUtils,
     CadBasic, CadObjects, CadObjects3D, CadDevices, CommonCAD;

//tDrawingFile

function tDrawingFile.ReadLn: String;
begin
  Result:= Strings[Index];
  Inc(Index);
end;

procedure tDrawingFile.Write(aStr: String);
begin
  Strings[Index]:= Strings[Index]+ aStr;
end;

procedure tDrawingFile.WriteLn(aStr: String);
begin
  Strings[Index]:= Strings[Index]+ aStr;
  Add('');
  Inc(Index);
end;

//tAwkLine

constructor tAwkLine.Create(Line, Sep: String);
var Ind: Integer;
procedure Clip(var aStr: String);
begin
  while aStr[1]=' ' do System.Delete(aStr, 1, 1);
  while aStr[Length(aStr)]=' ' do System.Delete(aStr, Length(aStr), 1);
end;
begin
  inherited Create;
  Clip(Line);
  while Pos(Sep, Line)>0 do
  begin
    Ind:= Pos(Sep, Line);
    Add(Copy(Line, 1, Ind- 1));
    System.Delete(Line, 1, Ind);
  end;
  Add(Line);
end;

//Geometry

function Ang(P1, P2: tPoint): Double;
var dY, dX: Double;
begin
  dX:= P2.X-P1.X;
  dY:= P2.Y-P1.Y;
  if (dX=0.0)and(dY=0.0) then
  begin
    Result:=-111;
    Exit;
  end;
  if dX= 0.0 then
  begin
    if dY>0 then Result:= PI/2 else Result:= PI+PI/2;
    Exit;
  end;
```

```
  if dY= 0.0 then
  begin
    if dX<0 then Result:= PI else Result:= 0.0;
    Exit;
  end;
  Result:=Abs(ArcTan(dY/dX));
  if (dx>0)and(dy>0) then Exit;
  if (dx<0)and(dy>0) then Result:= Pi- Result;
  if (dx<0)and(dy<0) then Result:= Pi+ Result;
  if (dx>0)and(dy<0) then Result:= 2*Pi- Result;
end;

function Angle(P1, P2: t3dPoint): Double;
var dY, dX: Double;
begin
  dX:= P2.X- P1.X;
  dY:= P2.Y- P1.Y;
  if (dX=0.0)and(dY=0.0) then
  begin
    Result:=-111;
    Exit;
  end;
  if dX= 0.0 then
  begin
    if dY>0 then Result:= PI/2 else Result:= PI+PI/2;
    Exit;
  end;
  if dY= 0.0 then
  begin
    if dX<0 then Result:= PI else Result:= 0.0;
    Exit;
  end;
  Result:=Abs(ArcTan(dY/dX));
  if (dx>0)and(dy>0) then Exit;
  if (dx<0)and(dy>0) then Result:= Pi- Result;
  if (dx<0)and(dy<0) then Result:= Pi+ Result;
  if (dx>0)and(dy<0) then Result:= 2*Pi- Result;
end;

function Dist(P1, P2: tPoint): Double;
begin
  Result:= 0;
  try
    Result:= Sqrt(Sqr(P1.X-P2.X* 1.0)+ Sqr(P1.Y-P2.Y* 1.0));
  except
    ShowMessage('Invalide floating pont operation');
  end;
end;

function Distance3D(const A, B: t3Dpoint): Double;
var dx, dy, dz: Double;
begin
  dx:= A.X- B.X;
  dy:= A.Y- B.Y;
  dz:= A.Z- B.Z;
  Result:= Sqrt(dx* dx+ dy* dy+ dz* dz);
end;

function Half(P1, P2: t3dPoint): t3dPoint;
begin
  Result.X:= (P1.X+ P2.X) / 2;
  Result.Y:= (P1.Y+ P2.Y) / 2;
```

```
  Result.Z:= (P1.Z+ P2.Z) / 2;
end;

function Inter(P1,P2,P3,P4: t3dPoint; TestOn: Boolean; var Res: t3dPoint): IntsType;
var A1,B1,C1,A2,B2,C2,Deter: Double;
begin
  A1:=P1.Y -P2.Y;
  B1:=P1.X -P2.X;
  C1:=P1.X *P2.Y -P2.X *P1.Y;
  A2:=P3.Y -P4.Y;
  B2:=P3.X -P4.X;
  C2:=P3.X *P4.Y -P4.X *P3.Y;
  Deter:= (A1 *B2 -A2 *B1);
  if Deter =0.0 then
  begin
    Result:=ints_Paralell;
    if (A2=0.0) or (B2=0.0) or (C2=0.0) then Exit;
    if (((A1/A2)=(B1/B2)) and ((A1/A2)=(C1/C2))) then
    begin
      Result:=ints_Equal;
      Exit;
    end;
  end;
  Result:=ints_Normal;
  Res.X:= (B1*C2-B2*C1)/Deter;
  Res.Y:= (C2*A1-C1*A2)/Deter;
  if not TestOn then Exit;
  if ((P1.X<Res.X) and (P2.X>Res.X) and (P1.Y<Res.Y) and (P2.Y>Res.Y)) or
     ((P1.X<Res.X) and (P2.X>Res.X) and (P1.Y>Res.Y) and (P2.Y<Res.Y)) or
     ((P1.X>Res.X) and (P2.X<Res.X) and (P1.Y<Res.Y) and (P2.Y>Res.Y)) or
     ((P1.X>Res.X) and (P2.X<Res.X) and (P1.Y>Res.Y) and (P2.Y<Res.Y)) then
  begin
    Result:=ints_OnFirstLine;
    Exit;
  end;
  if ((P3.X<Res.X) and (P4.X>Res.X) and (P3.Y<Res.Y) and (P4.Y>Res.Y)) or
     ((P3.X<Res.X) and (P4.X>Res.X) and (P3.Y>Res.Y) and (P4.Y<Res.Y)) or
     ((P3.X>Res.X) and (P4.X<Res.X) and (P3.Y<Res.Y) and (P4.Y>Res.Y)) or
     ((P3.X>Res.X) and (P4.X<Res.X) and (P3.Y>Res.Y) and (P4.Y<Res.Y)) then
    Result:=ints_OnSecondLine;
end;

function NearPoint(P1: tPoint; PP1, PP2: t3dPoint): t3dPoint;
var P33: t3dPoint;
    D, Ang: Double;
begin
  P33:= Make3dPoint(P1.X, P1.Y, 0);
  D:= Distance3D(P33, PP1);
  Ang:= Angle(PP1, PP2);
  Result:= Polar(PP1, Ang, D);
end;

function Perp(P1, PP1, PP2: tPoint);  //from P1 a perp line
var AA: Double;
    P1r, P2r, Res: t3dPoint;
begin
  AA:= Ang(PP1, PP2)+ Pi/2;
  P1r:= Make3dPoint(P1.X, P1.Y, 0);
  P2r:= Polar(P1r, AA, 10000);
  Inter(P1r, P2r, Make3dPoint(PP1.X, PP1.Y, 0), Make3dPoint(PP2.X, PP2.Y, 0), True, Res);
  Result.X:= Round(Res.X);
  Result.Y:= Round(Res.Y);
```

```
end;

function Pol(P1: tPoint; Ang, Dist: Double): tPoint;
begin
  Result.X:= Round(P1.X+ Dist*Cos(Ang));
  Result.Y:= Round(P1.Y+ Dist*Sin(Ang));
end;

function Polar(P1: t3dPoint; Ang, Dist: Double): t3dPoint;
begin
  Result.X:= P1.X+Dist*Cos(Ang);
  Result.Y:= P1.Y+Dist*Sin(Ang);
end;

function RadToAng(Rad: Double): Double;
begin
  Result:= Rad* 180/ Pi;
end;

function Indent(Num: Integer): String;
var i: Integer;
begin
  Result:= '';
  for i:= 0 to Num- 1 do
    Result:= Result+ ' ';
end;

function Make3dPoint(aX, aY, aZ: Double): t3dPoint;
begin
  Result.X:= aX;
  Result.Y:= aY;
  Result.Z:= aZ;
end;

function CreateMatrix(Alfa, Beta, Gamma: double): tMatrix;
function MatrixMultiple(A,B: tMatrix): tMatrix;
var sor, oszlop, i: integer;
begin
  for sor:=1 to 3 do
    for oszlop:=1 to 3 do
    begin
      Result[sor,oszlop]:=0;
      for i:=1 to 3 do
        Result[sor, oszlop]:=Result[sor, oszlop] + A[sor, i] * B[i, oszlop];
    end;
end;
var X,Y,Z: tMatrix;
begin
  X[1,1]:=1.0;          X[1,2]:=0.0;          X[1,3]:=0.0;
  X[2,1]:=0.0;          X[2,2]:=cos(Alfa);    X[2,3]:=-sin(Alfa);
  X[3,1]:=0.0;          X[3,2]:=sin(Alfa);    X[3,3]:=cos(Alfa);
  Y[1,1]:=cos(Beta);    Y[1,2]:=0.0;          Y[1,3]:=-sin(Beta);
  Y[2,1]:=0.0;          Y[2,2]:=1.0;          Y[2,3]:=0.0;
  Y[3,1]:=sin(Beta);    Y[3,2]:=0.0;          Y[3,3]:=cos(Beta);
  Z[1,1]:=cos(Gamma);   Z[1,2]:=-sin(Gamma);  Z[1,3]:=0.0;
  Z[2,1]:=sin(Gamma);   Z[2,2]:=cos(Gamma);   Z[2,3]:=0.0;
  Z[3,1]:=0.0;          Z[3,2]:=0.0;          Z[3,3]:=1.0;
  Result:= X;
  Result:= MatrixMultiple(Result, Y);
  Result:= MatrixMultiple(Result, Z);
end;
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadObjects;

interface

uses Classes, QGraphics, QT, Types,
     CadBasic, CadMisc, CommonCAD;

type
  tEllipse = class(tDrawComponent)
  private
  public
    procedure Draw(aCanvas :tCanvas); override;
    function GetRgn: QRegionH; override;
    procedure SetPropList; override;
  end;

  tPicture = class(tDrawComponent)
  private
  public
    FileName: String;
    procedure Draw(aCanvas :tCanvas); override;
    procedure SetPropList; override;
  end;

  tPolyline = class(tDrawComponent)
  private
  public
    constructor CreateByPoint(aOwner: tComponent; aPoint: t3DPoint); override;
    procedure Draw(aCanvas :tCanvas); override;
    procedure SetPropList; override;
    procedure SizeTo(aPoint: tPoint); override;
  end;

  tRectangle = class(tDrawComponent)
  private
  public
    procedure Draw(aCanvas :tCanvas); override;
    procedure SetPropList; override;
  end;

  tText = class(tDrawComponent)
  private
  public
    Text, FontName: String;
    FontSize, FontStyle: Integer;
    constructor CreateByPoint(aOwner: tComponent; aPoint: t3dPoint); override;
    procedure Draw(aCanvas :tCanvas); override;
    procedure ScaleByRubberLine; override;
    procedure SetPropList; override;
  end;
```

```
implementation

uses QDialogs,
     CadGrip, CadProperty;

//tEllipse

procedure tEllipse.Draw(aCanvas :tCanvas);
var L, R: tPoint;
begin
  L:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).GetPoint);
  R:= Drawing.Adjust(tDrawPoint(PointList.Items[2]).GetPoint);
  SetupCanvas(aCanvas);
  aCanvas.Ellipse(L.X, L.Y, R.X, R.Y);
end;

function tEllipse.GetRgn: QRegionH;
var L, R: tPoint;
    R1, R2: QRegionH;
begin
  L:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).GetPoint);
  R:= Drawing.Adjust(tDrawPoint(PointList.Items[2]).GetPoint);

  R1:= QRegion_create(L.X, L.Y, R.X - L.X, R.Y - L.Y, QRegionRegionType_Ellipse);
  if Not Filled then
  begin
     R2:= QRegion_create(L.X + 2, L.Y + 2, R.X - L.X - 4, R.Y - L.Y - 4,
QRegionRegionType_Ellipse);
     QRegion_subtract(R1, R1, R2);
     QRegion_destroy(R2);
  end;
  Result:= R1;
  //R1.Free;
end;

procedure tEllipse.SetPropList;
begin
  SetCaption('Ellipse#');
  Color:= clLime;
  LineColor:= clBlack;
  Closed:= True;
  inherited;
  PropList.Kill('Closed');
  PointList.PointType:= ptRect;
end;

//tPicture

procedure tPicture.Draw(aCanvas :tCanvas);
var //myGraphics : TGPGraphics;
//    myImage: tGPImage;
    PP:tPoint;
//    destinationPoints : array[0..2] of TGPPoint;
begin
  if FileName= '' then
  begin
    aCanvas.Brush.Style:= bsDiagCross;
    aCanvas.Brush.Color:= clGreen;
    aCanvas.Pen.Color:= clGreen;
    aCanvas.Pen.Width:= 1;
    PointList.DrawPolygon(aCanvas);
```

```
    end else
    begin
      PP:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).Value);
//    destinationPoints[0].X:= PP.X;
//    destinationPoints[0].Y:= PP.Y;
      PP:= Drawing.Adjust(tDrawPoint(PointList.Items[1]).Value);
//    destinationPoints[1].X:= PP.X;
//    destinationPoints[1].Y:= PP.Y;
      PP:= Drawing.Adjust(tDrawPoint(PointList.Items[3]).Value);
//    destinationPoints[2].X:= PP.X;
//    destinationPoints[2].Y:= PP.Y;
      {
      myGraphics := TGPGraphics.Create(aCanvas.Handle);
      myImage:= tGPImage.Create(FileName, True);
      myGraphics.DrawImage(myImage, PGPPoint(@destinationPoints), 3);
      myImage.Free;
      myGraphics.Free;
      }
    end;
  end;

  procedure tPicture.SetPropList;
  begin
    SetCaption('Picture#');
    Closed:= True;
    Filled:= True;
    inherited;
    with PropList do
    begin
      Kill('Color');
      Kill('Filled');
      Kill('Line.color');
      Kill('Line.width');
      Kill('Invisible');
      Add(tPictFileNameProp.CreateProp(@FileName, 'Filename'));
    end;
    PointList.PointType:= ptRect;
  end;

  //Polyline

  constructor tPolyline.CreateByPoint(aOwner: tComponent; aPoint: t3DPoint);
  begin
    CreateBy(aOwner);
    PointList.Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X, aPoint.Y, 0), Self));
    PointList.Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X, aPoint.Y, 0), Self));
  end;

  procedure tPolyline.Draw(aCanvas :tCanvas);
  begin
    SetupCanvas(aCanvas);
    if Closed then PointList.DrawPolygon(aCanvas)
              else PointList.DrawPolyline(aCanvas);
  end;

  procedure tPolyline.SetPropList;
  begin
    SetCaption('Nodo#');
    Color:= clPurple;
    inherited;
  end;
```

```
procedure tPolyline.SizeTo(aPoint: tPoint);
begin
  tDrawPoint(PointList.Items[PointList.Count- 1]).Value.X:= aPoint.X;
  tDrawPoint(PointList.Items[PointList.Count- 1]).Value.Y:= aPoint.Y;
  PointList.Update;
end;

//tRectangle

procedure tRectangle.Draw(aCanvas: tCanvas);
begin
  SetupCanvas(aCanvas);
  PointList.DrawPolygon(aCanvas);
end;

procedure tRectangle.SetPropList;
begin
  SetCaption('Rectangle#');
  Color:= clNavy;
  LineColor:= clBlack;
  Closed:= True;
  inherited;
  PropList.Kill('Closed');
  PointList.PointType:= ptRect;
end;

//tText

constructor tText.CreateByPoint(aOwner: tComponent; aPoint: t3dPoint);
begin
  inherited CreateByPoint(aOwner, aPoint);
  tDrawPoint(PointList.Items[1]).Value:= Make3dPoint(aPoint.X+ 1000, aPoint.Y, 0);
  Drawing.FinishCommand;
end;

procedure tText.Draw(aCanvas :tCanvas);
{
var Matrix: TGPMatrix;
    FontFamily: TGPFontFamily;
    Font: TGPFont;
    SolidBrush: TGPSolidBrush;
    StringFormat: TGPStringFormat;
    myGraphic: TGPGraphics;
    P0, P1: tPoint;
    NewRect: tGPRectF;
    RR, WW, HH: Double;
}
begin
{
  SetupCanvas;
  P0:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).Value);
  P1:= Drawing.Adjust(tDrawPoint(PointList.Items[1]).Value);
  RR:= Ang(P0, P1);
  Matrix:= TGPMatrix.Create;
  SolidBrush:= TGPSolidBrush.Create(MakeColor(255, GetBlue(Color), GetGreen(Color), GetRed
(Color)));
  FontFamily:= TGPFontFamily.Create(FontName);
  Font := TGPFont.Create(FontFamily, Round(FontSize* Drawing.Zoom.Factor),
FontStyleRegular, UnitPixel);
  Matrix.RotateAt(RadToAng(RR), MakePoint(P0.x*1.0, P0.y*1.0), MatrixOrderAppend);
```

```
    myGraphic:= TGPGraphics.Create(aCanvas.Handle);
    myGraphic.SetTransform(matrix);
    myGraphic.DrawString(Text, -1, Font, MakePoint(P0.X*1.0, P0.Y*1.0), SolidBrush);
    StringFormat:= TGPStringFormat.Create;
    StringFormat.SetFormatFlags(StringFormatFlagsNoWrap);
    myGraphic.MeasureString(Text, Length(Text), Font, MakePoint(P0.X* 1.0, P0.Y* 1.0),
  StringFormat, NewRect);
    StringFormat.Free;
    Matrix.Free;
    FontFamily.Free;
    Font.Free;
    SolidBrush.Free;
    myGraphic.Free;
    WW:= NewRect.Width/Drawing.Zoom.Factor/ 1.03;
    HH:= NewRect.Height/Drawing.Zoom.Factor/ 1.16;
    P1:= Pol(tDrawPoint(PointList.Items[0]).Get2dPoint, RR, WW);                    //setting
  up the pointlist's points for the region handling:
    tDrawPoint(PointList.Items[1]).Value:= Make3dPoint(P1.X, P1.Y, 0);
    P1:= Pol(tDrawPoint(PointList.Items[1]).Get2dPoint, RR+ Pi/2, HH);
    tDrawPoint(PointList.Items[2]).Value:= Make3dPoint(P1.X, P1.Y, 0);
    P1:= Pol(tDrawPoint(PointList.Items[0]).Get2dPoint, RR+ Pi/2, HH);
    tDrawPoint(PointList.Items[3]).Value:= Make3dPoint(P1.X, P1.Y, 0);
    PointList.Update;
  }
  end;

  procedure tText.ScaleByRubberLine;
  var D1, D2, SF: Double;
      i: Integer;
      FP, LP, SP, PP1, PP2, RP: tPoint;
  begin
    FP:= Point(Round(Drawing.RubberLine.First.X), Round(Drawing.RubberLine.First.Y));
    LP:= Point(Round(Drawing.RubberLine.Last.X), Round(Drawing.RubberLine.Last.Y));
    SP:= Point(Round(Drawing.RubberLine.Second.X), Round(Drawing.RubberLine.Second.Y));
    RP:= Drawing.Inverse(FP);
    D1:= Dist(FP, LP);
    D2:= Dist(FP, SP);
    if D1= 0 then Exit;
    SF:= D2/ D1;
    if SF< 0.01 then Exit;
    FontSize:= Round(FontSize* SF);
    for i:= 0 to 1 do
    begin
      PP1:= tDrawPoint(PointList.Items[i]).Get2dPoint;
      PP2:= Pol(RP, Ang(RP, PP1), Dist(RP, PP1)* SF);
      tDrawPoint(PointList.Items[i]).Value.X:= PP2.X;
      tDrawPoint(PointList.Items[i]).Value.Y:= PP2.Y;
    end;
  end;

  procedure tText.SetPropList;
  begin
    inherited;
    SetCaption('Text#');
    Text:= 'This is a text';
    FontName:= 'Arial';
    FontSize:= 1500;
    Closed:= True;
    Filled:= True;
    Color:= clNavy;
    with PropList do
```

```
  begin
    Kill('Color');
    Kill('Line.Color');
    Kill('Line.Width');
    Kill('Width');
    Kill('Height');
    Kill('Filled');
    Kill('Invisible');
    Add(tColorProp.CreateProp(@Color, 'Font.Color'));
    Add(tStringProp.CreateProp(@FontName, 'Font.Name'));
    Add(tIntegerProp.CreateProp(@FontSize, 'Font.Size'));
    Add(tIntegerProp.CreateProp(@FontStyle, 'Font.Style'));
    Add(tStringProp.CreateProp(@Text, 'Text'));
  end;
end;

initialization

  if (not Assigned(DrawCompItemList)) then
    DrawCompItemList:= tDrawCompItemList.Create;                    //
DrawCompItemList have to be assigned
  DrawCompItemList.Add(tDrawCompItem.Create('Ellipse', tEllipse));  //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Picture', tPicture));
  DrawCompItemList.Add(tDrawCompItem.Create('Polyline', tPolyline));
  DrawCompItemList.Add(tDrawCompItem.Create('Rectangle', tRectangle));
  DrawCompItemList.Add(tDrawCompItem.Create('Text', tText));
end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadObjects3D;

interface

uses Classes, QGraphics, QT,
     CadBasic, CadMisc, CommonCAD;

type
  tFace = class(tDrawComponent)
  private
  public
    procedure Draw(aCanvas :tCanvas); override;
    procedure SetPropList; override;
  end;

  tCube = class(tDrawComponent)
  private
    procedure MakeFaces;
  public
    constructor CreateByPoint(aOwner: tComponent; aPoint: t3dPoint); override;
    procedure Draw(aCanvas :tCanvas); override;
    function GetRgn: QRegionH; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropList; override;
    procedure WriteXml(aIndent: Integer); override;
  end;

implementation

uses QDialogs, SysUtils,
     CadGrip, CadProperty;

//tFace

procedure tFace.Draw(aCanvas :tCanvas);
begin
  SetupCanvas(aCanvas);
{//}  aCanvas.Brush.Style:= bsClear;
  PointList.DrawPolygon(aCanvas);
end;

procedure tFace.SetPropList;
begin
  SetCaption('Face#');
  Color:= clTeal;
  Closed:= True;
  Filled:= True;
  inherited;
  with PropList do
  begin
    Kill('Invisible');
    Kill('Line.Color');
    Kill('Line.Width');
```

```
  end;
end;

//tCube

procedure tCube.MakeFaces;
var Face: tFace;
    i: Integer;
begin
  Face:= TFace.CreateBy(Self);                                          //top face,
adding references to the face's pointlists
    for i:= 0 to 3 do Face.PointList.Add(PointList.Items[i]);
  Face:= TFace.CreateBy(Self);                                          //bottom
face
    for i:= 4 to 7 do Face.PointList.Add(PointList.Items[i]);
  Face:= TFace.CreateBy(Self);                                          //+x face
    Face.PointList.Add(PointList.Items[1]);
    Face.PointList.Add(PointList.Items[6]);
    Face.PointList.Add(PointList.Items[5]);
    Face.PointList.Add(PointList.Items[2]);
  Face:= TFace.CreateBy(Self);                                          //-x face
    Face.PointList.Add(PointList.Items[3]);
    Face.PointList.Add(PointList.Items[4]);
    Face.PointList.Add(PointList.Items[7]);
    Face.PointList.Add(PointList.Items[0]);
  Face:= TFace.CreateBy(Self);                                          //+y face
    Face.PointList.Add(PointList.Items[7]);
    Face.PointList.Add(PointList.Items[6]);
    Face.PointList.Add(PointList.Items[1]);
    Face.PointList.Add(PointList.Items[0]);
  Face:= TFace.CreateBy(Self);                                          //-y face
    Face.PointList.Add(PointList.Items[3]);
    Face.PointList.Add(PointList.Items[2]);
    Face.PointList.Add(PointList.Items[5]);
    Face.PointList.Add(PointList.Items[4]);
end;

constructor tCube.CreateByPoint(aOwner: tComponent; aPoint: t3dPoint);
begin
  CreateBy(aOwner);
  with PointList do
  begin                                                                 //adding
points to the cube's pointlist
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X -6000, aPoint.Y +6000, +6000),
Self));   //0
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X +6000, aPoint.Y +6000, +6000),
Self));   //1
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X +6000, aPoint.Y -6000, +6000),
Self));   //2
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X -6000, aPoint.Y -6000, +6000),
Self));   //3
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X -6000, aPoint.Y -6000, -6000),
Self));   //4
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X +6000, aPoint.Y -6000, -6000),
Self));   //5
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X +6000, aPoint.Y +6000, -6000),
Self));   //6
    Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X -6000, aPoint.Y +6000, -6000),
Self));   //7
  end;
```

```
  MakeFaces;
  Drawing.FinishCommand;
end;

procedure tCube.Draw(aCanvas :tCanvas);
var i: Integer;
begin
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).Paint(aCanvas);
end;

function tCube.GetRgn: QRegionH;
begin
  Result:= nil;
end;

procedure tCube.ReadXml(aString: String);
var CloseString, S: String;
    AW: tAwkLine;
begin
  PropList.ReadXml(aString);
  MakeFaces;
  if (LinkID> -1) then
    Drawing.LinkList.AddIndex(LinkId, Self);                    //register
the linksource
  if (Pos('/>', aString)<> 0) then Exit;                        //hasn't
children
  CloseString:= GetCloseString(aString);
  repeat
    S:= Drawing.DrawingFile.ReadLn;
    if S='' then Break;
    Clip(S);
    AW:= tAwkLine.Create(S, ' ');
    if (Pos('<', AW.Strings[0])<> 0)and
       (Pos('/', AW.Strings[0])= 0) then
    try
      with DrawCompItemList.GetClassType(AW.Strings[0]).CreateBy(Self) do      //creating
a child object
      begin
        S:= Drawing.DrawingFile.ReadLn;
        PointList.SetPointType;
      end;
    except
      ShowMessage('The '+ AW.Strings[0]+ ' didn''t registered for the system');
    end;
    AW.Free;
  until(Pos(CloseString, S)<>0);
end;

procedure tCube.SetPropList;
begin
  SetCaption('Cube#');
  inherited;
  with PropList do
  begin
    Kill('Closed');
    Kill('Color');
    Kill('Filled');
    Kill('Invisible');
    Kill('Line.Color');
    Kill('Line.Width');
```

```
  end;
end;

procedure tCube.WriteXML(aIndent: Integer);
var i: Integer;
begin
  Drawing.DrawingFile.Write(Indent(aIndent)+ '<'+ Copy(LowerCase(ClassName), 2, Length
(ClassName)-1));
  if ComponentCount = 0 then
  begin
    PropList.WriteXML(True);
    Exit;
  end;
  PropList.WriteXML(False);
  for i:= 7 to ComponentCount- 1 do                                    //first 6
ones are the faces
    tBasic(Components[i]).WriteXML(aIndent+ 2);
  Drawing.DrawingFile.Writeln(Indent(aIndent)+ '</'+ Copy(LowerCase(ClassName), 2, Length
(ClassName)-1)+ '>');
end;

initialization

  if (not Assigned(DrawCompItemList)) then
    DrawCompItemList:= tDrawCompItemList.Create;                       //
DrawCompItemList have to be assigned
  DrawCompItemList.Add(tDrawCompItem.Create('Face', tFace));           //
registering of drawing classes
  DrawCompItemList.Add(tDrawCompItem.Create('Cube', tCube));

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadObjLink;

interface

uses Classes, QT, Types, QGraphics,
     CadBasic, CadMisc;

type
  tLinkList = class(tList)                                          //link
handling in the Draw component
  private                                                           //
collection of linked objects
  public                                                            //the Link.
SourceID = ItemIndex
    procedure AddIndex(aIndex: Integer; aPointer: Pointer);
    procedure DeleteLinks(anID: Integer);
    function GetID: Integer;
  end;

  tLink = class(tDrawComponent)
  private
    procedure OnShowSource(Sender: TObject);
  public
    constructor CreateByPoint(aOwner: tComponent; aPoint: t3dPoint); override;
    procedure Draw(aCanvas :tCanvas); override;
    function GetRgn: QRegionH; override;
    procedure Mirror(P1, P2: tPoint); override;
    function OnConnect(aPoint: tPoint): tDrawComponent; override;
    procedure PutGrips; override;
    procedure RotateByRubberLine; override;
    procedure ScaleByRubberLine; override;
    procedure SetPropList; override;
  end;

implementation

uses QDialogs, QForms,
     CommonCAD, CadOsnap, CadProperty;

//tLinkList

procedure tLinkList.AddIndex(aIndex: Integer; aPointer: Pointer);
var i: Integer;
begin
  if tDrawComponent(aPointer) is tLink then Exit;
  if aIndex < Count
    then Items[aIndex]:= aPointer
    else for i:= Count to aIndex do Add(aPointer);
end;

procedure tLinkList.DeleteLinks(anID: Integer);
var DelList: tList;
```

```
    i: Integer;
  procedure LookIt(aComp: tBasic);
  var i: Integer;
    DC: tBasic;
  begin
    for i:= 0 to aComp.ComponentCount- 1 do
    begin
      DC:= tDrawComponent(aComp.Components[i]);
      if (DC is tLink) and (DC.LinkID= anID)
        then DelList.Add(aComp.Components[i])
        else LookIt(tDrawComponent(aComp.Components[i]));
    end;
  end;
  begin
    DelList:= tList.Create;
    LookIt(Drawing);
    for i:= 0 to DelList.Count- 1 do
      Drawing.DeleteObject(DelList.Items[i]);
    DelList.Free;
  end;

  function tLinkList.GetID: Integer;
  var PP: tComponent;
  begin
    if Drawing.SelectList.Count< 1 then
    begin
      Result:= -1;
      Exit;
    end;
    PP:= Drawing.SelectList.GetSelection;
    if PP is tLink then
    begin
      Result:= tLink(PP).LinkID;
    end else
    begin
      Result:= IndexOf(PP);
      if Result= -1 then
      begin
        Result:= Count;
        Add(PP);
      end;
    end;
    tDrawComponent(Items[Result]).LinkID:= Result;
  end;

  //tLink

  procedure tLink.OnShowSource(Sender: TObject);
  begin
    Drawing.SelectList.AddBy(tComponent(Drawing.LinkList.Items[LinkID]), False);
    Drawing.Invalidate;
  end;

  constructor tLink.CreateByPoint(aOwner: tComponent; aPoint: t3dPoint);
  begin
    if Drawing.SelectList.Count<> 1 then Exit;
    CreateBy(aOwner);
    PointList.Add(tDrawPoint.CreateBy(Make3dPoint(aPoint.X, aPoint.Y, 0), Self));
    Drawing.FinishCommand;
  end;
```

```
procedure tLink.Draw(aCanvas :tCanvas);
var DC: tDrawComponent;
    PP1, PP2: t3dPoint;
begin
  try
    DC:= tDrawComponent(Drawing.LinkList.Items[LinkID]);
    PP1:= GetLinkPoint;
    PP2:= DC.GetLinkPoint;
    DC.PaintTo(Round(PP1.X- PP2.X),
               Round(PP1.Y- PP2.Y),
               Round(PP1.Z- PP2.Z), aCanvas);
  except
    ShowMessage('Problem with Link.Paint.');
    Drawing.DeleteObject(Self);
  end;
end;

function tLink.GetRgn: QRegionH;
var IP, P1: tPoint;
    HH: QRegionH;
    Height, Width: Integer;
begin
  P1:= Drawing.Adjust(tDrawComponent(Drawing.LinkList.Items[LinkID]).GetLinkPoint);
  IP:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).Value);
  IP:= Point(IP.X- P1.X, IP.Y- P1.Y);
  HH:= tDrawComponent(Drawing.LinkList.Items[LinkID]).GetRgn;
  Height:= 50; Width:= 50; //Borrar
{
  GPRE:= tGPRegion.Create(HH);
  GPRE.Translate(IP.X, IP.Y);
  GG:= tGPGraphics.Create(aCanvas.Handle);
  Result:= GPRE.GetHRGN(GG);
  GG.Free;
  GPRE.Free;
}
  Result:=QRegion_Create(IP.X + P1.X, IP.Y + P1.Y, Width, Height,
QRegionRegionType_Rectangle);
end;

procedure tLink.Mirror(P1, P2: tPoint);
begin
end;

function tLink.OnConnect(aPoint: tPoint): tDrawComponent;
var DC: tDrawComponent;
    PP1, PP2, P3, P4, PP, Delta: t3dPoint;
    TP1, TP2, TDelta, aP, Ref: tPoint;
    HD: Double;
begin
  DC:= tDrawComponent(Drawing.LinkList.Items[LinkID]);
  PP1:= GetLinkPoint;
    TP1:= Drawing.Adjust(PP1);
  PP2:= DC.GetLinkPoint;
    TP2:= Drawing.Adjust(PP2);
  Delta:= Make3dPoint(PP1.X- PP2.X, PP1.Y- PP2.Y, 0);
    TDelta:= Point(TP1.X- TP2.X, TP1.Y- TP2.Y);
  Ref:= Point(Round(aPoint.X- TDelta.X), Round(aPoint.Y- TDelta.Y));
  if DC.PointList.OnConnect(Ref)
    then Result:= Self
    else Result:= nil;
  if Assigned(Result) then
```

```
    begin
      aP:= Drawing.Inverse(Ref);
      P3:= tDrawPoint(DC.PointList.Items[Drawing.Mouse.ConPoints.X]).Value;
      P4:= tDrawPoint(DC.PointList.Items[Drawing.Mouse.ConPoints.Y]).Value;
      if (P3.X= P4.X)and (P3.Y= P4.Y) then
         Exit;
      HD:= OsnapSize/ 2/ Drawing.Zoom.Factor;
      if Dist(Point(Round(P3.X), Round(P3.Y)), aP) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= Make3dPoint(P3.X+ Delta.X, P3.Y+ Delta.Y, 0);
        Drawing.OsnapShape.OsnapType:= ctEnd;
        Exit;
      end;
      if Dist(aP, Point(Round(P4.X), Round(P4.Y))) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= Make3dPoint(P4.X+ Delta.X, P4.Y+ Delta.Y, 0);
        Drawing.OsnapShape.OsnapType:= ctEnd;
        Exit;
      end;
      PP:= Half(P3, P4);
      if Dist(Point(Round(PP.X), Round(PP.Y)), aP) < HD then
      begin
        Drawing.OsnapShape.OsnapPoint:= Make3dPoint(PP.X+ Delta.X, PP.Y+ Delta.Y, 0);
        Drawing.OsnapShape.OsnapType:= ctMid;
        Exit;
      end;
      PP:= NearPoint(aP, P3, P4);
      Drawing.OsnapShape.OsnapPoint:= Make3dPoint(PP.X+ Delta.X, PP.Y+ Delta.Y, 0);
      Drawing.OsnapShape.OsnapType:= ctNear;
    end;
  end;

  procedure tLink.PutGrips;
  begin
    tDrawPoint(PointList.Items[0]).PutGrips;
  end;

  procedure tLink.RotateByRubberLine;
  begin
  end;

  procedure tLink.ScaleByRubberLine;
  begin
  end;

  procedure tLink.SetPropList;
  begin
    SetCaption('Link#');
    LinkID:= Drawing.LinkList.GetID;
    inherited;
    with PropList do
    begin
      Kill('Mirror');
      Kill('Rotate');
      Kill('Scale');
      Kill('Closed');
      Kill('Color');
      Kill('Filled');
      Kill('Line.Width');
      Kill('Line.Color');
      Add(tMenuProp.CreateProp(OnShowSource, 'ShowSource'));
```

```
  end;
end;

initialization

  if (not Assigned(DrawCompItemList)) then
    DrawCompItemList:= tDrawCompItemList.Create;                    //
DrawCompItemList have to be assigned
  DrawCompItemList.Add(tDrawCompItem.Create('Link', tLink));        //
registering of drawing class

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadOsnap;

interface

uses Classes, Types, QGraphics, QT,
     CadBasic, CadMisc;

type
  tOsnapType = (ctMid, ctNear, ctPerp, ctEnd);

  tOsnapShape = class
  private
  public
    OsnapPoint: t3dPoint;
    OsnapType: tOsnapType;
    Visible: Boolean;
    procedure Paint;
  end;

const
  OsnapColor= clRed;
  OsnapSize= 12;

implementation

uses CommonCAD;

procedure tOsnapShape.Paint;
var PP: tPoint;
begin
  if not Visible then
  begin
    Drawing.StatusText:= '';
    Exit;
  end;
  PP:= Drawing.Adjust(OsnapPoint);
  with Drawing.Canvas do
  begin
    Brush.Style:= bsClear;
    Pen.Width:= 1;
    Pen.Color:= OsnapColor;
    case OsnapType of
      ctEnd:
        begin
          Rectangle(PP.X- OsnapSize div 2, PP.Y- OsnapSize div 2,
                    PP.X+ OsnapSize div 2, PP.Y+ OsnapSize div 2);
          Drawing.StatusText:= 'End';
        end;
      ctMid:
        begin
```

```
          Polygon([Point(PP.X, PP.Y- OsnapSize div 2),
                   Point(PP.X+ OsnapSize div 2, PP.Y+ OsnapSize div 2),
                   Point(PP.X- OsnapSize div 2, PP.Y+ OsnapSize div 2)]);
          Drawing.StatusText:= 'Mid';
        end;
      else
        begin                                                          //ctNear
          Polyline([Point(PP.X- OsnapSize div 2, PP.Y- OsnapSize div 2),
                    Point(PP.X+ OsnapSize div 2, PP.Y+ OsnapSize div 2)]);
          Polyline([Point(PP.X+ OsnapSize div 2, PP.Y- OsnapSize div 2),
                    Point(PP.X- OsnapSize div 2, PP.Y+ OsnapSize div 2)]);
          Drawing.StatusText:= 'Nea';
        end;
    end;
  end;
end;

end.
```

```
{*****************************************************}
{                                                     }
{              CommonCAD Component Library            }
{              www.sitkei.com/commoncad               }
{              pal@sitkei.com                         }
{                                                     }
{*****************************************************}

unit CadProperty;

interface

uses QButtons, Classes, QControls, QExtCtrls, QDialogs, SysUtils,
     QForms, QGraphics, QGrids, QStdCtrls, QT, Types;

type
  tCustomProp = class;

  tPropList = class(tList)                                          //list of
props
    private
      fOwner: tComponent;
      procedure GetAttributes(aString: String; var aList: tList);
    public
      Default: tCustomProp;
      procedure Assign(Source: TPersistent);
      constructor Create;
      constructor CreateBy(aOwner: tComponent);
      procedure CopyTo(aPropList: tPropList);
      function Find(aName: String): tObject;
      function GetText: String;
      procedure Kill(aName: String);
      procedure PushAll(aPropList: tPropList);
      procedure ReadXml(aString: String);
      procedure SetPropEditors;
      procedure SetText(aString: String);
      procedure WriteXml(CloseTag: Boolean);
    end;

  tCustomProp = class                                               //the prop
is like a property
      Name: String;
      Value: Pointer;
    public
      InternalUse: Boolean;                                         //to hide
for user
      constructor CreateProp(aValue: Pointer; aName: String); overload; virtual;
      procedure CopyTo(aProp: tCustomProp); virtual; abstract;
      function GetValue: String; virtual; abstract;
      procedure ReadXml(aString: String); virtual; abstract;
      procedure SetPropEditor; virtual; abstract;
      procedure SetValue(aValue: String); virtual; abstract;
      procedure WriteXml; virtual; abstract;
    end;

  tStringProp = class(tCustomProp)
    public
      procedure CopyTo(aProp: tCustomProp); override;
      function GetValue: String; override;
      procedure ReadXml(aString: String); override;
```

```
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tNameProp = class(tStringProp)
  public
    procedure SetPropEditor; override;
  end;

  tPictFileNameProp = class(tStringProp)
  public
    procedure SetPropEditor; override;
  end;

  tIntegerProp = class(tCustomProp)
  public
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tColorProp = class(tIntegerProp)
  public
    procedure SetPropEditor; override;
  end;

 tFloatProp = class(tCustomProp)
  public
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

 tPointsProp = class(tCustomProp)
  public
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tListProp = class(tCustomProp)
    List: tStringList;
  public
    constructor CreateListProp(aValue: Pointer; aName: String; aList: tStringList);
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;
```

```
  tBooleanProp = class(tListProp)
  public
    constructor CreateProp(aValue: Pointer; aName: String); override;
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tLayerProp = class(tListProp)
  public
    constructor CreateProp(aValue: Pointer; aName: String); override;
  end;

  tCombinationListProp = class(tCustomProp)
  public
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tMenuProp = class(tCustomProp)
  private
    Notify: tNotifyEvent;
  public
    constructor CreateProp(aNotify: tNotifyEvent; aName: String); reintroduce;
    procedure CopyTo(aProp: tCustomProp); override;
    function GetValue: String; override;
    procedure ReadXml(aString: String); override;
    procedure SetPropEditor; override;
    procedure SetValue(aValue: String); override;
    procedure WriteXML; override;
  end;

  tGripMenuProp = class(tMenuProp)
  private
  public
    procedure SetPropEditor; override;
  end;

//PropEditors

//  const wmSpeedBtnClicked = WM_USER+ 100;

type
  tPropEditorList = class(tList)                                    //list of
propeditors
  public
    Default: tComponent;
    procedure Sort;
  end;

  tPropListBox = class(tListBox)                                    //listbox
for propeditors
    property OnClick;
```

```
      property OnExit;
    private
    public
      constructor Create(aOwner: tComponent); override;
      procedure ClickEvent(Sender: tObject);
      procedure ExitEvent(Sender: tObject);
      function GetIndex(aString: String): Integer;
      procedure Setting(List: tStringList);
    end;

    tCustomButton = class(tSpeedButton)                                    //button on
  a propeditor
      property OnClick;
    private
    public
      constructor Create(aOwner: tComponent); override;
      procedure ClickEvent(Sender: tObject);
    end;

    tPointsButton = class(tCustomButton)
      constructor Create(aOwner: tComponent); override;
    end;

    tArrowButton = class(tCustomButton)
      constructor Create(aOwner: tComponent); override;
    end;

    tPropEdit = class(tEdit)                                               //editor
  for a propeditor
    private
      property OnDblClick;
      property OnKeyPress;
      property OnChange;
    public
      Changed: Boolean;
      constructor Create(AOwner: TComponent); override;
      procedure ChangeEvent(Sender: tObject);
      procedure DblClickEvent(Sender: tObject);
      procedure KeyPressEvent(Sender:TObject; var Key: Char);
    end;

    tCustomPropEditor = class(tCustomControl)                              //basic
  propeditor
    private
      property OnClick;
      property OnEnter;
      property OnExit;
      property OnMouseMove;
    protected
      Value: Pointer;
      procedure Paint; override;
    public
      Dirty, Selected: Boolean;
      Button: tCustomButton;
      Editor: tPropEdit;
      ListBox: tPropListBox;
      Caption: String;
      constructor CreateByName(AOwner: TComponent; aName: String);
      destructor Destroy; override;
      procedure AddOwner(Sender: tObject);      {for PeAlias only}
      procedure ChangeEditorEvent(Sender: tObject);
```

```
      procedure ClickEvent(Sender: tObject); virtual;
      procedure DblClickEvent(Sender: tObject); virtual;
      procedure EnterEvent(Sender: tObject); virtual;
      procedure ExitEvent(Sender: tObject); virtual;
      procedure MouseMoveEvent(Sender :tObject; Shift: tShiftState; X,
    Y: Integer);
      function GetIdName: string;
      function GetValue: Pointer;
      procedure SendResult;
      procedure SetBounds(aLeft, aTop, aWidth, aHeight: Integer); override;
      procedure SetEditorValue(S: String);
    published
      procedure LostSelected; virtual;
    end;

    tPointsPropEditor = class(tCustomPropEditor)
    protected
      Value: Pointer;
    public
      procedure ButtonClicked(Sender: tObject);
      constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
      procedure DblClickEvent(Sender: tObject); override;
      procedure EnterEvent(Sender: tObject); override;
      procedure ExitEvent(Sender: tObject); override;
      procedure SetBounds(aLeft, aTop, aWidth, aHeight: Integer); override;
    end;

    tNamePropEditor = class(tCustomPropEditor)
    protected
      Value: pString;
      SavedValue: String;
    public
      constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
      procedure ExitEvent(Sender: tObject); override;
      function GetValue: pString;
    end;

    pDouble = ^Double;

    tFloatPropEditor = class(tCustomPropEditor)
    protected
      Value: pDouble;
      SavedValue: Double;
    public
      constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
      procedure ExitEvent(Sender: tObject); override;
    end;

    tIntegerPropEditor = class(tCustomPropEditor)
    protected
      Value: pInteger;
      SavedValue: Integer;
    public
      constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
      procedure ExitEvent(Sender: tObject); override;
    end;

    tColorPropEditor = class(tIntegerPropEditor)
    protected
    public
      constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
```

```
    procedure ButtonClicked(Sender: tObject);
    procedure EnterEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
    procedure SetBounds(aLeft, aTop, aWidth, aHeight: Integer); override;
  end;

  pIndex = ^Integer;

  tListPropEditor = class(tCustomPropEditor)
  protected
    Value: pString;
    SavedValue: String;
  public
    constructor CreateByNameValue(aOwner: TComponent; aName: String;
                  aList: tStringList; aValue: Pointer);
    procedure ButtonClicked(Sender: tObject);
    procedure DblClickEvent(Sender: tObject); override;
    procedure EnterEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
    procedure SelectNext;
    procedure SetBounds(aLeft, aTop, aWidth, aHeight: Integer); override;
    procedure SetListBox;
    procedure LostSelected; override;
  end;

  pBoolean = ^Boolean;

  tBooleanPropEditor = class(tListPropEditor)
  protected
    Value: pBoolean;
    SavedValue: Boolean;
  public
    constructor CreateByNameValue(aOwner: TComponent; aName: String;
                  aList: tStringList; aValue: Pointer);
    procedure ExitEvent(Sender: tObject); override;
  end;

  tStringPropEditor = class(tCustomPropEditor)
  protected
    Value: pString;
    SavedValue: String;
  public
    constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
    procedure DblClickEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
  end;

  tOpenDlgPropEditor = class(tCustomPropEditor)
  protected
    Value: pString;
    SavedValue: String;
  public
    procedure ButtonClicked(Sender: tObject);
    constructor CreateByNameValue(AOwner: TComponent; aName: String; aValue: Pointer);
    procedure EnterEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
    procedure SetBounds(aLeft, aTop, aWidth, aHeight: Integer); override;
  end;

  tOpenPicPropEditor = class(tOpenDlgPropEditor)
  public
```

```
    procedure ButtonClicked(Sender: tObject);
  end;

  tSeparator = class(tCustomPropEditor)
  protected
    procedure Paint; override;
  public
    procedure EnterEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
  end;

  tPropMenuItem = class(tCustomPropEditor)
  protected
    Notify: tNotifyEvent;
    procedure Paint; override;
  public
    constructor CreateByNameValue(AOwner: TComponent; aName: String; aNotify: tNotifyEvent);
    procedure ClickEvent(Sender: tObject); override;
    procedure EnterEvent(Sender: tObject); override;
    procedure ExitEvent(Sender: tObject); override;
  end;

  tGripPropMenuItem = class(tPropMenuItem)
  end;

  tPropPanel = class(tPanel)                                    //panel for
the propeditors
  private
    function GetPropEditor(aComp: tComponent): tComponent;
    procedure DoDeactivate(Sender: TObject);
  public
    Divider: Integer;
    PropEditorList: tPropEditorList;
    constructor Create(aOwner: tComponent); override;
    procedure Clear;
    function GetDataOwner: tComponent;
    procedure NewSelection;
    procedure SetCurrent(aComponent: tComponent);
    function SetupMenu: tPoint;
  end;

  tPopupMenu = class(TForm)                                     //popup
menu for the proppanel
  private
  public
    procedure DefaultFunc;
    procedure FormDeactivate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Popup;
    procedure Setup;
  end;

  tFieldData = class                    //field type definition
    fName, fType: String;
    constructor Create(aName, aType: String);
  end;

  tPointsForm = class(TForm)
  private
```

```
  public
    Panel1: TPanel;
    StringGrid1: TStringGrid;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Setup;
  end;

var PropPanel: tPropPanel;
    LayerList: tStringList;

implementation

uses CadBasic, CadLayer, CadMisc, CommonCAD;

//tPropList

procedure tPropList.GetAttributes(aString: String; var aList: tList);
var AW, BW: tAwkLine;
    i: Integer;
    S1, S2: String;
begin
  AW:= tAwkLine.Create(aString, '=');
  BW:=tAwkLine.Create(AW.Strings[0], ' ');
  S1:=BW.Strings[BW.Count-1];
  BW.Free;
  i:= 1;
  while i< AW.Count do
  begin
    BW:=tAwkLine.Create(AW.Strings[i], '"');
    S2:=BW.Strings[1];
    BW.Free;
    aList.Add(tFieldData.Create(S1, S2));
    Inc(i);
    if i< AW.Count then
    begin
      BW:=tAwkLine.Create(AW.Strings[i- 1], ' ');
      S1:=BW.Strings[BW.Count-1];
      BW.Free;
    end;
  end;
  AW.Free;
end;

function tPropList.Find(aName: String): tObject;
var i: Integer;
begin
  Result:= nil;
  for i:= 0 to Count- 1 do
    if UpperCase(tCustomProp(Items[i]).Name) = UpperCase(aName) then
    begin
      Result:= tCustomProp(Items[i]);
      Break;
    end;
end;

procedure tPropList.Assign(Source: TPersistent);
begin
end;

constructor tPropList.Create;
begin
  ShowMessage('You should use CREATEBY...');
```

```
    Halt;
  end;

  constructor tPropList.CreateBy(aOwner: tComponent);
  begin
    inherited Create;
    fOwner:= aOwner;
  end;

  procedure tPropList.CopyTo(aPropList: tPropList);
  var i: Integer;
      PO: tObject;
  begin
    for i:= 0 to Count- 1 do
      if (tCustomProp(Items[i]).Name <> 'Text') and
         (tCustomProp(Items[i]).Name <> 'Points') then
      begin
        PO:= aPropList.Find(tCustomProp(Items[i]).Name);
        if Assigned(PO) then
          tCustomProp(Items[i]).CopyTo(tCustomProp(PO));
      end;
  end;

  function tPropList.GetText: String;
  begin
    Drawing.DrawingFile.Clear;
    Drawing.DrawingFile.Add('');
    Drawing.DrawingFile.Index:= 0;
    WriteXml(False);
    Result:= Drawing.DrawingFile.Text;
  end;

  procedure tPropList.Kill(aName: String);
  var CP: tCustomProp;
  begin
    CP:= tCustomProp(Find(aName));
    Remove(CP);
    CP.Free;
  end;

  procedure tPropList.PushAll(aPropList: tPropList);
  var i: Integer;
      PO: tObject;
  begin
    for i:= 0 to Count- 1 do
    begin
      PO:= aPropList.Find(tCustomProp(Items[i]).Name);
      if Assigned(PO) then
        tCustomProp(Items[i]).CopyTo(tCustomProp(PO));
    end;
  end;

  procedure tPropList.ReadXml(aString: String);
  var AttList: tList;
      i: Integer;
      CP: tCustomProp;
  begin
    AttList:= tList.Create;
    GetAttributes(aString, AttList);
    for i:= 0 to AttList.Count- 1 do
    begin
      CP:= tCustomProp(Find(tFieldData(AttList.Items[i]).fName));
```

```
    if Assigned(CP) then
      CP.ReadXml(tFieldData(AttList.Items[i]).fType);
  end;
  AttList.Free;
end;

procedure tPropList.SetPropEditors;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
  begin
    tCustomProp(Items[i]).SetPropEditor;
    if tCustomProp(Items[i])= Default then
      PropPanel.PropEditorList.Default:= tComponent(PropPanel.PropEditorList.Last);
  end;
end;

procedure tPropList.SetText(aString: String);
begin
  ReadXml(aString);
end;

procedure tPropList.WriteXML(CloseTag: Boolean);
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tCustomProp(Items[i]).WriteXML;
  if CloseTag
    then Drawing.DrawingFile.Writeln('/>')
    else Drawing.DrawingFile.Writeln('>');
end;

//tCustomProp

constructor tCustomProp.CreateProp(aValue: Pointer; aName: String);
begin
  Value:= aValue;
  Name:= aName;
end;

//tStringProp

procedure tStringProp.CopyTo(aProp: tCustomProp);
begin
  String(aProp.Value^):= String(Value^);
end;

function tStringProp.GetValue: String;
begin
  Result:= String(Value^);
end;

procedure tStringProp.ReadXml(aString: String);
begin
  String(Value^):= aString;
end;

procedure tStringProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tStringPropEditor.CreateByNameValue(PropPanel, Name,
Value));
```

```
end;

procedure tStringProp.SetValue(aValue: String);
begin
  String(Value^):= aValue;
end;

procedure tStringProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ String(Value^)+ '"');
end;

//tNameProp

procedure tNameProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tNamePropEditor.CreateByNameValue(PropPanel, Name, Value));
end;

procedure tPictFileNameProp.SetPropEditor;
begin
  PropPanel.PropEditorList.Add(tOpenPicPropEditor.CreateByNameValue(PropPanel, Name,
Value));
end;

//tIntegerProp

procedure tIntegerProp.CopyTo(aProp: tCustomProp);
begin
  Integer(aProp.Value^):= Integer(Value^);
end;

function tIntegerProp.GetValue: String;
begin
  Result:= IntToStr(Integer(Value^));
end;

procedure tIntegerProp.ReadXml(aString: String);
begin
  Integer(Value^):= StrToInt(aString);
end;

procedure tIntegerProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tIntegerPropEditor.CreateByNameValue(PropPanel, Name,
Value));
end;

procedure tIntegerProp.SetValue(aValue: String);
begin
  Integer(Value^):= StrToInt(aValue);
end;

procedure tIntegerProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ IntToStr(Integer(Value^))+ '"');
end;

//tColorProp
```

```
procedure tColorProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tColorPropEditor.CreateByNameValue(PropPanel, Name,
Value));
end;

//tFloatProp

procedure tFloatProp.CopyTo(aProp: tCustomProp);
begin
  Double(aProp.Value^):= Double(Value^);
end;

function tFloatProp.GetValue: String;
begin
  Result:= FloatToStr(Double(Value^));
end;

procedure tFloatProp.ReadXml(aString: String);
begin
  Double(Value^):= StrToFloat(aString);
end;

procedure tFloatProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tFloatPropEditor.CreateByNameValue(PropPanel, Name,
Value));
end;

procedure tFloatProp.SetValue(aValue: String);
begin
  Double(Value^):= StrToFloat(aValue);
end;

procedure tFloatProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ FloatToStr(Double(Value^))+ '"');
end;

//tPointsProp

procedure tPointsProp.CopyTo(aProp: tCustomProp);
begin
  tPointList(aProp.Value^):= tPointList(Value^);
end;

function tPointsProp.GetValue: String;
begin
  Result:= tPointList(Value^).WriteToString;
end;

procedure tPointsProp.ReadXml(aString: String);
begin
  tPointList(Value^).ReadFromString(aString);
end;

procedure tPointsProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tPointsPropEditor.CreateByNameValue(PropPanel, Name,
```

```
Value));
end;

procedure tPointsProp.SetValue(aValue: String);
begin
  tPointList(Value^).ReadFromString(aValue);
end;

procedure tPointsProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ tPointList(Value^).WriteToString+
'"');
end;

//tListProp

constructor tListProp.CreateListProp(aValue: Pointer; aName: String; aList: tStringList);
begin
  inherited CreateProp(aValue, aName);
  List:= aList;
end;

procedure tListProp.CopyTo(aProp: tCustomProp);
begin
  String(aProp.Value^):= String(Value^);
end;

function tListProp.GetValue: String;
begin
  Result:= String(Value^);
end;

procedure tListProp.ReadXml(aString: String);
begin
  String(Value^):= aString;
end;

procedure tListProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tListPropEditor.CreateByNameValue(PropPanel, Name, List,
Value));
end;

procedure tListProp.SetValue(aValue: String);
begin
  String(Value^):= aValue;
end;

procedure tListProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ String(Value^)+ '"');
end;

//tBooleanProp

var slBoolean: tStringList;

constructor tBooleanProp.CreateProp(aValue: Pointer; aName: String);
begin
  inherited CreateListProp(aValue, aName, slBoolean);
end;
```

```
procedure tBooleanProp.CopyTo(aProp: tCustomProp);
begin
  Boolean(aProp.Value^):= Boolean(Value^);
end;

function tBooleanProp.GetValue: String;
begin
  if Boolean(Value^) then Result:= 'True'
                     else Result:= 'False';
end;

procedure tBooleanProp.ReadXml(aString: String);
begin
  if aString = slBoolean.Strings[0]
    then Boolean(Value^):= True
    else Boolean(Value^):= False;
end;

procedure tBooleanProp.SetPropEditor;
begin
  if not InternalUse then
    PropPanel.PropEditorList.Add(tBooleanPropEditor.CreateByNameValue(PropPanel, Name,
List, Value));
end;

procedure tBooleanProp.SetValue(aValue: String);
begin
  Boolean(Value^):= aValue = 'True';
end;

procedure tBooleanProp.WriteXML;
begin
  if Boolean(Value^)
    then Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ slBoolean.Strings[0]+ '"')
    else Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ slBoolean.Strings[1]+ '"');
end;

//tLayerProp

constructor tLayerProp.CreateProp(aValue: Pointer; aName: String);
begin
  inherited CreateListProp(aValue, aName, LayerList);
end;

//tCombinationListProp

procedure tCombinationListProp.CopyTo(aProp: tCustomProp); begin end;
function tCombinationListProp.GetValue: String; begin end;

procedure tCombinationListProp.ReadXml(aString: String);
begin
  tCombinationList(Value^).ReadFromString(aString);
end;

procedure tCombinationListProp.SetPropEditor; begin end;
procedure tCombinationListProp.SetValue(aValue: String); begin end;

procedure tCombinationListProp.WriteXML;
begin
  Drawing.DrawingFile.Write(' '+ LowerCase(Name)+ '="'+ tCombinationList(Value^).
WriteToString+ '"');
```

```
end;

//tMenuProp

constructor tMenuProp.CreateProp(aNotify: tNotifyEvent; aName: String);
begin
  Notify:= aNotify;
  Name:= aName;
end;

procedure tMenuProp.CopyTo(aProp: tCustomProp); begin end;
procedure tMenuProp.ReadXml(aString: String); begin end;

procedure tMenuProp.SetPropEditor;
begin
  PropPanel.PropEditorList.Add(tPropMenuItem.CreateByNameValue(PropPanel, Name, Notify));
end;

function tMenuProp.GetValue: String;
begin
  Result:= 'DD';
end;

procedure tMenuProp.SetValue; begin end;
procedure tMenuProp.WriteXml; begin end;

procedure tGripMenuProp.SetPropEditor;
begin
  PropPanel.PropEditorList.Add(tGripPropMenuItem.CreateByNameValue(PropPanel, Name,
Notify));
end;

//tPropEditorList

procedure tPropEditorList.Sort;
var i, j: Integer;
    P: tCustomPropEditor;
    SL: tStringList;
    S: String;
    PropL, FuncL, GripL: tList;
begin
  PropPanel.Divider:= 90;
  SL:= tStringList.Create;
  SL.Sorted:= True;
  for i:= 0 to Count-1 do
  begin
    P:= Items[i];
    S:= P.GetIdName;
    SL.Add(P.GetIdName);
  end;
  for i:= 0 to SL.Count-1 do
  begin
    S:= SL.Strings[i];
    for j:= 0 to Count-1 do
    begin
      P:= Items[j];
      if S= P.GetIdName then
      begin
        Move(j, i);
        Break;
      end;
    end;
```

```
    end;
    SL.Free;
    PropL:= tList.Create;
    FuncL:= tList.Create;
    GripL:= tList.Create;
    for i:= 0 to Count- 1 do
      if tComponent(Items[i]) is tGripPropMenuItem
        then GripL.Add(Items[i])
        else
      if tComponent(Items[i]) is tPropMenuItem
        then FuncL.Add(Items[i])
        else PropL.Add(Items[i]);
    Clear;
    if (GripL.Count> 0) and (Assigned(Drawing.Mouse.GripHolder))
  then                                                    //popup on the grip
    begin
      for i:= 0 to GripL.Count- 1 do
        Add(GripL.Items[i]);
      PropL.Free;
      FuncL.Free;
      GripL.Free;
      Exit;
    end;
    if FuncL.Count> 0 then                                                //normal
  object popup (it contains menu items too)
    begin
      for i:= 0 to FuncL.Count- 1 do
        Add(FuncL.Items[i]);
      Add(tSeparator.CreateByName(PropPanel, 'Anything'));
    end;
    for i:= 0 to PropL.Count- 1 do
      Add(PropL.Items[i]);
    PropL.Free;
    FuncL.Free;
  end;

  //PropListBox

  constructor tPropListBox.Create(aOwner: tComponent);
  begin
    inherited Create(aOwner);
    Parent:= PropPanel.Parent;
  //  Ctl3D:= False;
    OnClick:= ClickEvent;
    OnExit:= ExitEvent;
    with Font do
    begin
      Name:= 'MS Sans Serif';
      Size:= 8;
      Color:= clBlack;
    end;
  end;

  procedure tPropListBox.ClickEvent(Sender: tObject);
  begin
    ExitEvent(Sender);
  end;

  procedure tPropListBox.ExitEvent(Sender: tObject);
  begin
    with (Owner as tCustomPropEditor) do
    begin
```

```
      SetEditorValue(Items[ItemIndex]);
      ExitEvent(Sender);
    end;
    Hide;
  end;

  function tPropListBox.GetIndex(aString: String): Integer;
  var i: Integer;
  begin
  Result:= 0;
  try
    for i:= 0 to Items.Count - 1 do
      if Items[i] = aString then
      begin
        Result:= i;
        Break;
      end;
  except
     Result:=0;
  end;
  end;

  procedure tPropListBox.Setting(List: tStringList);
  begin
    Items.Assign(List);
  end;

  //tCustomButton

  constructor tCustomButton.Create(aOwner: tComponent);
  begin
    inherited Create(aOwner);
    Parent:= tCustomPropEditor(aOwner);
    OnClick:= ClickEvent;
  end;

  procedure tCustomButton.ClickEvent(Sender: tObject);
  begin
    if Parent is tListPropEditor then
       tListPropEditor(Parent).ButtonClicked(Sender);
    if Parent is tColorPropEditor then
       tColorPropEditor(Parent).ButtonClicked(Sender);
    if Parent is tPointsPropEditor then
       tPointsPropEditor(Parent).ButtonClicked(Sender);
    //SendMessage(Parent.Handle, wmSpeedBtnClicked, 0, 0);
  end;

  //tPointsButton

  constructor tPointsButton.Create(aOwner: tComponent);
  begin
    inherited Create(aOwner);
  //  Glyph.Handle:= LoadBitMap(hInstance, 'BITMAP_1');
  end;

  //tArrowButton

  constructor tArrowButton.Create(aOwner: tComponent);
  begin
    inherited Create(aOwner);
  //  Glyph.Handle:= LoadBitMap(hInstance, 'BITMAP_2');
  end;
```

```
//tPropEdit

constructor tPropEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Parent:= tCustomPropEditor(aOwner);
  OnDblClick:= DblClickEvent;
  OnKeyPress:= KeyPressEvent;
  OnChange:= ChangeEvent;
  Changed:= False;
//  Ctl3D:= False;
  BorderStyle:= bsNone;
  with Font do
  begin
    Name:= 'MS Sans Serif';
    Size:= 8;
    Color:= clBlack;
  end;
end;

procedure tPropEdit.ChangeEvent(Sender: tObject);
begin
  Changed:= True;
end;

procedure tPropEdit.DblClickEvent(Sender: tObject);
begin
  tCustomPropEditor(Parent).DblClickEvent(Sender);
end;

procedure tPropEdit.KeyPressEvent(Sender:TObject; var Key: Char);
begin
  case Key of
    #13: begin
          (Parent as tCustomPropEditor).ExitEvent(Sender);
           Key:=#0;
         end;
  end;
end;

//tCustomPropEditor

constructor tCustomPropEditor.CreateByName(AOwner: TComponent; aName: String);
begin
  inherited Create(AOwner);
  OnClick:= ClickEvent;
  OnEnter:= EnterEvent;
  OnExit:= ExitEvent;
  OnMouseMove:= MouseMoveEvent;
  Caption:= aName;
  Editor:= tPropEdit.Create(Self);
  Editor.onChange:= ChangeEditorEvent;
  Dirty:= False;
end;

destructor tCustomPropEditor.Destroy;
begin
  inherited;
end;

procedure tCustomPropEditor.AddOwner(Sender: tObject);
```

```
begin
end;

procedure tCustomPropEditor.ChangeEditorEvent(Sender: tObject);
begin
  Dirty:= True;
end;

procedure tCustomPropEditor.ClickEvent(Sender: tObject);
begin
end;

procedure tCustomPropEditor.DblClickEvent(Sender: tObject);
begin
end;

procedure tCustomPropEditor.EnterEvent(Sender: tObject);
begin
  PropPanel.NewSelection;
  Selected:= True;
  Editor.Show;
  Editor.SetFocus;
  Invalidate;
end;

procedure tCustomPropEditor.ExitEvent(Sender: tObject);
begin
end;

function tCustomPropEditor.GetIdName: string;
begin
  Result:= Caption;
end;

function tCustomPropEditor.GetValue: Pointer;
begin
  Result:= Value;
end;

procedure tCustomPropEditor.LostSelected;
begin
  Selected:= False;
  Editor.Hide;
  Invalidate;
end;

procedure tCustomPropEditor.MouseMoveEvent(Sender :tObject; Shift: tShiftState; X,
Y: Integer);
//procedure tCustomPropEditor.CMMouseEnter(var Message: TMessage);
begin
  Screen.Cursor:= crDefault;
  SetFocus;
end;

procedure tCustomPropEditor.Paint;
var
  RC: TRect;
  HR: QRegionH;
begin
  if not Assigned(PropPanel) then Exit;
  RC := GetClientRect;
  with Canvas do
```

```
    begin
      Brush.Style := bsSolid;
      with Font do
      begin
        Name:= 'MS Sans Serif';
        Size:= 8;
      end;
      if Selected then
      begin
        Brush.Color := clHighlight;
        FillRect(RC);
        Font.Color:= clHighlightedText; //clWhite;
        RC:= Rect(RC.Left+1, RC.Top+1, PropPanel.Divider-2, RC.Bottom-1);
        TextRect(RC, 3, 1, Caption);
      end else
      begin
        Brush.Color := clBtnFace;
        FillRect(RC);
        Font.Color:= clNormalText;//clBlack;
        RC:= Rect(RC.Left+1, RC.Top+1, PropPanel.Divider-2, RC.Bottom-1);
        TextRect(RC, 3, 1, Caption);
        //HR:= CreateRectRgn(PropPanel.Divider,1,Width-2,Height-1);
        HR:=QRegion_Create(PropPanel.Divider,1,Width-2,Height-1, QRegionRegionType_Rectangle);
        //SelectClipRgn(Handle, HR);
        QPainter_SetClipRegion(Handle, HR);
        TextOut(PropPanel.Divider+4,1,Editor.Text);
        //DeleteObject(HR);
        QRegion_Destroy(HR);
      end;
    end;
  end;

  procedure tCustomPropEditor.SendResult;
  begin
    Drawing.SelectList.ChangeValue(Caption);
    Drawing.Invalidate;
  end;

  procedure tCustomPropEditor.SetBounds(aLeft, aTop, aWidth, aHeight: Integer);
  begin
    inherited SetBounds(aLeft, aTop, aWidth, aHeight);
    Editor.SetBounds(PropPanel.Divider+2,1,Width- PropPanel.Divider -2, Height-1);
  end;

  procedure tCustomPropEditor.SetEditorValue(S: String);
  begin
    Editor.Text:= S;
  end;

  //tFloatPrcpEditor

  constructor tFloatPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
  Pointer);
  var S: String;
  var D: Real;
  begin
    inherited CreateByName(AOwner, aName);
    D := Double(aValue^);
    if (D>1000000) or (D<0.0001) then
      S:= FloatToStrF(Double(aValue^), ffExponent, 6, 3)
    else
```

```
    S:= FloatToStrF(Double(aValue^), ffFixed, 18, 4);
  Editor.Text:= S;
  SavedValue:= pDouble(aValue)^;
  Value:= aValue;
  Dirty:= False;
end;

procedure tFloatPropEditor.ExitEvent(Sender: tObject);
var Temp: Double;
begin
  if Dirty then
  begin
    try
      Temp:=  StrToFloat(Editor.Text);
      SavedValue:= Temp;
      Value^:= Temp;
      SendResult;
    except
      MessageDlg(''''+ Editor.Text+ ''' is not a valid floating point value',
                mtError, [mbOK], 0);
      Editor.Text:= FloatToStrF(SavedValue, ffFixed, 18, 4);
    end;
    Dirty:= False;
  end;
end;

//tIntegerPropEditor

constructor tIntegerPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
Pointer);
begin
  inherited CreateByName(AOwner, aName);
  Editor.Text:= IntToStr(pInteger(aValue)^);
  Value:= aValue;
  SavedValue:= pInteger(aValue)^;
  Dirty:= False;
end;

procedure tIntegerPropEditor.ExitEvent(Sender: tObject);
var Temp: Integer;
begin
  if Dirty then
  begin
    try
      Temp:= StrToInt(Editor.Text);
      SavedValue:= Temp;
      Value^:= Temp;
      SendResult;
    except
      MessageDlg(''''+ Editor.Text+ ''' is not a valid integer value',
                mtError, [mbOK], 0);
      Editor.Text:= IntToStr(SavedValue);
    end;
    Dirty:= False;
  end;
end;

//tColorPropEditor

constructor tColorPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
Pointer);
begin
```

```
  inherited CreateByNameValue(AOwner, aName, aValue);
  Button:= tPointsButton.Create(Self);
  with Button do
  begin
    Hide;
  end;
end;

procedure tColorPropEditor.ButtonClicked(Sender: tObject);
var CD: tColorDialog;
begin
  CD:= tColorDialog.Create(Application);
  CD.Color:= StrToInt(Editor.Text);
  if CD.Execute then Editor.Text:= IntToStr(CD.Color);
  CD.Free;
end;

procedure tColorPropEditor.EnterEvent(Sender: tObject);
begin
  PropPanel.NewSelection;
  Selected:= True;
  Editor.Show;
  Editor.SetFocus;
  Button.Show;
  Invalidate;
end;

procedure tColorPropEditor.ExitEvent(Sender: tObject);
begin
  inherited ExitEvent(Sender);
  Button.Hide;
  Invalidate;
end;

procedure tColorPropEditor.SetBounds(aLeft, aTop, aWidth, aHeight: Integer);
begin
  inherited SetBounds(aLeft, aTop, aWidth, aHeight);
  if Editor<>nil then Editor.SetBounds(PropPanel.Divider+2,1,Width- PropPanel.Divider -18,
Height-1);
  if Button<>nil then Button.SetBounds(Width-15,0,15,15);
end;

//tListPropEditor

constructor tListPropEditor.CreateByNameValue(AOwner: TComponent; aName: String;
                              aList: tStringList; aValue: Pointer);
begin
  inherited CreateByName(AOwner, aName);
  SavedValue:= pString(aValue)^;
  Value:= aValue;
  Button:= tArrowButton.Create(Self);
  with Button do
  begin
    Hide;
  end;
  ListBox:= tPropListBox.Create(Self);
  with ListBox do
  begin
    Setting(aList);
    ItemIndex:= GetIndex(pString(aValue)^);
    Hide;
  end;
```

```
    Editor.Text:= pString(aValue)^;
    Dirty:= False;
  end;

  procedure tListPropEditor.ButtonClicked(Sender: tObject);
  begin
    if ListBox.Visible then ListBox.Hide else
    begin
      SetListBox;
      ListBox.ItemIndex:= ListBox.GetIndex(SavedValue);
      ListBox.Show;
      ListBox.SetFocus;
      ListBox.Show;
    end;
  end;

  procedure tListPropEditor.DblClickEvent(Sender: tObject);
  begin
    SelectNext;
  end;

  procedure tListPropEditor.EnterEvent(Sender: tObject);
  begin
    PropPanel.NewSelection;
    Selected:= True;
    Editor.Show;
    Editor.SetFocus;
    if Button<>nil then Button.Show;
    Invalidate;
  end;

  procedure tListPropEditor.ExitEvent(Sender: tObject);
  var i: Integer;
      Need: Boolean;
  begin
    Need:= True;
    for i:= 0 to ListBox.Items.Count-1 do
    begin
      if Editor.Text= ListBox.Items[i] then
      begin
        Need:= False;
        Value^:= ListBox.Items[i];
        if SavedValue<> ListBox.Items[i] then SendResult;
        SavedValue:= ListBox.Items[i];
        Break;
      end;
    end;
    if Need then
    begin
      ShowMessage(''''+ Editor.Text+ ''' is not a valid list item.');
      Editor.Text:= SavedValue;
    end;
    ListBox.Hide;
    Dirty:= False;
  end;

  procedure tListPropEditor.SelectNext;
  begin
    if ListBox.ItemIndex<ListBox.Items.Count-1
      then ListBox.ItemIndex:= ListBox.ItemIndex+ 1
      else ListBox.ItemIndex:= 0;
    Editor.Text:= ListBox.Items[ListBox.ItemIndex];
```

```
    ExitEvent(Self);
  end;

  procedure tListPropEditor.SetBounds(aLeft, aTop, aWidth, aHeight: Integer);
  begin
    inherited SetBounds(aLeft, aTop, aWidth, aHeight);
    if Editor<>nil then Editor.SetBounds(PropPanel.Divider+2,1,Width- PropPanel.Divider -18,
  Height-1);
    if Button<>nil then Button.SetBounds(Width-15,0,15,15);
    if ListBox<>nil then SetListBox;
  end;

  procedure tListPropEditor.SetListBox;
  var P: tPoint;
      i, H1, H2, o: Integer;
  begin
    P:= Point(Button.Left+ 15, Button.Top+ 15);
    P:= PropPanel.ScreenToClient(ClientToScreen(P));
    with ListBox do
    begin
      Width:= Editor.Width+ 17;
      i:= Items.Count;
      if i>7 then H1:= 7* ItemHeight else H1:= i* ItemHeight;
      Inc(H1, 2);
      H2:= PropPanel.ClientHeight- Top;
      o:= H2 div ItemHeight;
      H2:= o* ItemHeight+ 2;
      if H2< H1 then Height:= H2 else Height:= H1;
      Left:= P.X- Width;
      Top:=  P.Y;
    end;
  end;

  procedure tListPropEditor.LostSelected;
  begin
    Selected:= False;
    Editor.Hide;
    Button.Hide;
    ListBox.Hide;
    Invalidate;
  end;

  //tBooleanPropEditor

  constructor tBooleanPropEditor.CreateByNameValue(AOwner: TComponent; aName: String;
                      aList: tStringList; aValue: Pointer);
  var i: String;
  begin
    if pBoolean(aValue)^ then i:= 'True'
                         else i:= 'False';
    inherited CreateByNameValue(AOwner, aName, aList, @i);
  //  if pBoolean(aValue)^ then Editor.Text:= slBoolean.Strings[0]
  //                       else Editor.Text:= slBoolean.Strings[1];
    SavedValue:= pBoolean(aValue)^;
    Value:= aValue;
  end;

  procedure tBooleanPropEditor.ExitEvent(Sender: tObject);
  var i: Integer;
      Need, Temp: Boolean;
  begin
    if ((ListBox.ItemIndex=0)and(SavedValue)) then Exit;
```

```
    if ((ListBox.ItemIndex=1)and(not SavedValue)) then Exit;
    Need:= True;
    Temp:= True;
    for i:= 0 to ListBox.Items.Count-1 do
    begin
      if Editor.Text= ListBox.Items[i] then
      begin
        Need:= False;
        if i=0 then Temp:= True;
        if i=1 then Temp:= False;
        Value^:= Temp;
        if Temp<> SavedValue then
        begin
          SendResult;
        end;
        SavedValue:= Temp;
        Break;
      end;
    end;
    if Need then
    begin
      MessageDlg(''''+ Editor.Text+ ''' is not a valid boolean value.',
                 mtError, [mbOK], 0);
      if SavedValue then Editor.Text:= ListBox.Items[0]
                    else Editor.Text:= ListBox.Items[1];
    end;
    Dirty:= False;
  end;

  constructor tPointsPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
  Pointer);
  begin
    inherited CreateByName(AOwner, aName);
    Value:= aValue;
    Editor.Text:= '[click]';
    Button:= tPointsButton.Create(Self);
    with Button do
    begin
      Parent:= Self;
      Hide;
    end;
  end;

  procedure tPointsPropEditor.ButtonClicked(Sender: tObject);
  begin
    DblClickEvent(Self);
  end;

  procedure tPointsPropEditor.DblClickEvent(Sender: tObject);
  var PF: tPointsForm;
      PL: tPointList;
      i, j: Integer;
      AW1, AW2: tAwkLine;
      SS: String;
  begin
    PF:= tPointsForm.CreateNew(Application);
    PF.Setup;
    PL:= tPointList(Value^);
    AW1:= tAwkLine.Create(PL.WriteToString, '/');
    PF.StringGrid1.RowCount:= AW1.Count+ 1;
    PF.StringGrid1.ColCount:= 3;
    PF.StringGrid1.FixedCols:= 0;
```

```
    PF.StringGrid1.Cells[0, 0]:= 'X';
    PF.StringGrid1.Cells[1, 0]:= 'Y';
    PF.StringGrid1.Cells[2, 0]:= 'Z';
    for i:= 0 to AW1.Count- 1 do
    begin
      AW2:= tAwkLine.Create(Aw1[i], '*');
      if AW2[0]<>'-1' then PF.StringGrid1.Cells[0, i+ 1]:= AW2[0];
      if AW2[1]<>'-1' then PF.StringGrid1.Cells[1, i+ 1]:= AW2[1];
      if AW2[2]<>'-1' then PF.StringGrid1.Cells[2, i+ 1]:= AW2[2];
      AW2.Free;
    end;
    AW1.Free;
    if PF.ShowModal= mrOK then
    begin
      SS:= '';
      for i:= 1 to PF.StringGrid1.RowCount- 1 do
      begin
        if i<> 1 then SS:= SS+ '/';
        for j:= 0 to 2 do if PF.StringGrid1.Cells[j, i]<> '' then
        begin
          if j<> 0 then SS:= SS+ '*';
          SS:= SS+ PF.StringGrid1.Cells[j, i];
        end
        else
        begin
          if j<> 0 then SS:= SS+ '*';
          SS:= SS+ '-1';
        end;
      end;
      PL.Clear;
      PL.ReadFromString(SS);
      Drawing.GripList.Clear;
      PL.PutGrips;
      Drawing.Invalidate;
    end;
    PF.Free;
  end;

  procedure tPointsPropEditor.EnterEvent(Sender: tObject);
  begin
    PropPanel.NewSelection;
    Selected:= True;
    Editor.Show;
    Editor.SetFocus;
    if Button<>nil then Button.Show;
    Invalidate;
  end;

  procedure tPointsPropEditor.ExitEvent(Sender: tObject);
  begin
    Button.Hide;
    Editor.Text:= '[click]';
    Dirty:= False;
  end;

  procedure tPointsPropEditor.SetBounds(aLeft, aTop, aWidth, aHeight: Integer);
  begin
    inherited SetBounds(aLeft, aTop, aWidth, aHeight);
    if Editor<>nil then Editor.SetBounds(PropPanel.Divider+2,1,Width- PropPanel.Divider -18,
  Height-1);
    if Button<>nil then Button.SetBounds(Width-15,0,15,15);
  end;
```

```
//tNamePropEditor

constructor tNamePropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
Pointer);
begin
  inherited CreateByName(AOwner, aName);
  Editor.Text:= pString(aValue)^;
  Value:= pString(aValue);
  SavedValue:= Value^;
  Dirty:= False;
end;

procedure tNamePropEditor.ExitEvent(Sender: tObject);
begin
  if Dirty then
  begin
    Value^:= Editor.Text;
    SendResult;
    SavedValue:= Editor.Text;
    tBasic(PropPanel.GetDataOwner).NameChanged;
    Dirty:= False;
  end;
end;

function tNamePropEditor.GetValue: pString;
begin
  Result:= Value;
end;

//tStringPropEditor

constructor tStringPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
Pointer);
begin
  inherited CreateByName(AOwner, aName);
  Editor.Text:= pString(aValue)^;
  Value:= pString(aValue);
  SavedValue:= Value^;
  Dirty:= False
end;

procedure tStringPropEditor.DblClickEvent(Sender: tObject);
begin
  Value^:= InputBox('Change text', 'New value', Value^);
  Drawing.Invalidate;
end;

procedure tStringPropEditor.ExitEvent(Sender: tObject);
begin
  if Dirty then
  begin
    Value^:= Editor.Text;
    SendResult;
    SavedValue:= Editor.Text;
    Dirty:= False;
  end;
end;

//tOpenDialogPropEditor

constructor tOpenDlgPropEditor.CreateByNameValue(AOwner: TComponent; aName: String; aValue:
```

```
Pointer);
begin
  inherited CreateByName(AOwner, aName);
  Editor.Text:= pString(aValue)^;
  Value:= pString(aValue);
  Button:= tPointsButton.Create(Self);
  with Button do
  begin
    Parent:= Self;
    Hide;
  end;
  SavedValue:= Value^;
end;

procedure tOpenDlgPropEditor.ButtonClicked(Sender: tObject);
var OD: tOpenDialog;
begin
  OD:= tOpenDialog.Create(Application);
  OD.FileName:= Editor.Text;
  if OD.Execute then Editor.Text:= OD.FileName;
  OD.Free;
end;

procedure tOpenDlgPropEditor.EnterEvent(Sender: tObject);
begin
  PropPanel.NewSelection;
  Selected:= True;
  Editor.Show;
  Editor.SetFocus;
  if Button<>nil then Button.Show;
  Invalidate;
end;

procedure tOpenDlgPropEditor.ExitEvent(Sender: tObject);
begin
  Value^:= Editor.Text;
  Button.Hide;
  if Editor.Text<> SavedValue then SendResult;
  SavedValue:= Editor.Text;
  Dirty:= False;
end;

procedure tOpenDlgPropEditor.SetBounds(aLeft, aTop, aWidth, aHeight: Integer);
begin
  inherited SetBounds(aLeft, aTop, aWidth, aHeight);
  if Editor<>nil then Editor.SetBounds(PropPanel.Divider+2,1,Width- PropPanel.Divider -18,
Height-1);
  if Button<>nil then Button.SetBounds(Width-15,0,15,15);
end;

//tOpenPicPropEditor

procedure tOpenPicPropEditor.ButtonClicked(Sender: tObject);
//var OD: tOpenPictureDialog;
begin
{
  OD:= tOpenPictureDialog.Create(Application);
  OD.FileName:= Editor.Text;
  if OD.Execute then Editor.Text:= OD.FileName;
  OD.Free;
}
end;
```

```
//tSeparator

procedure tSeparator.Paint;
var
  RC: TRect;
begin
  RC := GetClientRect;
  with Canvas do
  begin
    Brush.Style := bsSolid;
    Brush.Color := clBtnFace;
    FillRect(RC);
    Pen.Color:= clGray;
    PolyLine([Point(2,5),Point(RC.Right-2, 5)]);
    Pen.Color:= clWhite;
    PolyLine([Point(2,6),Point(RC.Right-2, 6)]);
  end;
end;

procedure tSeparator.EnterEvent(Sender: tObject);
begin
end;

procedure tSeparator.ExitEvent(Sender: tObject);
begin
end;

//tPropMenuItem

procedure tPropMenuItem.Paint;
var
  RC: TRect;
begin
  RC := GetClientRect;
  with Canvas do
  begin
    Brush.Style := bsSolid;
    with Font do
    begin
      Name:= 'MS Sans Serif';
      Size:= 8;
    end;
    if Selected then
    begin
      Brush.Color := clNavy;
      FillRect(RC);
      Font.Color:= clWhite;
    end else
    begin
      Brush.Color := clBtnFace;
      FillRect(RC);
      Font.Color:= clBlack;
    end;
    TextRect(RC, 3, 1, Caption);
  end;
end;

constructor tPropMenuItem.CreateByNameValue(AOwner: TComponent;
      aName: String; aNotify: tNotifyEvent);
begin
  inherited CreateByName(AOwner, aName);
```

```
  Notify:= aNotify;
end;

procedure tPropMenuItem.ClickEvent(Sender: tObject);
begin
  Drawing.PopupInspector.Close;
  if Assigned(Notify) then Notify(Drawing);
end;

procedure tPropMenuItem.EnterEvent(Sender: tObject);
begin
  PropPanel.NewSelection;
  Selected:= True;
  Invalidate;
end;

procedure tPropMenuItem.ExitEvent(Sender: tObject);
begin
end;

// tPropPanel

function tPropPanel.GetPropEditor(aComp: tComponent): tComponent;
begin
  if aComp is tCustomPropEditor then Result:= aComp else
  if tControl(aComp).Owner is tCustomPropEditor
    then Result:= tControl(aComp).Owner
    else Result:= nil;
end;

procedure tPropPanel.DoDeactivate(Sender: TObject);
begin
  if Assigned(tCustomPropEditor(tForm(Owner).ActiveControl)) then
    tCustomPropEditor(GetPropEditor(tCustomPropEditor(tForm(Owner).ActiveControl))).OnExit
(Self);
end;

constructor tPropPanel.Create(aOwner: tComponent);
begin
  inherited Create(aOwner);
  PropEditorList:= tPropEditorList.Create;
  Parent:= tForm(aOwner);
  Align:= alClient;
  tForm(aOwner).OnDeactivate:= DoDeactivate;
end;

procedure tPropPanel.Clear;
var i: Integer;
begin
  for i:= 0 to ControlCount- 1 do
    tCustomPropEditor(Controls[0]).Free;                                    //the first
object need delete always
  PropEditorList.Clear;
end;

function tPropPanel.GetDataOwner: tComponent;
begin
  Result:= Drawing.SelectList.GetSelection;
end;

procedure tPropPanel.NewSelection;
var i: Integer;
```

```
begin
  for i:= 0 to ControlCount-1 do
    with tCustomPropEditor(Controls[i]) do
      if Selected then LostSelected;
end;

procedure tPropPanel.SetCurrent(aComponent: tComponent);
begin
  Clear;
  tBasic(aComponent).PropList.SetPropEditors;
end;

function tPropPanel.SetupMenu: tPoint;
var i, j, ItemHeight, ItemWidth: Integer;
begin
  if PropEditorList.Count= 0 then Exit;
  PropEditorList.Sort;
  ItemWidth:= 180;
  ItemHeight:= 15;
  j:= 0;
  for i:= 0 to PropEditorList.Count-1 do
    with tCustomPropEditor(PropEditorList.Items[i]) do
    begin
      Parent:= Self;
      SetBounds(2, 2+ i* ItemHeight, ItemWidth, ItemHeight);
      Tabstop:= True;
      TabOrder:= i+ 1;
      j:= i;
      Editor.Hide;
    end;
  Result:= Point(ItemWidth+ 4, (j+ 1)* ItemHeight+ 4);
end;

//tPopupMenu

procedure TPopupMenu.FormDeactivate(Sender: TObject);
begin
//  Screen.OnActiveFormChange:= nil;
  Close;
  PropPanel.Visible:= False;
end;

procedure TPopupMenu.FormResize(Sender: TObject);
begin
  if (Left+ Width)> Screen.Width then Left:= Left- Width;
  if (Top+ Height)> Screen.Height then Top:= Screen.Height- Height;
end;

procedure tPopupMenu.DefaultFunc;
begin
  Popup;
  if Assigned(PropPanel.PropEditorList.Default) then
    tCustomPropEditor(PropPanel.PropEditorList.Default).DblClickEvent(Self);
end;

procedure tPopupMenu.Popup;
var P: tPoint;
begin
  if not Assigned(PropPanel) then
    PropPanel:= tPropPanel.Create(Self);
  PropPanel.Visible:= True;                                    //the
inspector is handleable
```

```
  PropPanel.SetCurrent(Drawing.SelectList.GetSelection);
  P:= PropPanel.SetupMenu;
  Width:= P.X;
  Height:= P.Y;
  GetCursorPos(P);
  if (P.X+ Width+ 5)< Screen.Width
    then Left:= P.X
    else Left:= Screen.Width- Width- 5;
  if P.Y+ Height+ 33< Screen.Height
    then Top:= P.Y
    else Top:= Screen.Height- Height- 33;
  Show;
  //Screen.OnActiveFormChange:= FormDeactivate;
end;

procedure tPopupMenu.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  case Key of
    Key_ESCAPE: FormDeactivate(tForm(Owner));
  end;
  //Key:= 0;
end;

procedure tPopupMenu.SetUp;
begin
  Borderstyle:= fbsNone;
  Keypreview:= True;
  onDeactivate:= FormDeactivate;
  onResize:= FormResize;
  onKeyDown:= FormKeyDown;
end;

constructor tFieldData.Create(aName, aType: String);
begin
  fName:= aName;
  fType:= aType;
end;

procedure TPointsForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
{
  try
    with tRegistry.Create do
    begin
      OpenKey('\Software\'+ Application.Title+ '\Edit points', True);
      WriteInteger('Left', Left);
      WriteInteger('Top', Top);
      WriteInteger('Width', Width);
      WriteInteger('Height', Height);
      Free;
    end;
  except
  end;
}
end;

procedure TPointsForm.Setup;
begin
  Caption:= 'Edit points';
  Panel1:= tPanel.Create(Self);
```

```
    onClose:= FormClose;
    with Panel1 do
    begin
      Parent:= Self;
      Width:= 107;
      Align:= alRight;
    end;
    with tButton.Create(Self) do
    begin
      Parent:= Panel1;
      Left:= 16;
      Top:= 24;
      Width:= 75;
      Height:= 25;
      Caption:= 'OK';
      Default:= True;
      ModalResult:= mrOK;
    end;
    with tButton.Create(Self) do
    begin
      Parent:= Panel1;
      Left:= 16;
      Top:= 64;
      Width:= 75;
      Height:= 25;
      Caption:= 'Cancel';
      ModalResult:= mrCancel;
    end;
    StringGrid1:= tStringGrid.Create(Self);
    with StringGrid1 do
    begin
      Parent:= Self;
      Align:= alClient;
      Options:= Options + [goEditing];
      DefaultRowHeight:= 15;
    end;
  {
    try
      with tRegistry.Create do
      begin
        OpenKey('\Software\'+ Application.Title+ '\Edit points', False);          //
  HKEY_CURRENT_USER
        Left:= ReadInteger('Left');
        Top:= ReadInteger('Top');
        Width:= ReadInteger('Width');
        Height:= ReadInteger('Height');
        Free;
      end;
    except
      ShowMessage('The showing of the first time will generate a compiler exeption.');
    end;
  }
  end;

  initialization

    slBoolean:= tStringList.Create;   //it's declared before part of tBooleanProp
  implementation
    with slBoolean do
    begin
      Add('True');
      Add('False');
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadRubber;

interface

uses Classes, QGraphics, QT,
     CadBasic, CadMisc, Types;

type
  tRubberLine = class(tComponent)
  private
    Canvas: tCanvas;
    procedure Show; virtual;
  public
    Visible: Boolean;
    First, Second, Last: t3dPoint;
    constructor CreateBy(aOwner: tComponent; aCanvas: tCanvas);
    function Hide: tRect;
    procedure Paint;
    procedure Start(aPoint: tPoint);
    procedure ExpandTo(aPoint: tPoint);
  end;

  tRubberBox = class(tRubberLine)
  private
    procedure Show; override;
  public
  end;

implementation

uses QDialogs,
     CommonCAD;

procedure tRubberLine.Show;
var FP, SP: tPoint;
begin
  FP:= Point(Round(First.X), Round(First.Y));
  SP:= Point(Round(Second.X), Round(Second.Y));
  with Canvas do
  begin
    Pen.Width:= 1;
    Pen.Color:= clRed;
    Pen.Style:= psDot;
    Polyline([FP, SP]);
    Pen.Style:= psSolid;
  end;
  Visible:= True;
```

```
end;

constructor tRubberLine.CreateBy(aOwner: tComponent; aCanvas: tCanvas);
begin
  inherited Create(aOwner);
  Canvas:= aCanvas;
  Visible:= False;
end;

procedure tRubberLine.ExpandTo(aPoint: tPoint);
begin
  Last:= Second;
  Second.X:= aPoint.X;
  Second.Y:= aPoint.Y;
  Drawing.Invalidate;
end;

function tRubberLine.Hide: tRect;
begin
  Show;
  Result:= Rect(Round(First.X), Round(First.Y), Round(Second.X), Round(Second.Y));
  Visible:= False;
end;

procedure tRubberLine.Paint;
begin
  if Visible then Show;
end;

procedure tRubberLine.Start(aPoint: tPoint);
begin
  First.X:= aPoint.X;
  First.Y:= aPoint.Y;
  Second.X:= aPoint.X;
  Second.Y:= aPoint.Y;
  Visible:= True;
end;

procedure tRubberBox.Show;
var P1, P2: tPoint;
begin
  if not Assigned(Canvas) then ShowMessage('Rubbers''s canvas isn''t initialised');
  if First.X< Second.X then
  begin
    P1.X:= Round(First.X);
    P2.X:= Round(Second.X);
  end else
  begin
    P1.X:= Round(Second.X);
    P2.X:= Round(First.X);
  end;
  if First.Y< Second.Y then
  begin
    P1.Y:= Round(First.Y);
    P2.Y:= Round(Second.Y);
  end else
```

```
    begin
      P1.Y:= Round(Second.Y);
      P2.Y:= Round(First.Y);
    end;
    with Canvas do
    begin
      Brush.Style:= bsSolid;
      Brush.Color:= clWhite;
      DrawFocusRect(Rect(P1.X, P1.Y, P2.X, P2.Y));
    end;
    Visible:= True;
  end;

  end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadSelect;

interface

uses Classes, QGraphics, QT, Types,
     CadBasic;

type
 tSelectList = class(tList)                                                    //
collection of selected objects
  private
    fVirtualComp: tDrawComponent;
    function GetVirtualComp: tDrawComponent;
    function IsAlready(aComponent: tComponent): Boolean;
    function GetElement: tDrawComponent;
    property VirtualComp: tDrawComponent read GetVirtualComp;
  public
    constructor Create;
    destructor Destroy; override;
    procedure AddBy(aComponent: tComponent; shiftPressed: Boolean);
    procedure AddByFrame(aRect: tRect);
    procedure ChangeValue(aName: String);
    procedure Clear; override;
    procedure Clone;
    procedure CopyToClipboard;
    procedure DeleteComponent;
    function GetSelection: tComponent;
    procedure Mirror;
    procedure MoveTo(X, Y, Z: Integer);
    procedure PushToUndo;
    procedure PasteFromClipboard;
    procedure RotateByRubberLine;
    procedure Rotate3dByRubberLine;
    procedure ScaleByRubberLine;

    procedure GripCopy(Sender: TObject);
    procedure GripMirror(Sender: TObject);
    procedure GripMove(Sender: TObject);
    procedure GripRotate(Sender: TObject);
    procedure GripRotate3D_X(Sender: TObject);
    procedure GripRotate3D_Y(Sender: TObject);
    procedure GripScale(Sender: TObject);
  end;

implementation

uses QClipbrd, SysUtils,
     CadGrip, CadMisc, CadProperty, CadUndo, CommonCad;

function tSelectList.GetVirtualComp: tDrawComponent;
var i: Integer;
    PL: tPropList;
```

```
begin
  PL:= tBasic(Items[0]).PropList;                                    //find the
shortest proplist
  for i:= 1 to Count- 1 do                                           //
    if tBasic(Items[i]).PropList.Count < PL.Count then               //
      PL:= tBasic(Items[i]).PropList;                                //
  fVirtualComp.PropList.Clear;                                       //clearing
of the PropList
  for i:= 0 to PL.Count- 1 do                                        //add props
to the PropList
    if (PL.Items[i] <> tMenuProp) and                                //
       (tCustomProp(PL.Items[i]).Name <> 'Name') then                //
      fVirtualComp.PropList.Add(PL.Items[i]);                        //
  Result:= fVirtualComp;                                             //warning:
this methode is a draft only
end;

function tSelectList.IsAlready(aComponent: tComponent): Boolean;
var i: Integer;
begin
  Result:= False;
  for i:= 0 to Count- 1 do
    if tComponent(Items[i])= aComponent then
    begin
      Result:= True;
      Break;
    end;
end;

function tSelectList.GetElement: tDrawComponent;
var DP, DDP: tDrawPoint;
    DC, DDC: tDrawComponent;
    i, j: Integer;
    D1, DD: Double;
begin
  if Assigned(Drawing.Mouse.GripHolder) then
    DP:= tDrawPoint(tGrip(Drawing.Mouse.GripHolder).DrawPoint);
  Result:= nil;
  if Assigned(DP.Component) then
  begin
    for i:= 0 to Count-1 do
      if (tDrawComponent(Items[i]).PointList.IndexOf(DP) <> -1) then
      begin
        Result:= tDrawComponent(Items[i]);
        Break;
      end;
  end else
  begin
    DD:= MaxInt;
    for i:= 0 to Count- 1 do
    begin
      DC:= tDrawComponent(Items[i]);
      for j:= 0 to DC.PointList.Count- 1 do
      begin
        DP:= tDrawPoint(DC.PointList.Items[j]);
        D1:= Dist(Drawing.Mouse.Pos, Drawing.Adjust(DP.Value));
        if D1 < DD then
        begin
          DD:= D1;
          DDP:= DP;
```

```
          DDC:= DC;
        end;
      end;
    end;
    for i:= 0 to Count- 1 do
      if tGrip(Drawing.GripList.Items[i]).DrawPoint = DDP then
      begin
        Drawing.Mouse.GripHolder:= DDP;
        Break;
      end;
    Result:= DDC;
  end;
end;

//public

constructor tSelectList.Create;
begin
  fVirtualComp:= tDrawComponent.Create(nil);
end;

destructor tSelectList.Destroy;
begin
  fVirtualComp.Free;
  inherited;
end;

procedure tSelectList.AddBy(aComponent: tComponent; ShiftPressed: Boolean);
begin
  if not Assigned(aComponent) then Exit;
  if isAlready(aComponent) then Exit;
  if not ShiftPressed then Clear;
  Add(aComponent);
  tDrawComponent(aComponent).PutGrips;
  if (Assigned(PropPanel) and PropPanel.Visible) then PropPanel.SetCurrent(GetSelection);
  Drawing.UndoList.AddBy(tPropCommand.CreateBy(aComponent.Owner, aComponent));
end;

procedure tSelectList.AddByFrame(aRect: tRect);
var Crossing: Boolean;
    Temp: tRect;
    HR: QRegionH;
    i: Integer;
begin
  Temp:= aRect;
  Crossing:= False;
  if Temp.Left > Temp.Right then
  begin
    Crossing:= True;
    aRect.Left:= Temp.Right;
    aRect.Right:= Temp.Left;
  end;
  if Temp.Top < Temp.Bottom then
  begin
    aRect.Top:= Temp.Bottom;
    aRect.Bottom:= Temp.Top;
  end;
  HR:= nil;
  try
    //HR:= CreateRectRgnIndirect(aRect);
    HR:=QRegion_create(@aRect, QRegionRegionType_Rectangle);
```

```
    for i:= Drawing.ComponentCount- 1 downto 0 do                    //perform a
select on all objects
      tDrawComponent(Drawing.Components[i]).OnSelectingRect(HR, Crossing);
    Drawing.TreeSynchronize;                                         //update
    Drawing.GripList.Synchronize;                                    //update
  finally
    //DeleteObject(HR);
    QRegion_Destroy(HR);
  end;
  PushToUndo;
end;

procedure tSelectList.ChangeValue(aName: String);
var i: Integer;
begin
  if Count> 1 then
    for i:= 0 to Count- 1 do
      tCustomProp(fVirtualComp.PropList.Find(aName)).CopyTo(tCustomProp(tBasic(Items[i]).
PropList.Find(aName)));
end;

procedure tSelectList.Clear;
begin
  Drawing.GripList.Clear;
  inherited;
  if (Assigned(PropPanel) and PropPanel.Visible) then
    PropPanel.SetCurrent(GetSelection);
end;

procedure tSelectList.Clone;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawComponent(Items[i]).Clone(tDrawComponent(Items[i]).Owner);
end;

procedure tSelectList.CopyToClipboard;
var i: Integer;
//    PC: pChar;
begin
  Drawing.DrawingFile.Clear;
  Drawing.DrawingFile.Index:= 0;
  Drawing.DrawingFile.Add('');
  for i:= 0 to Count- 1 do if tBasic(Items[i])is tDrawComponent then
    tDrawComponent(Items[i]).WriteXml(0);
  //PC:= AllocMem(Length(Drawing.DrawingFile.Text)+ 1);
  //StrPCopy(PC, Drawing.DrawingFile.Text);
  Clipboard.AsText:=WideString(Drawing.DrawingFile.Text);
  //Clipboard.SetTextBuf(PC);
  //FreeMem(PC);
end;

procedure tSelectList.DeleteComponent;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    Drawing.DeleteObject(tDrawComponent(Items[i]));
  Clear;
  Drawing.RefreshMouse;
end;
```

```
function tSelectList.GetSelection: tComponent;
begin
  case Count of
    0: Result:= Drawing;
    1: Result:= Items[0];
  else
    Result:= VirtualComp;
  end;
end;

procedure tSelectList.Mirror;
var i: Integer;
begin
  for i:= 0 to Count- 1 do
    tDrawComponent(Items[i]).Mirror(Point(Round(Drawing.RubberLine.First.X),
                                          Round(Drawing.RubberLine.First.Y)),
                                    Point(Round(Drawing.RubberLine.Second.X),
                                          Round(Drawing.RubberLine.Second.Y)));
  Drawing.GripList.Clear;
  Drawing.GripList.Paint(Drawing.Canvas);
end;

procedure tSelectList.MoveTo(X, Y, Z: Integer);
var i: Integer;
begin
  for i:= 0 to Count- 1 do if tBasic(Items[i])is tDrawComponent then
    tDrawComponent(Items[i]).MoveTo(X, Y, Z);
end;

procedure tSelectList.PasteFromClipboard;
procedure Clip(var aStr: String);
begin while aStr[1]=' ' do System.Delete(aStr, 1, 1); end;
var //PC: pChar;
    //i1, i2: Integer;
    //DC: tBasic;
    S: String;
    AW: tAwkLine;
begin
  //i1:= 512000;
  //PC:= AllocMem(i1+ 1);
  //i2:= Clipboard.GetTextBuf(PC, i1);
  //if i2> (i1-2) then Exit;
  //Drawing.DrawingFile.Text:= String(PC);
  Drawing.DrawingFile.Text:= String(Clipboard.AsText);
  //FreeMem(PC);
  Drawing.DrawingFile.Index:= 0;
  if Drawing.DrawingFile.Count < 1 then Exit;
  while Drawing.DrawingFile.Index< Drawing.DrawingFile.Count - 1 do
  begin
    S:= Drawing.DrawingFile.ReadLn;
    if S='' then Break;
    Clip(S);
    AW:= tAwkLine.Create(S, ' ');
    if (Pos('<', AW.Strings[0])<> 0)and
       (Pos('/', AW.Strings[0])= 0) then
    with DrawCompItemList.GetClassType(AW.Strings[0]).CreateBy(Drawing) do       //creating
a child object
    begin
      ReadXml(S);
      if PointList.Count>0 then PointList.Update;
```

```
        PointList.SetPointType;
      end;
      AW.Free;
    end;
    Drawing.Invalidate;
  end;

  procedure tSelectList.PushToUndo;
  var i: Integer;
  begin
    for i:= 0 to Count- 1 do if tBasic(Items[i])is tDrawComponent then
      Drawing.UndoList.AddBy(tPropCommand.CreateBy(tDrawComponent(Items[i]).Owner,
  tDrawComponent(Items[i])));
  end;

  procedure tSelectList.RotateByRubberLine;
  var i: Integer;
  begin
    for i:= 0 to Count- 1 do if tBasic(Items[i])is tDrawComponent then
      tDrawComponent(Items[i]).RotateByRubberLine;
  end;

  procedure tSelectList.Rotate3dByRubberLine;
  var A1, A2: Double;
      i: Integer;
      MM: tMatrix;
      FP, LP, SP: tPoint;
  begin
    FP:= Point(Round(Drawing.RubberLine.First.X), Round(Drawing.RubberLine.First.Y));
    LP:= Point(Round(Drawing.RubberLine.Last.X), Round(Drawing.RubberLine.Last.Y));
    SP:= Point(Round(Drawing.RubberLine.Second.X), Round(Drawing.RubberLine.Second.Y));
    A1:= Ang(FP, LP);
    A2:= Ang(FP, SP);
    case Drawing.Command.Status of
      csObjectRotate3dX: MM:= CreateMatrix(A2- A1, 0, 0);
      csObjectRotate3dY: MM:= CreateMatrix(0, A2- A1, 0);
    end;
    for i:= 0 to Count- 1 do
      tDrawComponent(Items[i]).Rotate3D(MM);
  end;

  procedure tSelectList.ScaleByRubberLine;
  var i: Integer;
  begin
    for i:= 0 to Count- 1 do if tBasic(Items[i])is tDrawComponent then
      tDrawComponent(Items[i]).ScaleByRubberLine;
  end;

  //published

  procedure tSelectList.GripCopy(Sender: TObject);
  begin
    Clone;
    Drawing.Command.Status:= csObjectMove;
  end;

  procedure tSelectList.GripMirror(Sender: TObject);
  var DC: tDrawComponent;
  begin
    DC:= GetElement;
    if not Assigned(DC) then Exit;
    DC.SetRubberLine;
```

```
  Mirror;
  Drawing.Command.Status:= csObjectRotate;
end;

procedure tSelectList.GripMove(Sender: TObject);
begin
  Drawing.Command.Status:= csObjectMove;
end;

procedure tSelectList.GripRotate(Sender: TObject);
var DC: tDrawComponent;
begin
  DC:= GetElement;
  if not Assigned(DC) then Exit;
  Drawing.Command.Status:= csObjectRotate;
  DC.SetRubberLine;
end;

procedure tSelectList.GripRotate3D_X(Sender: TObject);
var DC: tDrawComponent;
begin
  DC:= GetElement;
  if not Assigned(DC) then Exit;
  Drawing.Command.Status:= csObjectRotate3dX;
  DC.SetRubberLine;
end;

procedure tSelectList.GripRotate3D_Y(Sender: TObject);
var DC: tDrawComponent;
begin
  DC:= GetElement;
  if not Assigned(DC) then Exit;
  Drawing.Command.Status:= csObjectRotate3dY;
  DC.SetRubberLine;
end;

procedure tSelectList.GripScale(Sender: TObject);
var DC: tDrawComponent;
begin
  DC:= GetElement;
  if not Assigned(DC) then Exit;
  Drawing.Command.Status:= csObjectScale;
  DC.SetRubberLine;
end;

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CadTree;

interface

uses
  QT, SysUtils, Variants, Classes, QGraphics, QControls, QForms, QMenus,
  QDialogs, QComCtrls;

type
  TTreeDialog = class(TForm)
    TreeView1: TTreeView;
    procedure FormShow(Sender: TObject);
    procedure TreeView1DragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure TreeView1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure TreeView1Edited(Sender: TObject; Node: TTreeNode;
      var S: String);
    procedure TreeView1KeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure TreeView1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
    MenuItem: tMenuItem;
    procedure Refresh;
  end;

var
  TreeDialog: TTreeDialog;

implementation

{$R *.xfm}

uses //Registry,
     CadBasic, CommonCAD;

procedure TTreeDialog.FormShow(Sender: TObject);
begin
  if not Assigned(Drawing.Tree) then Drawing.Tree:= TreeView1;
  Refresh;
  MenuItem.Checked:= True;
{  try
    with tRegistry.Create do
    begin
      OpenKey('\Software\'+ Application.Title+ '\TreeView', False);              //
HKEY_CURRENT_USER
      Left:= ReadInteger('Left');
      Top:= ReadInteger('Top');
```

```
      Width:= ReadInteger('Width');
      Height:= ReadInteger('Height');
      Free;
    end;
  except
    ShowMessage('The showing of the first time will generate a compiler exeption.');
  end;
}
end;

procedure TTreeDialog.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  MenuItem.Checked:= False;
{
  try
    with tRegistry.Create do
    begin
      OpenKey('\Software\'+ Application.Title+ '\TreeView', True);
      WriteInteger('Left', Left);
      WriteInteger('Top', Top);
      WriteInteger('Width', Width);
      WriteInteger('Height', Height);
      Free;
    end;
  except
  end;
}
end;

procedure TTreeDialog.TreeView1DragDrop(Sender, Source: TObject; X,
  Y: Integer);
//var
//  AnItem: TTreeNode;
//  AttachMode: TNodeAttachMode;
//  HT: THitTests;
//  AI: Integer;
begin
{
  if TreeView1.Selected = nil then Exit;
  HT := TreeView1.GetHitTestInfoAt(X, Y);
  AnItem := TreeView1.GetNodeAt(X, Y);
  if (htOnItem in HT) or (htOnIcon in HT) then AttachMode := naAddChild
                                          else  AttachMode := naAdd;
  if htOnIndent in HT then AttachMode := naInsert;
  TreeView1.Selected.MoveTo(AnItem, AttachMode);
  if Assigned(TreeView1.Selected.Parent) then
  begin
    TreeView1.Selected.MoveTo(AnItem, AttachMode);
    case AttachMode of
      naAdd, naAddChild: tBasic(TreeView1.Selected.Data).ChangeOwner(TreeView1.Selected.
Parent.Data);
      naInsert:
        begin
          AI:= TreeView1.Selected.AbsoluteIndex;
          tBasic(TreeView1.Selected.Data).ChangePosition(TreeView1.Items.Item[AI- 1].Data);
        end;
    end;
  end else
  begin
    TreeView1.Selected.MoveTo(Drawing.Node, naAddChild);
    tBasic(TreeView1.Selected.Data).ChangeOwner(Drawing);
  end;
```

```
    Drawing.Invalidate;
  }
end;

procedure TTreeDialog.TreeView1DragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  Accept:= Source is tTreeView;
end;

procedure TTreeDialog.TreeView1Edited(Sender: TObject;
  Node: TTreeNode; var S: String);
begin
  tBasic(Node.Data).Caption:= S;
end;

procedure TTreeDialog.TreeView1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Key= Key_Delete) then Drawing.SelectList.DeleteComponent;
end;

procedure TTreeDialog.TreeView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var Node: tTreeNode;
    DC: tComponent;
    NT: String;
begin
  Node:= TreeView1.GetNodeAt(X, Y);
  if not Assigned(Node) then Exit;
  DC:= Node.Data;
  NT:= Node.Text;
  if DC is tDrawComponent
    then Drawing.SelectList.AddBy(DC, ssShift in Shift)
    else Drawing.SelectList.Clear;
  Drawing.Invalidate;
  if ssRight in Shift then
  begin
    Application.ProcessMessages;
    Drawing.PopupInspector.Popup;
  end;
end;

procedure TTreeDialog.Refresh;
var i: Integer;
begin
  TreeView1.Items.Clear;
  Drawing.Node:= TreeView1.Items.AddObject(nil, Drawing.Caption, Drawing);
  for i:= 0 to Drawing.ComponentCount- 1 do
    tBasic(Drawing.Components[i]).AddTree(TreeView1, Drawing.Node);
  TreeView1.SetFocus;
  Drawing.TreeSynchronize;
end;

initialization

  TreeDialog:= tTreeDialog.Create(Application.MainForm);

end.
```

```
{*******************************************************}
{                                                       }
{               CommonCAD Component Library             }
{               www.sitkei.com/commoncad                }
{               pal@sitkei.com                          }
{                                                       }
{*******************************************************}

unit CadUndo;

interface

uses Classes, CadProperty;

type
  tRedoList = class(tList)
  private
  public
    destructor Destroy; override;
    procedure AddBy(Item: Pointer);
    procedure ExecOne; virtual;
  end;

  tUndoList = class(tRedoList)
  private
  public
    procedure ExecOne; override;
  end;

  tCustomCommand = class
  private
    fOwner, fComp: tComponent;
  public
    constructor CreateBy(aOwner, aComp: tComponent); virtual;
    procedure Redo; virtual; abstract;
    procedure Undo; virtual; abstract;
  end;

  tCreateCommand = class(tCustomCommand)
  private
  public
    procedure Redo; override;
    procedure Undo; override;
```

```
    end;

    tDeleteCommand = class(tCustomCommand)
    private
    public
      constructor CreateBy(aOwner, aComp: tComponent); override;
      procedure Redo; override;
      procedure Undo; override;
    end;

    tPropCommand = class(tCustomCommand)
    private
      fText: String;
    public
      constructor CreateBy(aOwner, aComp: tComponent); override;
      procedure Redo; override;
      procedure Undo; override;
    end;

var Depth: Integer;

implementation

uses CadBasic, CommonCAD;

procedure tRedoList.AddBy(Item: Pointer);
begin
  if Add(Item) > Depth then
  begin
    tCustomCommand(Items[0]).Free;
    Delete(0);
  end;
end;

procedure tRedoList.ExecOne;
begin
  if Count< 1 then Exit;
  tCustomCommand(Items[Count- 1]).Redo;
  Drawing.UndoList.Add(Items[Count- 1]);
  Delete(Count- 1);
end;

destructor tRedoList.Destroy;
var i: Integer;
```

```
begin
  for i:= Count- 1 downto 0 do
    tObject(Items[i]).Free;
  inherited;
end;

procedure tUndoList.ExecOne;
begin
  if Count< 1 then Exit;
  Drawing.GripList.Clear;
  tCustomCommand(Items[Count- 1]).Undo;
  Drawing.RedoList.Add(Items[Count- 1]);
  Delete(Count- 1);
end;

constructor tCustomCommand.CreateBy(aOwner, aComp: tComponent);
begin
  fOwner:= aOwner;
  fComp:= aComp;
end;

procedure tCreateCommand.Redo;
begin
  fOwner.InsertComponent(fComp);
  Drawing.Invalidate;
end;

procedure tCreateCommand.Undo;
begin
  Drawing.SelectList.Clear;
  fOwner.RemoveComponent(fComp);
  Drawing.Invalidate;
end;

constructor tDeleteCommand.CreateBy(aOwner, aComp: tComponent);
begin
  inherited;
  fOwner.RemoveComponent(fComp);
end;

procedure tDeleteCommand.Redo;
begin
  fOwner.RemoveComponent(fComp);
  Drawing.Invalidate;
```

```
end;

procedure tDeleteCommand.Undo;
begin
  fOwner.InsertComponent(fComp);
  Drawing.Invalidate;
end;

constructor tPropCommand.CreateBy(aOwner, aComp: tComponent);
begin
  inherited;
  fText:= tDrawComponent(aComp).PropList.GetText;
end;

procedure tPropCommand.Redo;
begin
  Undo;
end;

procedure tPropCommand.Undo;
var Temp: String;
begin
  Temp:= tDrawComponent(fComp).PropList.GetText;
  tDrawComponent(fComp).PropList.SetText(fText);
  Drawing.Invalidate;
  fText:= Temp;
end;

initialization

  Depth:= 1000;

end.
```

```
{*******************************************************}
{                                                       }
{              CommonCAD Component Library              }
{              www.sitkei.com/commoncad                 }
{              pal@sitkei.com                           }
{                                                       }
{*******************************************************}

unit CommonCAD;

interface

uses Classes, QGraphics, Types, QForms, QControls, QT, QComCtrls,
     CadBasic, CadGrip, CadLayer, CadObjLink, CadOsnap, CadProperty, CadRubber,
     CadMisc, CadSelect, CadTree, CadUndo;

type
  tCommandStatus = (csAddPoint,                                          //
continuing polyline
                    csCreate,                                            //rectangle
                    csCreateAddPoint,                                    //polyline
                    csGripMove,
                    csMultiSelect,                                       //rubberbox
                    csNone,
                    csObjectMirror,
                    csObjectMove,
                    csObjectRotate,
                    csObjectRotate3dX,
                    csObjectRotate3dY,
                    csObjectScale,
                    csSecondPoint);                                      //finishing
rectangle

  tCommand = record
    Element: tDrawComponent;
    ObjectType: tDrawingClass;
    Status: tCommandStatus;
    vFloat: Double;
    vFloatLive: Boolean;
  end;

  tMouseData = record
    ActionPoint, Pos, Shift, ConPoints: tPoint;                          //conpoints
are 2 indices
    GripHolder: tObject;                                                 //grip
    Holder: tDrawComponent;
    Connected: tDrawComponent;
  end;

  tZoom = record
    Factor: Double;
  end;

  tAnalisis = record
    n, s, k, g: Integer;
    freq, timestep, endtime :Double;
    Tipo, NetList: Boolean;
    Puertos : Array of Integer;
  end;
```

```
  tDraw = class(tBasic)
  private
    procedure DoMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
    procedure DoMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
    procedure DoMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
    procedure DoMouseWheel(Sender: TObject; Shift: TShiftState; WheelDelta: Integer;
MousePos: TPoint; var Handled: Boolean);
    procedure DoPaint(Sender: TObject);
    procedure IsConnected;
    procedure MouseMoveEmpty;
    procedure MouseMoveLeft;
    procedure MouseMoveMiddle;
    procedure SetStatusText(aValue: String);
    procedure ZoomWheel(aWheelDelta: Integer; aMousePos: tPoint);
  public
    Canvas: tCanvas;
    Changed: Boolean;
    Command: tCommand;
    CombinationList: tCombinationList;                                    //layer
management
    DialogLayers: tLayersDialog;
    DialogTree: tTreeDialog;
    DrawCompItemList: tDrawCompITemList;                                  //this is
the CadBasic.DrawCompItemList
    DrawingFile: tDrawingFile;
    DrawForm: tForm;
    GripList: tGripList;
    LinkList: tLinkList;
    Mouse: tMouseData;
    OsnapShape: tOsnapShape;
    PopupInspector: tPopupMenu;
    RedoList: tRedoList;
    RubberBox: tRubberBox;
    RubberLine: tRubberLine;
    SelectList: tSelectList;
    StatusPanel: tStatusPanel;
    Tree: tTreeView;
    UndoList: tUndoList;
    Zoom: tZoom;
    constructor Create(aOwner: tComponent); override;
    destructor Destroy; override;
    function Adjust(aPoint: t3dPoint): tPoint;                            //world
point -> screen point
    function CreateObject(aPoint: tPoint): tDrawComponent;
    procedure DeleteObject(aDrawComponent: tDrawComponent);
    procedure FinishCommand;
    procedure GripMove;
    procedure Invalidate;
    function Inverse(aPoint: tPoint): tPoint;                             //screen
point -> world point
    function Inverse3D(aPoint: tPoint): t3DPoint;                         //screen
point -> world point
    procedure LoadFromFile(aFileName: String);
    procedure NameChanged; override;
    function OnConnect(aPoint: tPoint): tDrawComponent; override;
    procedure Paint(aCanvas :tCanvas); override;
    procedure WriteCIR(aIndent: Integer); override;
    procedure RefreshMouse;
    procedure SaveToFile(aFileName: String);
```

```
    procedure SaveNetList;
    procedure SetPropList; override;
    procedure TreeSynchronize;
    procedure DoKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    property StatusText: String write SetStatusText;
  published
    procedure MakeSomeObjects(Sender: TObject);
    procedure Redo;
    procedure Undo;
    procedure CopyToClipboard;
    procedure Delete;
    procedure PasteFromClipboard;
  end;

var
  Drawing: tDraw;
  Analisis: tAnalisis;
  Puertos : string;
  //Array[0..6] of Integer=(0,3,5,1,0,0,0);

const
  crPanning  = 500;
  crRotating = 501;
  crZooming  = 502;
  crOnElement= 503;
  crHand     = 504;
  crPaint    = 505;
  crCross    = 506;

implementation

{$R CommonCAD.res}

uses SysUtils, CadDevices;

//tDraw

procedure tDraw.DoKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
var L, R: tPoint;
begin
  case Key of
    Key_Delete: SelectList.DeleteComponent;
    Key_R:
          if (Drawing.SelectList.Count> 0) and (tBasic(Drawing.SelectList.Items[0]) is
tDevice) then
          With tDevice(Drawing.SelectList.Items[0]) do
          begin
            L:= Drawing.Adjust(tDrawPoint(PointList.Items[0]).GetPoint);

            if (Algn90=1) then Algn360:=Algn360*-1;
            Algn90:=Abs(Algn90-1);

            if Algn90=0 Then R.X:= L.X + Round(126*Zoom) Else R.X:= L.X + Round(96*Zoom);
            if Algn90=0 Then R.Y:= L.Y + Round(96*Zoom) Else R.Y:= L.Y + Round(126*Zoom);
            Command.Element.SizeTo(Inverse(R));
            PointList.Update;

            Drawing.Invalidate;
            Key:=0;
          end;
  end;
end;
```

```
procedure tDraw.DoMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
var PP: tPoint;
begin
  if Drawing.PopupInspector.Visible then
        Drawing.PopupInspector.FormDeactivate(Self);
  case Command.Status of
    csCreate:
      begin
        SetSTatusText('Click for the other point');
        Command.Element:= CreateObject(Point(Mouse.Pos.X, Mouse.Pos.Y));
        Invalidate;
      end;
    csCreateAddPoint:
      begin
        SetSTatusText('Click for the next point or right click for the finishing');
        Command.Element:= CreateObject(Point(Mouse.Pos.X, Mouse.Pos.Y));
        Invalidate;
      end;
    csAddPoint:
      if mbRight=Button then
      begin
        Command.Element.PointList.RemoveLast;
        FinishCommand;
        Exit;
      end else
      begin
        SetSTatusText('Click for the next point or right click for the finishing');
        Command.Element.PointList.Add(tDrawPoint(Command.Element.PointList.
          Items[Command.Element.PointList.Count- 1]).Clone(Command.Element));
        Invalidate;
      end;
    csObjectMirror, csObjectRotate, csObjectRotate3dX, csObjectRotate3dY, csObjectScale:
      RubberLine.Hide;
    csNone:
      if Assigned(Mouse.GripHolder) then
      begin
        Command.Status:= csGripMove;
      end else
      if Assigned(Mouse.Holder) then
      begin
        SelectList.AddBy(Mouse.Holder, ssShift in Shift);
        Invalidate;
        TreeSynchronize;
        Command.Status:= csObjectMove;
        Command.Element:= Mouse.Holder;
      end else
      begin
        SelectList.Clear;
        Invalidate;
        TreeSynchronize;
        RubberBox.Start(Point(X, Y));
        Command.Status:= csMultiSelect;
      end;
  end;
  Mouse.ActionPoint:= Mouse.Pos;
  if mbRight=Button then
  begin
    Command.Status:= csNone;
    PP:= QForms.Mouse.CursorPos;                                    //get cursor
position
```

```
    PopupInspector.Popup;                                          //popup the
inspector
    PP:= Point(PP.X- 1, PP.Y- 1);                                  //shift
position for 1
    QForms.Mouse.CursorPos:= PP;                                  //shift
cursor to the new position
  end;
end;

procedure tDraw.DoMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  Mouse.Pos:= Point(X, Y);
  if ssLeft in Shift then MouseMoveLeft else
  if ssMiddle in Shift then MouseMoveMiddle else
    MouseMoveEmpty;
end;

procedure tDraw.DoMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  case Command.Status of
    csSecondPoint:
    begin
      FinishCommand;
      if Assigned(Command.Element) then Command.Element.AddTree(Tree, Self.Node);
    end;
    csCreate:
      Command.Status:= csSecondPoint;
    csCreateAddPoint, csAddpoint:
      Command.Status:= csAddPoint;
    csMultiSelect:
    begin
      SelectList.AddByFrame(RubberBox.Hide);
      Command.Status:= csNone;
    end;
    else Command.Status:= csNone;
  end;
  Invalidate;
  Changed:= True;
  OsnapShape.Visible:= False;
  Command.vFloatLive:= False;
end;

procedure tDraw.DoMouseWheel(Sender: TObject; Shift: TShiftState;
  WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean);
begin
  ZoomWheel(WheelDelta, DrawForm.ScreenToClient(MousePos));
  Handled:= True;
  Changed:= True;
end;

procedure tDraw.DoPaint(Sender: TObject);
begin
  Paint(Canvas);
end;

procedure tDraw.IsConnected;
var
   NodeValue: string;
   NodeName: string;
   AW1, AW2: tAwkLine;
   i, j, y : integer;
```

```
begin
  Mouse.Connected:= tDrawComponent(OnConnect(Mouse.Pos));
  if Assigned(Mouse.Connected) then
  begin
    if Mouse.Connected.ClassParent.ClassNameIs('tDevice') then
      if Assigned(Command.Element) then
        if Command.Element.ClassNameIs('tPolyline') then
    begin
         NodeValue := tDevice(Mouse.Connected).NodeList.WriteToString;
         //NodeName:=FloatToStr(tDrawPoint(Command.Element.NodeList.Items[0]).Value.X);
         NodeName := IntToStr(Drawing.ChildNames.GetNumber(Command.Element.Caption));
         AW2:= tAwkLine.Create(NodeValue, '/');
         y := 0;
         NodeValue:='';
         for j:= 0 to AW2.Count- 1 do
         begin
              AW1:= tAwkLine.Create(AW2[j], '*');
              for i:= 0 to AW1.Count- 1 do
              begin
                if (AW1[i]='-1') and (NodeName<>'-1') and (y<>1) then
                   AW1[i]:=NodeName;
                if (AW1[i]=NodeName) then
                   y:=1;
              end;
              if (AW2.Count>1) and (j>0) then
                   NodeValue:= NodeValue + '/';
              NodeValue := NodeValue + AW1[0]+'*'+AW1[1]+'*'+AW1[2];
              AW1.Free;
         end;
         AW2.Free;
         tDevice(Mouse.Connected).NodeList.Clear;
         tDevice(Mouse.Connected).NodeList.ReadFromString(NodeValue);
    end;
    OsnapShape.Visible:= True;
    Mouse.Pos:= Adjust(OsnapShape.OsnapPoint);
  end else
    OsnapShape.Visible:= False;
end;

procedure tDraw.MouseMoveEmpty;
begin
  case Command.Status of
  csNone:
    begin
      Mouse.GripHolder:= GripList.OnItem(Mouse.Pos);
      if Assigned(Mouse.GripHolder) then
      begin
        Screen.Cursor:= crCross;
        StatusText:= 'Press right button for the available functions';
      end else
      begin
        Screen.Cursor:= crDefault;
        StatusText:= '';
      end;
      if Assigned(Mouse.GripHolder) then Exit;
      Mouse.Holder:= tDrawComponent(OnItem(Mouse.Pos));
      if Assigned(Mouse.Holder)then
      begin
        Screen.Cursor:= crOnElement;
        StatusText:= Mouse.Holder.Caption;
      end else
```

```
        begin
          Screen.Cursor:= crDefault;
          StatusText:= '';
        end;
      end;
    csSecondPoint, csAddPoint:
      if Assigned(Command.Element) then
      begin
        IsConnected;
        Command.Element.SizeTo(Inverse(Mouse.Pos));
        Invalidate;
      end;
    csCreate, csCreateAddPoint:
      begin
        IsConnected;
        StatusText:= 'Click a point to create an object';
        Screen.Cursor:= crCross;
      end;
    csObjectRotate, csObjectRotate3dX, csObjectRotate3dY, csObjectScale:
      begin
        IsConnected;
        RubberLine.ExpandTo(Mouse.Pos);
        case Command.Status of
          csObjectRotate: SelectList.RotateByRubberLine;
          csObjectRotate3dX, csObjectRotate3dY: SelectList.Rotate3dByRubberLine;
          csObjectScale: SelectList.ScaleByRubberLine;
        end;
        Invalidate;
        Mouse.ActionPoint:= Mouse.Pos;
        Changed:= True;
        Exit;
      end;
    csObjectMove:
      begin
        IsConnected;
        SelectList.MoveTo(Round((Mouse.Pos.X- Mouse.ActionPoint.X)/ Zoom.Factor),
                          Round((Mouse.Pos.Y- Mouse.ActionPoint.Y)/ Zoom.Factor), 0);
        Invalidate;
        Mouse.ActionPoint:= Mouse.Pos;
        Changed:= True;
        Exit;
      end;
    end;
  end;

procedure tDraw.MouseMoveLeft;
begin
  case Command.Status of
    csCreate, csSecondPoint, csAddPoint:
      if Assigned(Command.Element) then
      begin
        Command.Element.SizeTo(Inverse(Mouse.Pos));
        Invalidate;
      end;
    csMultiSelect:
      begin
        Mouse.ActionPoint:= Mouse.Pos;
        if RubberBox.Visible then RubberBox.ExpandTo(Mouse.Pos);
        StatusText:= '<-cross, window->';
      end;
    csGripMove:
      begin
```

```
        IsConnected;
        GripMove;
        Invalidate;
        Mouse.ActionPoint:= Mouse.Pos;
      end;
    csObjectMove:
      begin
        SelectList.MoveTo(Round((Mouse.Pos.X- Mouse.ActionPoint.X)/ Zoom.Factor),
                          Round((Mouse.Pos.Y- Mouse.ActionPoint.Y)/ Zoom.Factor), 0);
        Invalidate;
        Mouse.ActionPoint:= Mouse.Pos;
      end;
    end;
  end;

  procedure tDraw.MouseMoveMiddle;
  begin
    Mouse.Shift.X:= Mouse.Shift.X- (Mouse.Pos.X- Mouse.ActionPoint.X);
    Mouse.Shift.Y:= Mouse.Shift.Y- (Mouse.Pos.Y- Mouse.ActionPoint.Y);
    Invalidate;
    Mouse.ActionPoint:= Mouse.Pos;
  end;

  procedure tDraw.SetStatusText(aValue: String);
  begin
    if Assigned(StatusPanel) then
      StatusPanel.Text:= aValue;
  end;

  procedure tDraw.ZoomWheel(aWheelDelta: Integer; aMousePos: tPoint);
  var ZF: Single;
  begin
    if aWheelDelta > 0
      then ZF:= 1.1
      else ZF:= 0.9;
    Zoom.Factor:= Zoom.Factor* ZF;
    Mouse.Shift.X := Mouse.Shift.X + Round((Mouse.Shift.X+ aMousePos.X)*(ZF- 1));
    Mouse.Shift.Y := Mouse.Shift.Y + Round((Mouse.Shift.Y+ aMousePos.Y)*(ZF- 1));
    Invalidate;
  end;

  constructor tDraw.Create(aOwner: tComponent);
  begin
    inherited Create(aOwner);
    DrawForm:= tForm(Owner);
    DrawForm.Color:= clWhite;
    Canvas:= DrawForm.Canvas;
    Canvas.Brush.Style:= bsClear;
    Caption:= 'Untitled';
    CombinationList:= CadLayer.CombinationList;
    Command.Status:= csNone;
    DialogLayers:= LayersDialog;
    DialogTree:= TreeDialog;
    DrawCompItemList:= CadBasic.DrawCompItemList;                    //this is
  created in the initialization time
    DrawForm.OnKeyDown:= DoKeyDown;
    DrawForm.OnMouseDown:= DoMouseDown;
    DrawForm.OnMouseMove:= DoMouseMove;
    DrawForm.OnMouseUp:= DoMouseUp;
    DrawForm.OnMouseWheel:= DoMouseWheel;
    DrawForm.OnPaint:= DoPaint;
  //  DrawForm.DoubleBuffered:= True;
```

```
  DrawingFile:= tDrawingFile.Create;
  GripList:= tGripList.Create;
  LinkList:= tLinkList.Create;
  OsnapShape:= tOsnapShape.Create;
  PopupInspector:= tPopupMenu.CreateNew(nil);
  PopupInspector.SetUp;
  PropList:= tPropList.CreateBy(Self);
  RedoList:= tRedoList.Create;
  RubberBox:= tRubberBox.CreateBy(Owner, Canvas);
  RubberLine:= tRubberLine.CreateBy(Owner, Canvas);
  SetPropList;
  SelectList:= tSelectList.Create;
  UndoList:= tUndoList.Create;
  Zoom.Factor:= 0.01;
end;

destructor tDraw.Destroy;
begin
  SelectList.Free;
  OsnapShape.Free;
  GripList.Free;
  //DrawCompItemList.Free;
end;

function tDraw.Adjust(aPoint: t3dPoint): tPoint;
begin
  Result.X:= Round(aPoint.X* Zoom.Factor)- Mouse.Shift.X;
  Result.Y:= Round(aPoint.Y* Zoom.Factor)- Mouse.Shift.Y;
end;

function tDraw.CreateObject(aPoint: tPoint): tDrawComponent;
begin
  Result:= Command.ObjectType.CreateByPoint(Self, Inverse3D(aPoint));
  Changed:= True;
end;

procedure tDraw.DeleteObject(aDrawComponent: tDrawComponent);
begin
  GripList.Clear;
  if (Assigned(Tree) and Tree.Visible) then
    Tree.Items.Delete(aDrawComponent.Node);
  if (not(aDrawComponent is tLink)) and (aDrawComponent.LinkID > -1) then
    Drawing.LinkList.DeleteLinks(aDrawComponent.LinkID);
//  aDrawComponent.Free;
  UndoList.AddBy(tDeleteCommand.CreateBy(aDrawComponent.Owner, aDrawComponent));
  Invalidate;
  Changed:= True;
end;

procedure tDraw.FinishCommand;
begin
  Command.Status:= csNone;
  Command.Element:= nil;
  OsnapShape.Visible:= False;
end;

procedure tDraw.GripMove;
begin
  tGrip(Mouse.GripHolder).DragTo(Point(Round((Mouse.Pos.X- Mouse.ActionPoint.X)/ Zoom.
Factor),
                                     Round((Mouse.Pos.Y- Mouse.ActionPoint.Y)/ Zoom.
Factor)));
```

```
end;

procedure tDraw.Invalidate;
begin
  DrawForm.Invalidate;
end;

function tDraw.Inverse(aPoint: tPoint): tPoint;
begin
  Result.X:= Round((aPoint.X+ Mouse.Shift.X)/ Zoom.Factor);
  Result.Y:= Round((aPoint.Y+ Mouse.Shift.Y)/ Zoom.Factor);
end;

function tDraw.Inverse3D(aPoint: tPoint): t3DPoint;
begin
  Result.X:= (aPoint.X+ Mouse.Shift.X)/ Zoom.Factor;
  Result.Y:= (aPoint.Y+ Mouse.Shift.Y)/ Zoom.Factor;
  Result.Z:= 0.0;
end;

procedure tDraw.LoadFromFile(aFileName: String);
begin
  GripList.Clear;                                                    //if there
is a selection
  DestroyComponents;                                                 //deleting
of the drawing elements
  Screen.Cursor:= crHourGlass;
  DrawingFile.LoadFromFile(aFileName);
  DrawingFile.Index:= 0;
  ReadXml(DrawingFile.
ReadLn);                                                        //started the
reading of the object tree
  Screen.Cursor:= crDefault;
  NameChanged;
  Invalidate;
  Changed:= False;
  if (Assigned(DialogLayers) and DialogLayers.Visible) then DialogLayers.Refresh;
  if (Assigned(DialogTree) and DialogTree.Visible) then DialogTree.Refresh;
end;

procedure tDraw.NameChanged;                                         //setting up
the treenode
begin
  inherited;
  //Application.MainForm.Caption:= 'Simulador de Circuitos de RF y Microondas - '+ Drawing.
Caption;
end;

function tDraw.OnConnect(aPoint: tPoint): tDrawComponent;          //index of second point
in the pointlist
var i: Integer;
begin
  Result:= nil;
  for i:= ComponentCount- 1 downto 0 do
  begin
    if tDrawComponent(Components[i]) = Command.Element
      then Continue;
    Result:= tDrawComponent(Components[i]).OnConnect(aPoint);
    if Assigned(Result) then Break;
  end;
end;
```

```
procedure tDraw.Paint(aCanvas: tCanvas);
var i: Integer;
    BMP :TBitmap;
    ARect: TRect;
begin
  BMP := TBitMap.Create;
  BMP.Height:=Screen.Height;
  BMP.Width :=Screen.Width;
  with BMP.Canvas do
  begin
    CopyMode := cmWhiteness;
    ARect := Rect(0, 0, BMP.Width, BMP.Height);
    CopyRect(ARect, BMP.Canvas, ARect);
    CopyMode := cmSrcCopy; { restore the copy mode }
  end;
  for i:= 0 to ComponentCount- 1 do
    tDrawComponent(Components[i]).Paint(BMP.Canvas);
  GripList.Paint(BMP.Canvas);
  DrawForm.Canvas.Draw(0,0,BMP);
  OsnapShape.Paint;
  RubberLine.Paint;
  RubberBox.Paint;
  BMP.Free;
  //inherited;
end;

procedure tDraw.RefreshMouse;
begin
  MouseMoveEmpty;
end;

procedure tDraw.SaveToFile(aFileName: String);
var CA: String;
begin
  Screen.Cursor:= crHourGlass;
  CA:= ExtractFileName(aFileName);                                    //setting
up the Caption by the FileName
  System.Delete(CA, Pos('.', CA), 255);                                    //
  Caption:= CA;
  DrawingFile.Clear;
  DrawingFile.Index:= 0;
  DrawingFile.Add('');
  WriteXml(0);
  DrawingFile.SaveToFile(aFileName);
  Screen.Cursor:= crDefault;
  Changed:= False;
end;

procedure tDraw.SaveNetList;
var aFileName, print, prac: String;
    plotcmd: TStrings;
    i : Integer;

begin
  if not Changed then
  begin
        Screen.Cursor:= crHourGlass;
        aFileName:=Caption+'.cir';
        DrawingFile.Clear;
        DrawingFile.Index:= 0;
        DrawingFile.Add('');
```

```
        WriteCir(0);
        DrawingFile.SaveToFile(ExtractFileDir(Application.ExeName)+'\circuitfiles
\'+aFileName);
        Screen.Cursor:= crDefault;

        plotcmd:=tStringList.Create;
        try
           plotcmd.Add('Archivo de control de Graficos de salida');
           plotcmd.Add('.control');
           plotcmd.Add('load circuitfiles\rawspice.raw');
           if Analisis.Tipo then
              plotcmd.Add('setplot tran1');
           print := 'plot';
           prac  := 'plot';
           for i:=0 to Analisis.n do
           begin
               if (Analisis.Puertos[i]<>0) then print:= print + ' V(' + IntToStr(Analisis.
Puertos[i]) + ')';
               if (Analisis.Puertos[i]<>0) then prac := prac  + ' vm(' + IntToStr(Analisis.
Puertos[i]) + ')';
           end;
           plotcmd.Add(print);
           if Analisis.Tipo then
           begin
              plotcmd.Add('setplot ac1');
              plotcmd.Add(prac);
           end;

           plotcmd.SaveToFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\plot.cir');
        finally
           plotcmd.Free;
        end;
  end
  else
  begin
        Application.MessageBox(''''+ Caption + ''' has not be saved first',
                'Critical', [smbOK], smsCritical);
  end;
end;

procedure tDraw.SetPropList;
begin
  inherited;
  LinkID:= -1;
  with PropList do
  begin
    Add(tMenuProp.CreateProp(MakeSomeObjects, 'Make objects'));
    Add(tFloatProp.CreateProp(@Zoom.Factor, 'Zoomfactor'));
    Add(tIntegerProp.CreateProp(@Mouse.Shift.X, 'ShiftX'));
    Add(tIntegerProp.CreateProp(@Mouse.Shift.Y, 'ShiftY'));
    Add(tCombinationListProp.CreateProp(@CombinationList, 'Combinations'));     //for the
read/write only

    Add(tIntegerProp.CreateProp(@Analisis.n, 'no_of_nodes'));
    Add(tIntegerProp.CreateProp(@Analisis.s, 'no_of_srces'));
    Add(tIntegerProp.CreateProp(@Analisis.k, 'no_of_freqs'));
    Add(tFloatProp.CreateProp(@Analisis.freq, 'frequency'));
    Add(tFloatProp.CreateProp(@Analisis.timestep, 'timestep'));
    Add(tFloatProp.CreateProp(@Analisis.endtime, 'endtime'));
    Add(tBooleanProp.CreateProp(@Analisis.Tipo, 'HB'));
    Add(tBooleanProp.CreateProp(@Analisis.NetList, 'NetList'));
    Add(tStringProp.CreateProp(@CommonCad.Puertos, 'Puertos'));
```

```
    end;
  end;

  procedure tDraw.TreeSynchronize;
  var i: Integer;
  begin
    if ((not Assigned(Tree))or (not Tree.Visible)) then Exit;
    if SelectList.Count = 0
      then Tree.Selected:=Node
      else
        for i:= 0 to SelectList.Count- 1 do
          Tree.Selected:=tBasic(SelectList.Items[i]).Node;
  end;

  procedure tDraw.MakeSomeObjects(Sender: TObject);
  var i: Integer;
      CC: tDrawingClass;
      OO: tDrawComponent;
  begin
    Randomize;
    for i:= 0 to 11 do
    begin
      case Random(2) of
        0: CC:= DrawCompItemList.GetClassType('Ellipse');
        else CC:= DrawCompItemList.GetClassType('Rectangle');
      end;
      OO:= CC.CreateByPoint(Self, Make3dPoint(Random(DrawForm.Width* 70), Random(DrawForm.
  Height* 70), 0));
      case Random(2) of
        0: OO.Filled:= True;
      end;
      tDrawPoint(OO.PointList.Items[1]).MoveTo(120*100, 0, 0);
      tDrawPoint(OO.PointList.Items[2]).MoveTo(120*100, 70*100, 0);
      tDrawPoint(OO.PointList.Items[3]).MoveTo(0, 70*100, 0);
      OO.AddTree(Tree, Node);
    end;
    FinishCommand;
    Invalidate;
  end;

  procedure tDraw.Redo;
  begin
    RedoList.ExecOne;
  end;

  procedure tDraw.Undo;
  begin
    UndoList.ExecOne;
  end;

  procedure tDraw.CopyToClipboard;
  begin
    SelectList.CopyToClipboard;
  end;

  procedure tDraw.Delete;
  begin
    SelectList.DeleteComponent;
  end;

  procedure tDraw.PasteFromClipboard;
  begin
```

```
  SelectList.PasteFromClipboard;
end;

procedure tDraw.WriteCIR(aIndent: Integer);
var
   i : integer;
   Models, Devices: TStrings;
   print : string;

begin
  Models:= TStringList.Create;
  Devices:= TStringList.Create;

  try

  Drawing.DrawingFile.WriteLn(Indent(aIndent)+ Caption);
  if Analisis.Tipo then
     Drawing.DrawingFile.WriteLn(Indent(aIndent)+ '.ac lin 3 '+ FloatToStr(Analisis.freq) +
' ' + FloatToStr(Analisis.freq))
  else
     Drawing.DrawingFile.WriteLn(Indent(aIndent)+ '.tran '+ FloatToStr(Analisis.timestep) +
' ' + FloatToStr(Analisis.endtime));

  Drawing.DrawingFile.WriteLn(Indent(aIndent));
  if ComponentCount = 0 then
    Exit;

  Analisis.g:=0;

  for i:= 0 to ComponentCount- 1 do
  begin
    tBasic(Components[i]).WriteCIR(aIndent);
    if Components[i].ClassNameIs('tDiode') OR
       Components[i].ClassNameIs('tBJT') OR
       Components[i].ClassNameIs('tBSource') OR
//     Components[i].ClassNameIs('tDSource') OR
       Components[i].ClassNameIs('tAnground') then
       begin
          Dec(Drawing.DrawingFile.Index);
          Devices.Add(Drawing.DrawingFile.Readln);
          if Components[i].ClassNameIs('tAnground') then
             Analisis.g:=Analisis.g+1
          else
             if Analisis.Tipo then
             begin
                Drawing.DrawingFile.Delete(Drawing.DrawingFile.Index);
                Dec(Drawing.DrawingFile.Index);
                Drawing.DrawingFile.Delete(Drawing.DrawingFile.Index);
                Drawing.DrawingFile.Add('');
             end;
       end;
  end;
  Devices.SaveToFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\devices.cir');

  Drawing.DrawingFile.WriteLn(Indent(aIndent));

  Models.LoadFromFile(ExtractFileDir(Application.ExeName)+'\circuitfiles\models.cir');

  for i:=0 to Models.Count-1 do
    Drawing.DrawingFile.WriteLn(Indent(aIndent)+ Models[i]);

  print := '.print';
```

```
    if Analisis.Tipo then
       print:=print + ' ac'
    else
       print:=print + ' tran';

    for i:=0 to Length(Analisis.Puertos)-1 do
        if (Analisis.Puertos[i]<>0) then print:= print + ' V(' + IntToStr(Analisis.Puertos
[i]) + ')';

    Drawing.DrawingFile.WriteLn(Indent(aIndent)+ print);
    Drawing.DrawingFile.WriteLn(Indent(aIndent)+ '.end');

    finally
        Models.Free;
        Devices.Free;
    end;
//}
end;

initialization

  with Screen do
  begin
    Cursors[crPanning]:=Cursors[crDefault];   //LoadCursor(hInstance, 'PAN');
    Cursors[crRotating]:=Cursors[crDefault];  //LoadCursor(hInstance, 'TWIST');
    Cursors[crZooming]:=Cursors[crDefault];   //LoadCursor(hInstance, 'ZOOM');
    Cursors[crOnElement]:=Cursors[crDefault]; //LoadCursor(hInstance, 'CROSSER');
    Cursors[crHand]:=Cursors[crDefault];      //LoadCursor(hInstance, 'HAND');
    Cursors[crPaint]:=Cursors[crDefault];     //LoadCursor(hInstance, 'PAINT');
    Cursors[crCRoss]:=Cursors[crDefault];     //LoadCursor(hInstance, 'CROSS');
                                              //QCursor_create();
   end;

end.
```

Borland Forms Source Files

/

- CLXAbout.dfm
- CLXChildWin.dfm
- CLXMain.dfm
- CLXSetup.dfm

Lib/

- CadLayer.dfm
- CadTree.dfm

```
object AboutBox: TAboutBox
  Left = 261
  Top = 231
  ActiveControl = OKButton
  AutoScroll = False
  Caption = 'Acerca de ...'
  ClientHeight = 273
  ClientWidth = 292
  Color = clBackground
  Icon.Data = {
    0000010001002020100000000000E80200001600000028000000200000004000
    0000010004000000000000800200000000000000000000000000000000000000
    0000000080000080000000808000800000008000800080800000C0C0C0008080
    80000000FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF000000
    0000000000000000000000000000000000000000000000000000000000000888
    8888888888888888888888888880000000000000000000000000000080F88
    888888888888888888888888080F7F7F7F7F7F7F7F7F7F7F7F7F78080FF7
    F7F788888888888888F7F7F7F8080F7F7F70000000000000087F7F7F78080FF7
    F7F0FFFFFFFFFFFFF08F7F7F7F8080F7F7F70FFFFFFFFFFFFF087F7F7F78080FF7
    F000FFFFFFFFFFFF08F7F7F7F8080F7F70F0FFFFFFFFFFFF087F7F7F78080FF7
    F0F0FFFFFFFFFFFF08F7F7F7F8080F7000F0CCCCCC070707087F7F7F78080FF0
    F0F000000000000007F7F7F7F8080F70F0FFFFFFFFFFFF087F7F7F7F78080FF0
    F0CCCCCC07070708F7F7F7F7F8080F70F000000000000000F7F7F7F7F78080FF0
    FFFFFFFFFFFF08F7F7F7F7F7F8080F70CCCCCC070707087F7F7F7F7F78080FF0
    00000000000007F7F7F7F7F7F8080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080F7F7F7F7F7F7F7F7F7F7F7F7F7F7F080F00
    07F000F70007F7F7F7F7F7F7F7080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080CCCCCCCCCCCCCCCCCCCCCC070707080000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000FFFF
    FFFFFFFFFFFF8000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000001FFFFFFFFFFFFFFFFFFFFFFFF}
  Position = poScreenCenter
  PixelsPerInch = 75
  TextHeight = 13
  TextWidth = 6
  object Panel1: TPanel
    Left = 8
    Top = 8
    Width = 281
    Height = 217
    BevelOuter = bvLowered
    TabOrder = 0
    object ProgramIcon: TImage
      Left = 8
      Top = 8
      Width = 32
      Height = 32
      Picture.Data = {
        055449636F6E0000010001002020100000000000E80200001600000028000000
        2000000040000000010004000000000080020000000000000000000000000000
        0000000000000000000080000080000000808000800000008000800080800000
        C0C0C0008080800000000FF0000FF000000FFFF00FF000000FF00FF00FFFF0000
        FFFFFF0000000000000000000000000000000000000000000000000000000000
        0000000008888888888888888888888888888888000000000000000000000000
        000000080F8888888888888888888888888080F7F7F7F7F7F7F7F7F7F7F7F7F7F
        7F7F78080FF7F7F788888888888888F7F7F7F8080F7F7F70000000000000087F
        7F7F78080FF7F7F0FFFFFFFFFFFFF08F7F7F7F8080F7F7F70FFFFFFFFFFFFF087F
```

```
      7F7F78080FF7F000FFFFFFFFFFFF08F7F7F7F8080F7F70F0FFFFFFFFFFFF087F
      7F7F78080FF7F0F0FFFFFFFFFFFF08F7F7F7F8080F7000F0CCCCCC070707087F
      7F7F78080FF0F0F000000000000007F7F7F7F8080F70F0FFFFFFFFFFF087F7F
      7F7F78080FF0F0CCCCCC07070708F7F7F7F7F8080F70F00000000000000F7F7F
      7F7F78080FF0FFFFFFFFFFF08F7F7F7F7F7F8080F70CCCCCC070707087F7F7F
      7F7F78080FF000000000000007F7F7F7F7F7F8080FFFFFFFFFFFFFFFFFFFFFF
      FFFFFF0800000000000000000000000000000080F7F7F7F7F7F7F7F7F7F7F7F
      7F7F7F080F0007F000F70007F7F7F7F7F7F7F7080FFFFFFFFFFFFFFFFFFFFFFF
      FFFFFF0800000000000000000000000000000080CCCCCCCCCCCCCCCCCCCCCC
      0707070800000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000
      00000000FFFFFFFFFFFFFFFF8000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000001FFFFFFFFFFFFFFF
      FFFFFFFF)
    end
    object ProductName: TLabel
      Left = 75
      Top = 8
      Width = 200
      Height = 49
      AutoSize = False
      Caption =
        'Simulador de Circuitos Electr'#243'nicos basado en el M'#233'todo del Bala' +
        'nce Arm'#243'nico'
      Font.Color = clBlack
      Font.Height = 12
      Font.Name = 'lucidabright'
      Font.Pitch = fpVariable
      Font.Style = [fsBold]
      ParentFont = False
      Layout = tlCenter
      WordWrap = True
    end
    object Version: TLabel
      Left = 75
      Top = 56
      Width = 70
      Height = 15
      Caption = 'versi'#243'n 1.0.9'
      Font.Color = clBlack
      Font.Height = 12
      Font.Name = 'fangsong ti'
      Font.Pitch = fpVariable
      Font.Style = [fsItalic]
      ParentFont = False
      Layout = tlCenter
    end
    object Copyright: TLabel
      Left = 8
      Top = 88
      Width = 200
      Height = 25
      AutoSize = False
      Caption = 'Copyright '#169' 2007. Universidad Don Bosco, El Salvador.'
      Font.Color = clBlack
      Font.Height = 10
      Font.Name = 'fangsong ti'
      Font.Pitch = fpVariable
      Font.Style = []
      Font.Weight = 40
```

```
      ParentFont = False
      Layout = tlCenter
      WordWrap = True
    end
    object Comments: TLabel
      Left = 8
      Top = 120
      Width = 249
      Height = 89
      AutoSize = False
      Caption =
        'Dise'#241'ado y desarrollado por Antonio Orantes bajo la supervisi'#243'n ' +
        'del Ingeniero Jorge L'#243'pez.                    Idea original Ing' +
        'eniero Wenceslao Rivas.           Basado en SPICE. SPICE se des' +
        'arroll'#243' en los laboratorios de investigaci'#243'n en Electr'#243'nica de l' +
        'a Universidad de California, en Berkeley '#169' 1972.'
      Font.Color = clBlack
      Font.Height = 10
      Font.Name = 'fangsong ti'
      Font.Pitch = fpVariable
      Font.Style = []
      Font.Weight = 40
      ParentFont = False
      Layout = tlCenter
      WordWrap = True
    end
  end
  object OKButton: TButton
    Left = 112
    Top = 234
    Width = 65
    Height = 33
    Caption = 'OK'
    Default = True
    ModalResult = 1
    TabOrder = 1
  end
end
```

```
object MDIChild: TMDIChild
  Left = 218
  Top = 213
  Width = 400
  Height = 282
  Caption = 'MDI Child'
  Color = clWhite
  FormStyle = fsMDIChild
  KeyPreview = True
  Position = poDefault
  Visible = True
  OnActivate = FormActivate
  OnClose = FormClose
  PixelsPerInch = 75
  TextHeight = 13
  TextWidth = 6
  object Memo1: TMemo
    Left = 0
    Top = 0
    Width = 400
    Height = 282
    Align = alClient
    Enabled = False
    Lines.Strings = (
      'Memo1')
    TabOrder = 0
    Visible = False
  end
end
```

```
object MainForm: TMainForm
  Left = 198
  Top = 116
  Width = 502
  Height = 397
  VertScrollBar.Range = 61
  HorzScrollBar.Range = 67
  ActiveControl = ToolButton1
  Anchors = [akTop, akRight, akBottom]
  Caption = 'Simulador de Circuitos de RF y Microondas'
  Color = clMid
  FormStyle = fsMDIForm
  Icon.Data = {
    0000010001002020100000000000E80200001600000028000000200000004000
    0000010004000000000008002000000000000000000000000000000000000000
    0000000080000080000000808000800000008000800080800000C0C0C0008080
    80000000FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF000000
    0000000000000000000000000000000000000000000000000000000000000888
    8888888888888888888888888800000000000000000000000000000080F88
    888888888888888888888888080F7F7F7F7F7F7F7F7F7F7F7F7F7F78080FF7
    F7F788888888888888F7F7F7F8080F7F7F70000000000000087F7F7F78080FF7
    F7F0FFFFFFFFFFFF08F7F7F7F8080F7F7F70FFFFFFFFFFFF087F7F7F78080FF7
    F000FFFFFFFFFFFF08F7F7F7F8080F7F70F0FFFFFFFFFFFF087F7F7F78080FF7
    F0F0FFFFFFFFFFFF08F7F7F7F8080F7000F0CCCCCC070707087F7F7F78080FF0
    F0F000000000000007F7F7F7F8080F70F0FFFFFFFFFFFF087F7F7F7F78080FF0
    F0CCCCCC07070708F7F7F7F7F8080F70F000000000000000F7F7F7F7F78080FF0
    FFFFFFFFFFFF08F7F7F7F7F7F8080F70CCCCCC070707087F7F7F7F7F78080FF0
    000000000000007F7F7F7F7F7F8080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080F7F7F7F7F7F7F7F7F7F7F7F7F7F7F080F00
    07F000F70007F7F7F7F7F7F7F7080FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080CCCCCCCCCCCCCCCCCCCCCCC070707080000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000FFFF
    FFFFFFFFFFFF8000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000001FFFFFFFFFFFFFFFFFFFFFFFF}
  KeyPreview = True
  Menu = MainMenu1
  Position = poDefault
  OnClose = FormClose
  OnCreate = FormCreate
  OnKeyDown = FormKeyDown
  PixelsPerInch = 96
  TextHeight = 13
  TextWidth = 6
  object StatusBar: TStatusBar
    Left = 0
    Top = 354
    Width = 502
    Height = 20
    AutoHint = True
    Panels = <
      item
      end>
  end
```

```
object ControlBar1: TControlBar
  Left = 0
  Top = 0
  Width = 502
  Height = 41
  Align = alTop
  TabOrder = 1
  OnMouseMove = ControlBar1MouseMove
  object ToolBar2: TToolBar
    Left = 15
    Top = 6
    Width = 330
    Height = 27
    AutoSize = True
    BorderWidth = 1
    ButtonHeight = 23
    Color = clButton
    EdgeBorders = []
    Images = ImageList1
    Indent = 5
    ParentColor = False
    ParentShowHint = False
    ShowHint = True
    TabOrder = 0
    Wrapable = False
    object ToolButton9: TToolButton
      HelpType = htContext
      Left = 6
      Top = 3
      Height = 23
      Action = FileNew1
      Caption = '&New'
    end
    object ToolButton1: TToolButton
      HelpType = htContext
      Left = 52
      Top = 3
      Height = 23
      Action = FileOpen1
      Caption = '&Open'
    end
    object ToolButton2: TToolButton
      HelpType = htContext
      Left = 29
      Top = 3
      Height = 23
      Action = FileSave1
      Caption = '&Save'
      Enabled = False
    end
    object ToolButton3: TToolButton
      Left = 75
      Top = 3
      Width = 8
      Height = 23
      Style = tbsSeparator
      Caption = 'ToolButton3'
```

```
    ImageIndex = 2
  end
  object ToolButton4: TToolButton
    HelpType = htContext
    Left = 83
    Top = 3
    Height = 23
    Action = EditCut1
    Caption = 'Cu&t'
  end
  object ToolButton5: TToolButton
    HelpType = htContext
    Left = 106
    Top = 3
    Height = 23
    Action = EditCopy1
    Caption = '&Copy'
  end
  object ToolButton6: TToolButton
    HelpType = htContext
    Left = 129
    Top = 3
    Height = 23
    Action = EditPaste1
    Caption = '&Paste'
  end
  object ToolButton7: TToolButton
    Left = 152
    Top = 3
    Width = 8
    Height = 23
    Style = tbsSeparator
    Caption = 'ToolButton7'
    ImageIndex = 3
  end
  object ToolButton8: TToolButton
    HelpType = htContext
    Left = 160
    Top = 3
    Height = 23
    Hint = 'Layers'
    Caption = '&Cascade'
    ImageIndex = 17
    OnClick = Layers1Click
  end
  object ToolButton10: TToolButton
    HelpType = htContext
    Left = 183
    Top = 3
    Height = 23
    Action = WindowTile1
    Caption = '&Tile'
  end
  object ToolButton12: TToolButton
    HelpType = htContext
    Left = 206
    Top = 3
```

```
      Height = 23
      Hint = 'TreeView'
      Caption = 'C&lose'
      ImageIndex = 18
      OnClick = TreeView1Click
    end
    object ComboBox1: TComboBox
      Left = 238
      Top = 3
      Width = 60
      Height = 21
      Hint = 'Zoom| Amplia o reduce el '#225'rea visualizada del circuito.'
      Style = csDropDownList
      ItemHeight = 15
      Items.Strings = (
        '25%'
        '50%'
        '75%'
        '100%'
        '150%'
        '200%')
      ItemIndex = 3
      TabOrder = 11
      Text = '100%'
      OnChange = ComboBox1Change
    end
    object ToolButton30: TToolButton
      Left = 306
      Top = 3
      Height = 23
      Hint = 'Simulate|Create the netlist and begin simulation'
      Caption = 'ToolButton30'
      ImageIndex = 19
      OnClick = ToolButton30Click
    end
    object ToolButton31: TToolButton
      Left = 298
      Top = 3
      Width = 8
      Height = 23
      Style = tbsSeparator
      Caption = 'ToolButton31'
      ImageIndex = 20
    end
    object ToolButton32: TToolButton
      Left = 229
      Top = 3
      Width = 9
      Height = 23
      Style = tbsSeparator
      Caption = 'ToolButton32'
      ImageIndex = 20
    end
  end
end
object ToolBar1: TToolBar
  Left = 435
```

```
Top = 41
Width = 67
Height = 313
Align = alRight
AutoSize = True
ButtonHeight = 33
ButtonWidth = 33
Caption = 'ToolBar1'
Color = clButton
Images = DeviceList1
ParentColor = False
TabOrder = 2
Wrapable = False
object ToolButton11: TToolButton
  Left = 1
  Top = 202
  Height = 33
  Caption = 'ToolButton11'
  ImageIndex = 0
  Visible = False
end
object ToolButton13: TToolButton
  Left = 1
  Top = 4
  Height = 33
  Hint = 'Resistencia|Inserta una resistencia en el circuito.'
  Caption = 'ToolButton13'
  ImageIndex = 1
  ParentShowHint = False
  ShowHint = True
  OnClick = ToolButton13Click
end
object ToolButton14: TToolButton
  Left = 1
  Top = 37
  Height = 33
  Hint = 'Capacitor|Inserta un capacitor en el circuito.'
  Caption = 'ToolButton14'
  ImageIndex = 2
  ParentShowHint = False
  ShowHint = True
  OnClick = ToolButton14Click
end
object ToolButton15: TToolButton
  Left = 1
  Top = 70
  Height = 33
  Hint = 'Bobina|Inserta una inductancia en el circuito.'
  Caption = 'ToolButton15'
  ImageIndex = 3
  ParentShowHint = False
  ShowHint = True
  OnClick = ToolButton15Click
end
object ToolButton16: TToolButton
  Left = 1
  Top = 103
```

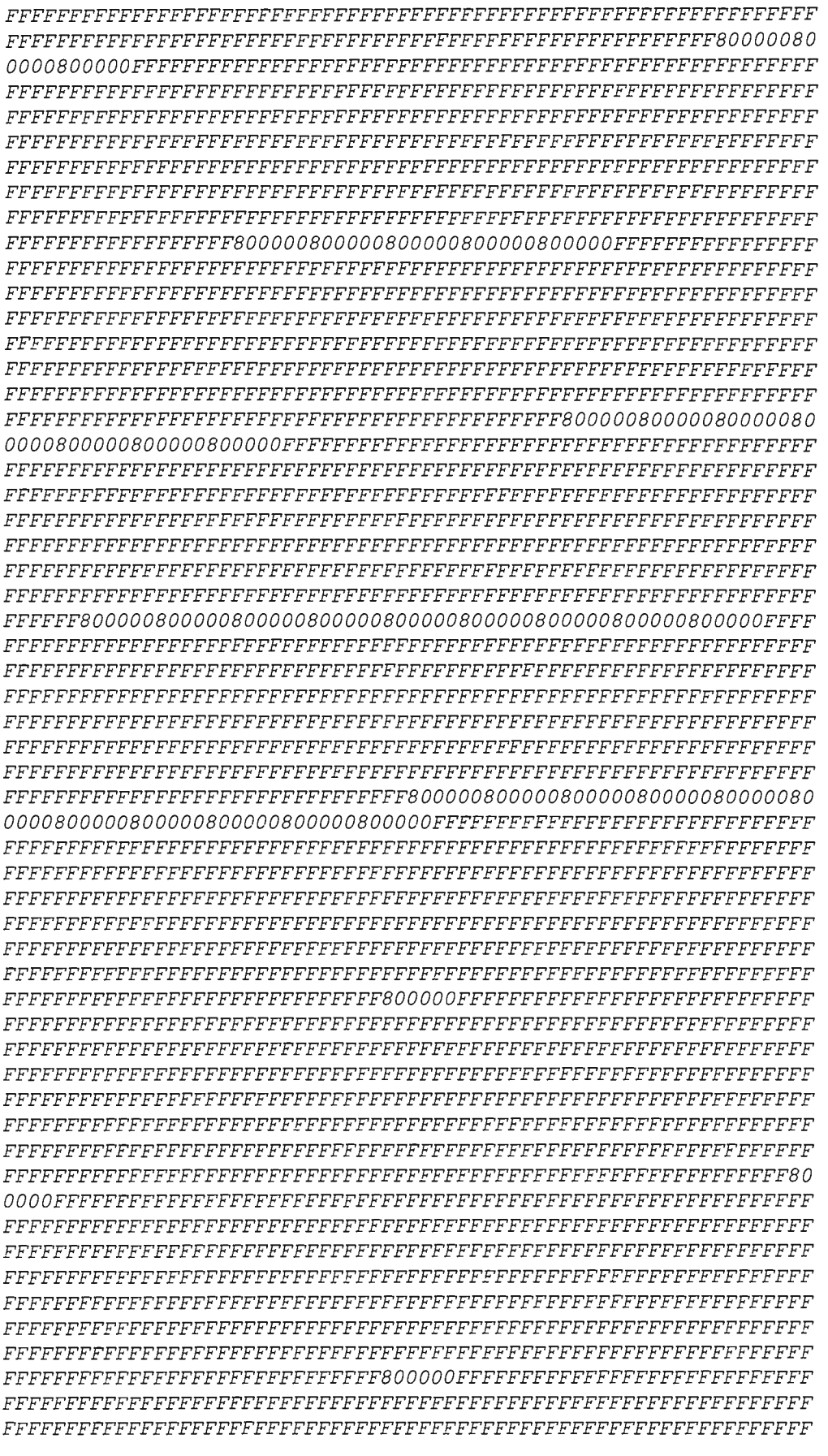
```
    Height = 33
    Hint = 'Diodo|Inserta un diodo en el circuito.'
    Caption = 'ToolButton16'
    ImageIndex = 4
    ParentShowHint = False
    ShowHint = True
    OnClick = ToolButton16Click
  end
  object ToolButton17: TToolButton
    Left = 1
    Top = 169
    Height = 33
    Hint = 'Fuente DC|Inserta una fuente DC en el circuito.'
    Caption = 'ToolButton17'
    ImageIndex = 5
    ParentShowHint = False
    ShowHint = True
    OnClick = ToolButton17Click
  end
  object ToolButton18: TToolButton
    Left = 1
    Top = 235
    Height = 33
    Caption = 'ToolButton18'
    ImageIndex = 6
    Visible = False
  end
  object ToolButton19: TToolButton
    Left = 1
    Top = 268
    Height = 33
    Caption = 'ToolButton19'
    ImageIndex = 7
    Visible = False
  end
  object ToolButton20: TToolButton
    Left = 1
    Top = 301
    Height = 33
    Caption = 'ToolButton20'
    ImageIndex = 8
    Wrap = True
    OnClick = ToolButton20Click
  end
  object ToolButton21: TToolButton
    Left = 34
    Top = 0
    Height = 33
    Hint = 'Alambre|Realiza una conexi'#243'n entre dos o m'#225's dispositivos.'
    Caption = 'ToolButton21'
    ImageIndex = 9
    ParentShowHint = False
    ShowHint = True
    OnClick = ToolButton21Click
  end
  object ToolButton22: TToolButton
    Left = 1
```

```
    Top = 136
    Height = 33
    Hint = 'Transistor|Inserta un transistor en el circuito.'
    Caption = 'ToolButton22'
    ImageIndex = 10
    ParentShowHint = False
    ShowHint = True
    OnClick = ToolButton22Click
  end
  object ToolButton23: TToolButton
    Left = 34
    Top = 33
    Height = 33
    Caption = 'ToolButton23'
    ImageIndex = 11
    OnClick = ToolButton23Click
  end
  object ToolButton24: TToolButton
    Left = 34
    Top = 66
    Height = 33
    Hint = 'Tierra|Inserta una conexion a tierra al circuito.'
    Caption = 'ToolButton24'
    ImageIndex = 12
    ParentShowHint = False
    ShowHint = True
    OnClick = ToolButton24Click
  end
  object ToolButton25: TToolButton
    Left = 34
    Top = 99
    Height = 33
    Caption = 'ToolButton25'
    ImageIndex = 13
    OnClick = ToolButton25Click
  end
  object ToolButton26: TToolButton
    Left = 34
    Top = 132
    Height = 33
    Caption = 'ToolButton26'
    ImageIndex = 14
    OnClick = ToolButton26Click
  end
  object ToolButton27: TToolButton
    Left = 34
    Top = 165
    Height = 33
    Caption = 'ToolButton27'
    ImageIndex = 15
    OnClick = ToolButton27Click
  end
  object ToolButton28: TToolButton
    Left = 34
    Top = 198
    Height = 33
    Caption = 'ToolButton28'
```

```
      ImageIndex = 16
      OnClick = ToolButton28Click
    end
    object ToolButton29: TToolButton
      Left = 34
      Top = 231
      Height = 33
      Caption = 'ToolButton29'
      ImageIndex = 17
      Visible = False
    end
  end
  object OpenDialog: TOpenDialog
    Height = 0
    Width = 0
    Filter = 'All files (*)|*'
    FilterIndex = 0
    Title = 'Open'
    Left = 8
    Top = 200
  end
  object ActionList1: TActionList
    Images = ImageList1
    Left = 72
    Top = 200
    object FileNew1: TAction
      Category = 'File'
      Caption = '&New'
      Hint = 'New|Create a new file'
      ImageIndex = 6
      ShortCut = 16462
      OnExecute = FileNew1Execute
    end
    object FileOpen1: TAction
      Category = 'File'
      Caption = '&Open'
      Hint = 'Open|Open a file'
      ImageIndex = 7
      ShortCut = 16463
      OnExecute = FileOpen1Execute
    end
    object FileClose1: TWindowClose
      Category = 'File'
      Caption = '&Close'
      Hint = 'Close|Close current file'
    end
    object FileSave1: TAction
      Category = 'File'
      Caption = '&Save'
      Hint = 'Save|Save current file'
      ImageIndex = 8
      ShortCut = 16467
      OnExecute = FileSaveItemClick
    end
    object FileSaveAs1: TAction
      Category = 'File'
      Caption = 'Save &As...'
```

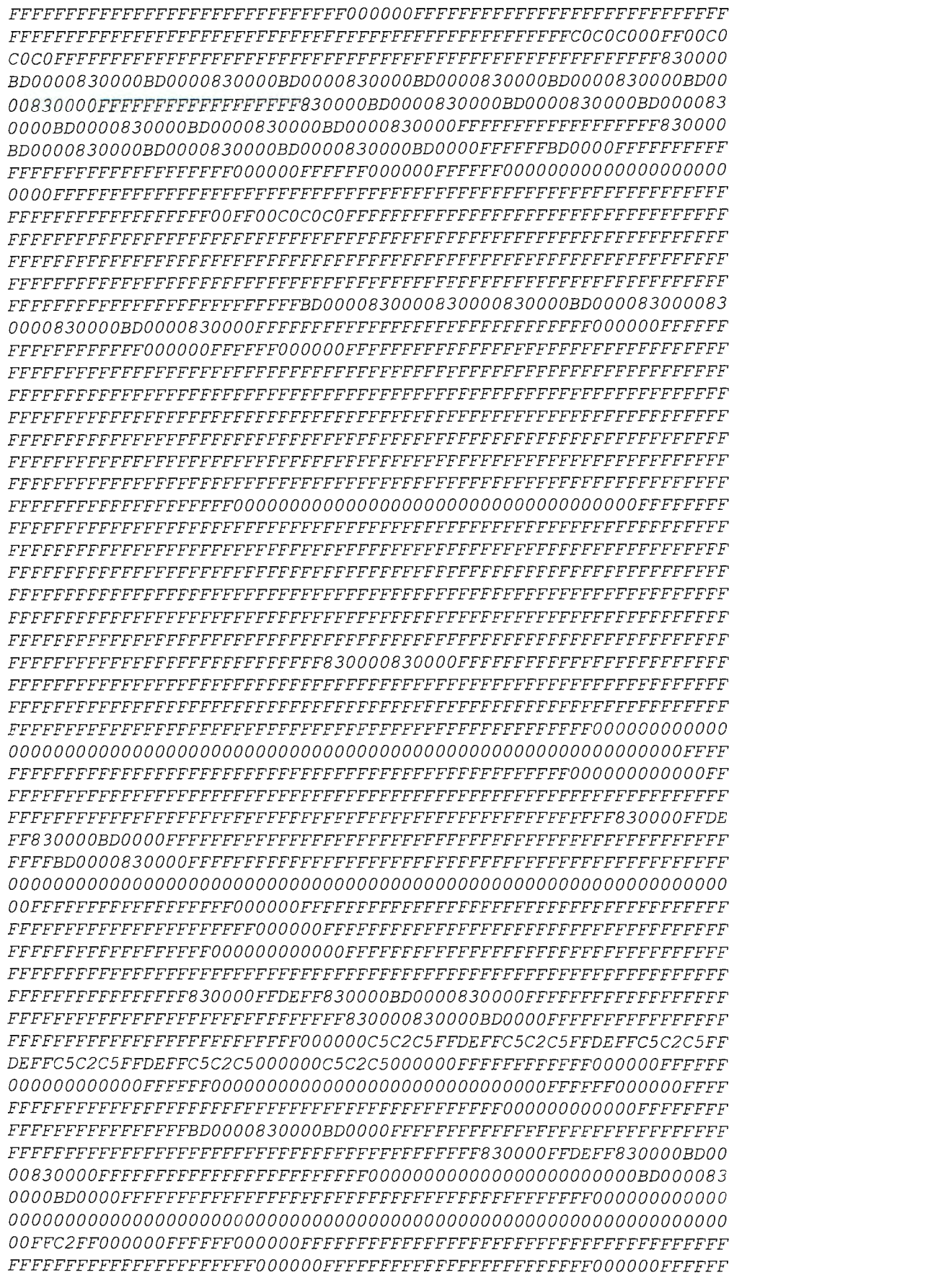
```
    Hint = 'Save As|Save current file with different name'
  end
  object FileExit1: TAction
    Category = 'File'
    Caption = 'E&xit'
    Hint = 'Exit|Exit application'
    OnExecute = FileExit1Execute
  end
  object EditCut1: TEditCut
    Category = 'Edit'
    Caption = 'Cu&t'
    Hint = 'Cut|Cuts the selection and puts it on the Clipboard'
    ImageIndex = 0
    ShortCut = 16472
    OnExecute = CutItemClick
  end
  object EditCopy1: TEditCopy
    Category = 'Edit'
    Caption = '&Copy'
    Hint = 'Copy|Copies the selection and puts it on the Clipboard'
    ImageIndex = 1
    ShortCut = 16451
    OnExecute = CopyItemClick
  end
  object EditPaste1: TEditPaste
    Category = 'Edit'
    Caption = '&Paste'
    Hint = 'Paste|Inserts Clipboard contents'
    ImageIndex = 2
    ShortCut = 16470
    OnExecute = PasteItemClick
  end
  object WindowCascade1: TWindowCascade
    Category = 'Window'
    Caption = '&Cascade'
    Hint = 'Cascade'
    ImageIndex = 17
  end
  object WindowMinimizeAll1: TWindowMinimizeAll
    Category = 'Window'
    Caption = '&Minimize All'
    Hint = 'Minimize All'
  end
  object HelpAbout1: TAction
    Category = 'Help'
    Caption = '&About...'
    Hint =
      'About|Displays program information, version number, and copyrigh' +
      't'
    OnExecute = HelpAbout1Execute
  end
  object WindowClose1: TWindowClose
    Category = 'Window'
    Caption = 'C&lose'
    Hint = 'Close'
    ImageIndex = 18
  end
```

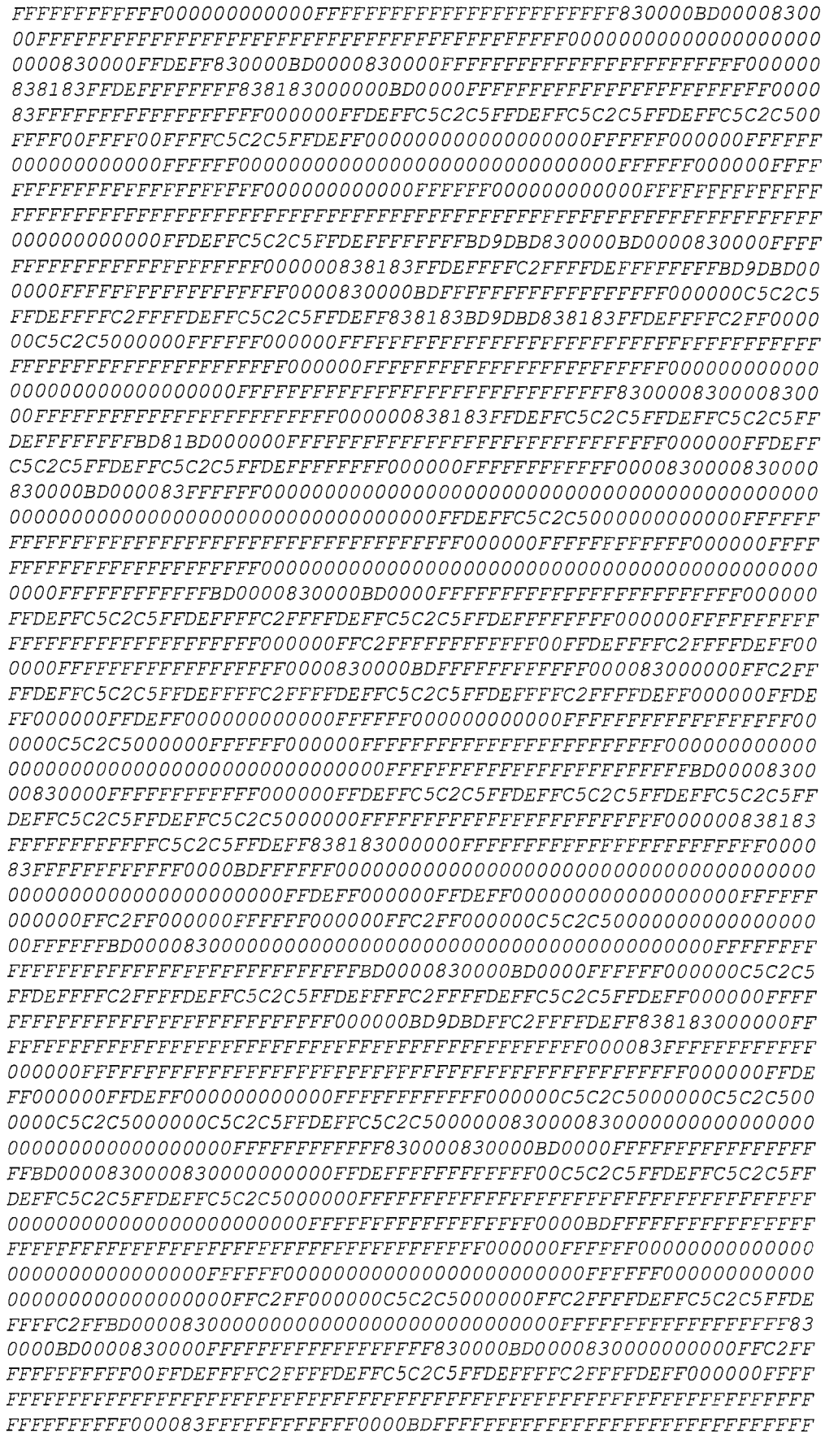
```
    object WindowTile1: TWindowTile
      Category = 'Window'
      Caption = '&Tile'
      Hint = 'Tile'
      ImageIndex = 15
    end
  end
  object ImageList1: TImageList
    Left = 112
    Top = 200
    Bitmap = {
      494D474C010001001000000010000000150000000424D364B000000000003600
      0000280000005000000050000000010018000000000004B0000120B0000120B
      00000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
      0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF800000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
      0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF800000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
      0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      8000008000008000008000008000008000008000008000008000008000008000
      00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

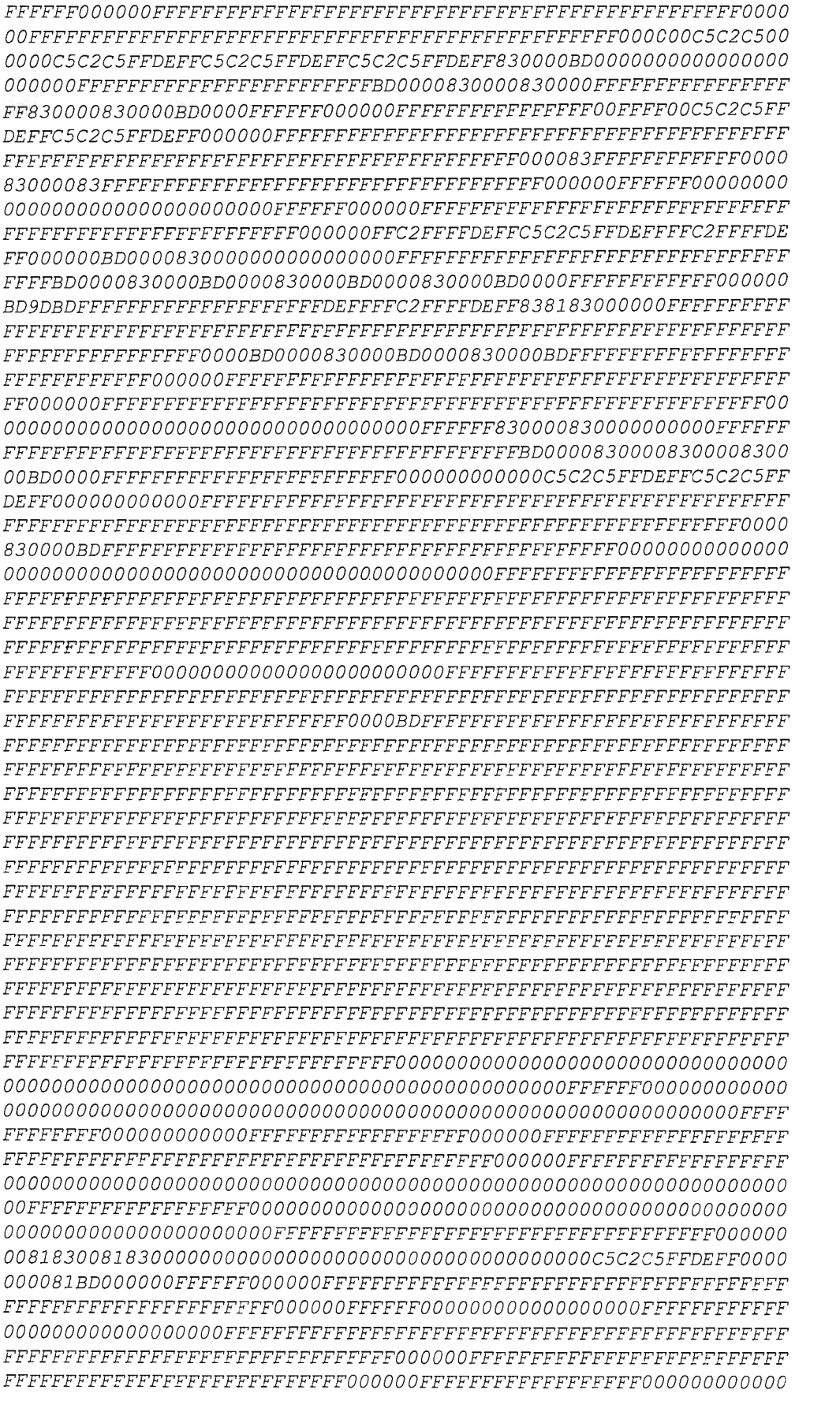


FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF800000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000BD
0000830000830000830000BD0000830000830000830000BD0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF838583FFFFFF8385
83FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
BD0000830000BD0000830000BD0000830000BD0000830000BD0000830000BD00
00830000FFFFFFFFFFFFFFFFFF830000BD0000830000BD0000830000BD000083
0000BD0000830000BD0000830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFF
FFFFFFFFFF838583AC95ACFFFFFFAC95AC838583FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C0000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFBD0000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000838583000000
000000000000000000000000FFFFFFFFFFFF838583FFFFFF838583FFFFFF8385
83FFFFFF838583FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0C0C000FF00C0
C0C0000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFF830000000000000000000000000000000000000000FFFFFF83
8583AC95AC838583FFFFFFFFFFFFFFFFFF838583AC95AC838583FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C000FF00C0C0C0000000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000830000830000830000BD
0000830000830000830000BD0000830000830000830000BD0000000000FFFFFF
FFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0C0C000FF00C0
C0C000FF00C0C0C0000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000

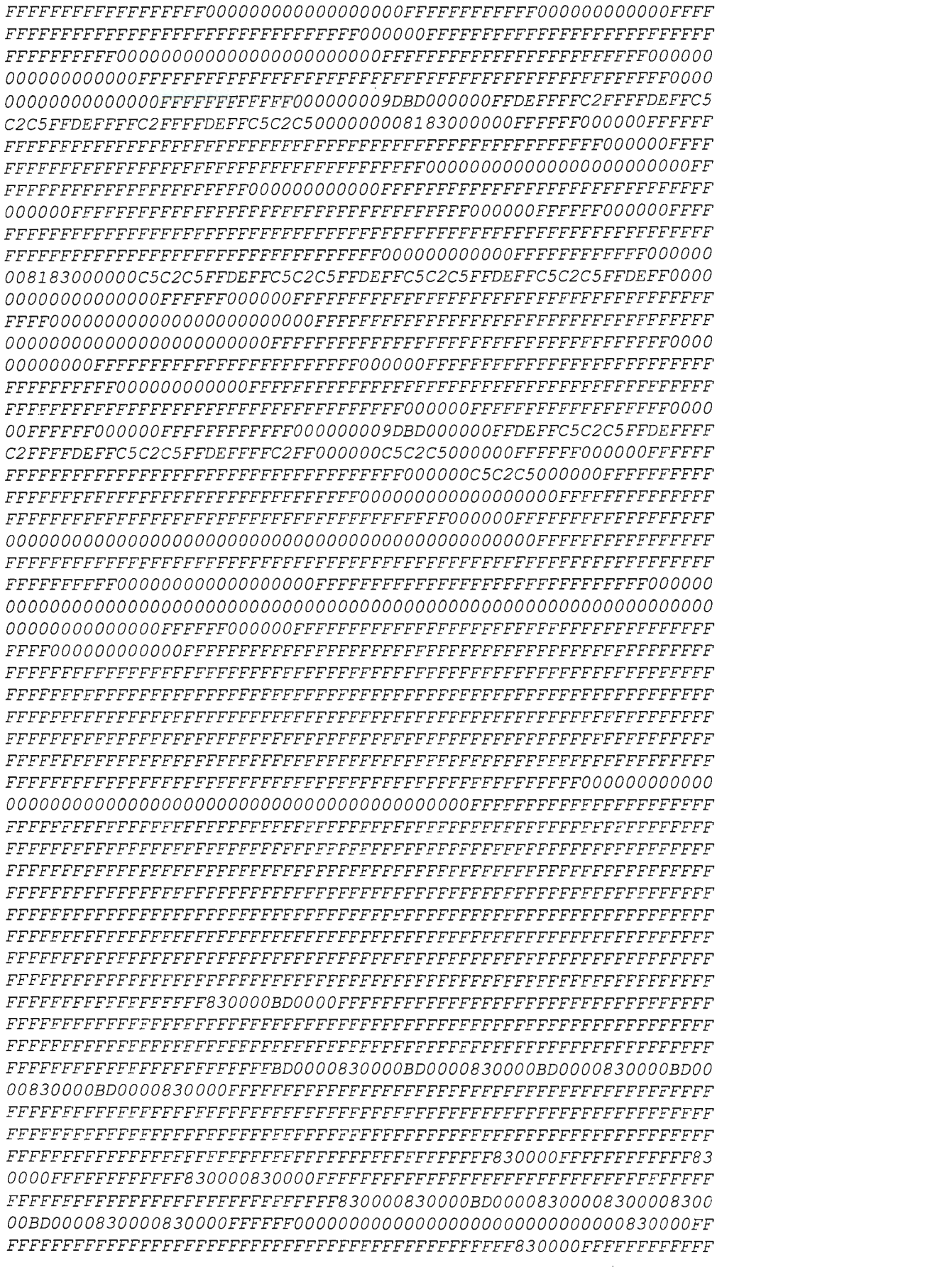
BD0000830000BD0000830000BD0000830000BD0000830000BD0000830000BD00
00830000FFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF830000FFFFFFFFFFFFBD0000830000BD0000830000BD0000830000BD00
00830000FFFFFF830000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
8583AC95AC838583FFFFFFFFFFFFFFFFFF838583AC95AC838583FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C000FF00C0C0C000FF00C0C0C0000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFBD000083000083000083000BD000083000083
0000830000BD000083000083000FFFFFFBD0000FFFFFFFFFFFFFFFFFFBD0000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFF83000083
0000830000BD000083000083000083000BD000083000083000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF838583FFFFFF838583FFFFFF8385
83FFFFFF838583FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0C0C000FF00C0
C0C000FF00C0C0C000FF00C0C0C0000000FFFFFFFFFFFFFFFFFFFFFFFFFF830000
BD0000830000BD0000830000BD0000830000BD0000830000BD0000830000BD00
00830000FFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD00
00FFFFFFFFFFFFFFFFFF0000000000000000000000000000000000000000000000
0000FFFFFF838583AC95ACFFFFFFAC95AC838583FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C000FF00C0C0C000FF00C0C0C000FF00C0C0
C0FFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFF830000830000BD0000830000830000830000BD
0000830000830000830000BD0000830000FFFFFFFFFFFFFFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFF838583FFFFFF8385
83FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0C0C000FF00C0
C0C000FF00C0C0C000FF00C0C0C0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFF830000
FFFFFF830000BD0000830000BD0000830000BD0000830000BD0000FFFFFFBD00
00FFFFFFFFFFFFFFFFFF000000FFFFFF0000000000000000000000000FFFFFF00
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C000FF00C0C0C000FF00C0C0C0FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFBD0000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFBD0000FFFFFFFFFFFFFFFFFFBD0000FFFFFF830000830000BD000083000083
0000830000BD000083000083000083000FFFFFFFFFFFFFFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0C0C000FF00C0
C0C000FF00C0C0C0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
BD0000830000BD0000830000BD0000830000BD0000830000BD0000830000BD00
00830000FFFFFFFFFFFFFFFFFF830000BD0000830000BD0000830000BD000083
0000BD0000830000BD0000830000BD0000830000FFFFFFFFFFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFF000000FFFFFF0000000000000000000000000FFFFFF00
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF00FF00C0C0C000FF00C0C0C0FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF830000830000BD0000830000830000830000BD
0000830000830000830000BD0000FFFFFF830000FFFFFFFFFFFFFFFFFFFF830000
830000BD0000830000830000FFFFFFBD0000830000830000830000BD0000FFFF
FF830000FFFFFFFFFFFFFFFFFF830000830000BD0000830000830000830000BD
0000830000830000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF




FFFFFFFFFFFF000000000000FFFFFFFFFFFFFFFFFFFFFFFF830000BD00008300
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000
0000830000FFDEFF830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFF000000
838183FFDEFFFFFFFF838183000000BD0000FFFFFFFFFFFFFFFFFFFFFFFF0000
83FFFFFFFFFFFFFFFFFF000000FFDEFFC5C2C5FFDEFFC5C2C5FFDEFFC5C2C500
FFFF00FFFF00FFFFC5C2C5FFDEFF000000000000000000FFFFFF000000FFFFFF
000000000000FFFFFF000000000000000000000000000000FFFFFF000000FFFF
FFFFFFFFFFFFFFFFFFFF000000000000FFFFFF000000000000FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000000000FFDEFFC5C2C5FFDEFFFFFFFFBD9DBD830000BD0000830000FFFF
FFFFFFFFFFFFFFFFFFFF000000838183FFDEFFFFC2FFFFDEFFFFFFFFBD9DBD00
0000FFFFFFFFFFFFFFFFFF0000830000BDFFFFFFFFFFFFFFFFFF000000C5C2C5
FFDEFFFFC2FFFFDEFFC5C2C5FFDEFF838183BD9DBD838183FFDEFFFFC2FF0000
00C5C2C5000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFF000000000000
0000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000830000830000
00FFFFFFFFFFFFFFFFFFFFFFFFFF000000838183FFDEFFC5C2C5FFDEFFC5C2C5FF
DEFFFFFFFFBD81BD000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFDEFF
C5C2C5FFDEFFC5C2C5FFDEFFFFFFFFF000000FFFFFFFFFFFFF0000830000830000
830000BD000083FFFFFF000000000000000000000000000000000000000000000
0000000000000000000000000000000000FFDEFFC5C2C5000000000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFF000000FFFF
FFFFFFFFFFFFFFFFFFFF000000000000000000000000000000000000000000000
0000FFFFFFFFFFFFFBD0000830000BD0000FFFFFFFFFFFFFFFFFFFFFFFFF000000
FFDEFFC5C2C5FFDEFFFFC2FFFFDEFFC5C2C5FFDEFFFFFFFFF000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFF000000FFC2FFFFFFFFFFFF00FFDEFFFFC2FFFFDEFF00
0000FFFFFFFFFFFFFFFFFF0000830000BDFFFFFFFFFFFFF000083000000FFC2FF
FFDEFFC5C2C5FFDEFFFFC2FFFFDEFFC5C2C5FFDEFFFFC2FFFFDEFF000000FFDE
FF000000FFDEFF000000000000FFFFFF000000000000FFFFFFFFFFFFFFFFFF00
0000C5C2C5000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFF000000000000
0000000000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFBD00008300
00830000FFFFFFFFFFFF000000FFDEFFC5C2C5FFDEFFC5C2C5FFDEFFC5C2C5FF
DEFFC5C2C5FFDEFFC5C2C5000000FFFFFFFFFFFFFFFFFFFFFFFF000000838183
FFFFFFFFFFFFC5C2C5FFDEFF838183000000FFFFFFFFFFFFFFFFFFFFFFFF0000
83FFFFFFFFFFFF0000BDFFFFFF000000000000000000000000000000000000
0000000000000000000000FFDEFF000000FFDEFF000000000000000000FFFFFF
000000FFC2FF000000FFFFFF000000FFC2FF000000C5C2C500000000000000000
00FFFFFFBD000083000000000000000000000000000000000000000000FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000830000BD0000FFFFFF000000C5C2C5
FFDEFFFFC2FFFFDEFFC5C2C5FFDEFFFFC2FFFFDEFFC5C2C5FFDEFF000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF000000BD9DBDFFC2FFFFDEFF838183000000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000083FFFFFFFFFFFF
000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFDE
FF000000FFDEFF000000000000FFFFFFFFFFFF000000C5C2C5000000C5C2C500
0000C5C2C5000000C5C2C5FFDEFFC5C2C5000000830000830000000000000000
0000000000000000000FFFFFFFFFFFF830000830000BD0000FFFFFFFFFFFFFFFF
FFBD000083000083000000000FFDEFFFFFFFFFFFFFF00C5C2C5FFDEFFC5C2C5FF
DEFFC5C2C5FFDEFFC5C2C5000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000000000000000000000000FFFFFFFFFFFFFFFFFFF0000BDFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF000000000000000
0000000000000000FFFFFF0000000000000000000000000FFFFFF000000000000
0000000000000000000FFC2FF000000C5C2C5000000FFC2FFFFDEFFC5C2C5FFDE
FFFFC2FFBD000083000000000000000000000000000FFFFFFFFFFFFFFFFFF83
0000BD0000830000FFFFFFFFFFFFFFFFFF830000BD000083000000000FFC2FF
FFFFFFFFFF00FFDEFFFFFC2FFFFDEFFC5C2C5FFDEFFFFFC2FFFFDEFF000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF000083FFFFFFFFFFFFF0000BDFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF



```
009DBD008183009DBD008183009DBD008183009DBD008183009DBD000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF000000009DBD008183000000000000000000000
0000000000000000FFDEFFC5C2C5000000008183000000FFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000000000000
0000000000000FFFFFFFFFFFFFFFFFF00000000000000000000000FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFF
000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFF00000000FFFF0000000081BD008183008183008183300
81BD008183008183081830081BD000000FFFFFFFFFFFFFFFFFFFFFFFF000000
0081830081BD0000000000000000000000000000000C5C2C5FFDEFF0000
00008183000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
0000838183FFDEFFC5C2C5BD81BD000000000000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFF000000FFFFFF
00FFFF000000009DBD008183009DBD008183009DBD008183009DBD008183009D
BD000000FFFFFFFFFFFFFFFFFF000000009DBD008183000000000000000000000
0000000000000000000000000000000000008183000000FFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF000000838183FFDEFFC5C2C5FFFF00838183BD9D
BD000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000FF
FFFFFFFFFFFFFFFFFFFFFFFF000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFF00000000FFFFFFFFFF00FFFF0000000081BD00818300
8183008183008183008183008183008183008183008183008183008183008183
830081BD000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FF
DEFFC5C2C5FFDEFFC5C2C5838183C5C2C5000000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF000000000000000000FFFFFFFFFFFF000000000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFF000000FFFFFF
00FFFFFFFFFF00FFFF000000000000000000000000000000000000000000000000
0000000000000000000FFFFFF000000009DBD008183000000000000000000000
0000000000000000000000000000009DBD008183000000FFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF000000C5C2C5FFFF00FFC2FFFFDEFF838183FFDE
FF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000
0000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFF00000000FFFFFFFFFF00FFFFFFFFFF00FFFFFFFFFF00
FFFFFFFFFF00FFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
008183000000C5C2C5FFDEFFC5C2C5FFDEFFC5C2C5FFDEFFC5C2C5FFDEFF0000
00008183000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000BD
81BDFFFF00FFFF00C5C2C5BD81BD838183000000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000000000000000000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFF000000FFFFFF
00FFFFFFFFFF00FFFFFFFFFF00FFFFFFFFFF00FFFFFFFFFF000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF000000009DBD000000FFDEFFC5C2C5FFDEFFFF
C2FFFFDEFFC5C2C5FFDEFFFFC2FF00000000008183000000FFFFFF000000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000BD9DBDC5C2C5FFDEFF8381830000
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000
0000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFF00000000FFFFFFFFFF00FFFF0000000000000000000000
0000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
008183000000C5C2C5FFDEFFC5C2C5FFDEFFC5C2C5FFDEFFC5C2C5FFDEFF0000
000081BD000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF0000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```



FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFBD0000FFFFFFFFFFFF830000FFFFFF830000FFFFFFFFFFFFBD00
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000000000838183
009DBD838183009DBD838183BD0000FFFFFF0000000000000000000000000000
00000000FFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFBD
0000FFFFFF830000FFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFF0000000000000000000000
00000000FFFFFFBD000000000000818383818300 81BD838183008183830000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF830000BD0000830000FFFFFF830000FFFFFFFFFFFFBD00
00FFFFFFFFFFFFFFFFFFFFFFFF00000000000000000000000000000000000083
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000000000838183
009DBD838183009DBD838183BD0000FFFFFF000000000000000000FFFFFFBD00
00830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFBD0000830000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFF0000000000000000000000
00000000FFFFFF8300000000000081BD8381830081838381830081BD830000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFF830000FFFFFFFFFFFFFFFFFF
830000830000830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8300
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFBD0000830000830000830000BD0000FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000000000830000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF00000000000000000000000083
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000000000838183
009DBD838183009DBD838183BD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD00
00830000FFFFFFFFFFFFFFFFFFFFFFFFBD0000830000BD0000830000FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
BD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000830000BD00
00830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFF000000000000FFFFFF8300
00830000830000BD00000000000081838381830081BD838183008183830000BD
0000830000830000830000BD0000830000000000FFFFFFFFFFFFFFFFFFFFFFFF
830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFBD0000830000830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000000000000000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF00000000000000000000000083
0000FFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFBD0000FFFFFF000000838183
009DBD838183009DBD838183009DBD838183009DBD838183009DBD838183009D
BD000000FFFFFFFFFFFFFFFFFFFFFFFFBD0000830000FFFFFF830000FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
BD0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFBD0000FFFFFFBD00
00830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
0000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF8300
00BD0000FFFFFFFFFFFF0000000081BD83818300000000000000000000000000
0000000000000000000000838183838183000000FFFFFFFFFFFFFFFFFFFFFFFF
830000FFFFFFFFFFFFFFFFFF830000830000FFFFFFFFFFFFFFFFFFFFFFFF8300
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFF
FFFF830000BD0000FFFFFFFFFFFFFFFFFFBD0000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF000000000000FFFFFF000000000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF000000000000FFFFFF00000083
0000BD0000830000BD0000830000BD0000FFFFFFFFFFFFFFFFFF000000838183
BD9DBD000000FFDEFFFFC2FFFFDEFFC5C2C5FFDEFFFFC2FF000000838183009D
BD000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFBD0000830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFBD0000830000BD0000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FF
FFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF000000FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF0000000081838381830081BD00000000FFFF00000000
000000FFFF0000008381830081BD838183000000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFF000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFF00000000
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
00000000000000000000000000FFFF00FFFF0000000000000000000000000000
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FF
FFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
000000000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000000000
0000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF424DFE030000000000003E0000002800000050000000
5000000001000100000000000C0030000120B0000120B0000020000000200000
FFFFFF00000000000100FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFFFFFFFFFF
0100FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFF
FFFFFFFF3FF8FFFFFFFFFFFFFFFFFFFF0380FFFFFFFFFFFFFFFFFFFF07C0FFFF
FFFFFFFFFFFFFFFF0FE0FFFFFFFFFFFFFFFFFFFF1FF0FFFFFFFFFFFFFFFFFFFF
3FF8FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFF
FFFFFFFF0100FFFFFFFFFFFFFFFFFFFF0100FFFFFFFFFFFFFFFFFFFF0100FFFF
FFFFFFFFFFFFFFFF0000000003FF00280600FFFF7FFC7FFC0201086C0700FFFF
400441040201FCAA0780FFFF400441040201FDC707C0FFFF400441041FFF8800
07E0FFFF7FFC410413FD81C707F0FFFF7FF4410413FF00AA07F8FFFF7FFC4104
1008FF6C07F8FFFF400441047FF8812807F0FFFF400441045FE8BD0007E0FFFF
400441045FF8810007C0FFFF7FFC7FFC4020BD000780FFFF7FF47DF47FE08100
0700FFFF7FFC7FFC7FA0AF000600FFFF000000007FE08A000000FFFF00000000

```
    0000FC000000FFFF00000000000C00000000FFFFFFF00600001E00C03FF8FFFF
    80100600003E01C07FFCFFFFB7D00C38007C3F80FFFEFFFF80108C380FF87708
    FFFEFFFFB7D0D8003EF0FB18FFFEFFFF8010F8387F60FD3EFFFFFFFF8090FF38
    7FA0DF19FFFFFFFFB1D0FE1CFFF0CF097FFFFFFFBBFBFC0EFFF07E01201FFFFF
    9FFFF9C7DFF03C4017DEFFFFFFFFF1C7DFF000481008FFFF07FFE1C74FE0004C
    0BE8FFFF03FFC0FE63E0003E0804FFFF01FB807C3FC0000C07FCFFFF00000000
    0F0000080000FFFF000000000000000000000FFFF0000000000003FFEFFF3FFFF
    10023FF8FFE07FFE8017FFFF38002008FFF07FFE80FEFFFF3C042008FFF87FFE
    81FCFFFF1C082008BFFC7FFE83FCFFFF0E182008DFFE7FFE83FCFFFF07302008
    AFFF7FFE83FCFFFF03E02008D5607FFE83FCFFFF01C02008AAA07FFE81F8FFFF
    03E02008DFE07FFE80F0FFFF07302078700E7FFE8010FFFF1E18205000067FFE
    80F0FFFF3C0C2060008A7FFE80E0FFFF38023FC000707FFE80C0FFFF00000000
    00000000FF80FFFF000000000000000000000FFFF0600000003FF00000000FFFF
    093001FF7E0100000000FFFF09480101FEFD00000000FFFF0948017DFE010000
    0000FFFF07487F01FEEF00081000FFFF0170417DFE0A3E08107CFFFF01C05F01
    FE0C3C04203CFFFF0080416FFFFC3804201CFFFF01C05F0AFFFC3404202CFFFF
    0140410CFFFC230810C4FFFF03605BF8FFFC00F00F00FFFF02204280FFFC0000
    0000FFFF022043007FF800000000FFFF02207E00078000000000FFFF00000000
    000000000000FFFF}
end
object MainMenu1: TMainMenu
  Images = ImageList1
  Left = 40
  Top = 200
  object File1: TMenuItem
    Caption = '&Archivo'
    Hint = 'File related commands'
    OnClick = File1Click
    object FileNewItem: TMenuItem
      Caption = '&Nuevo'
      Hint = 'New|Create a new file'
      ImageIndex = 6
      ShortCut = 16462
      OnClick = FileNew1Execute
    end
    object FileOpenItem: TMenuItem
      Caption = 'A&brir'
      Hint = 'Open|Open a file'
      ImageIndex = 7
      ShortCut = 16463
      OnClick = FileOpen1Execute
    end
    object FileCloseItem: TMenuItem
      Action = FileClose1
      Caption = '&Cerrar'
      Enabled = False
    end
    object FileSaveItem: TMenuItem
      Caption = '&Guardar'
      Hint = 'Save|Save current file'
      ImageIndex = 8
      ShortCut = 16467
      OnClick = FileSaveItemClick
    end
    object FileSaveAsItem: TMenuItem
      Action = FileSaveAs1
      Caption = 'Guardar C&omo...'
```

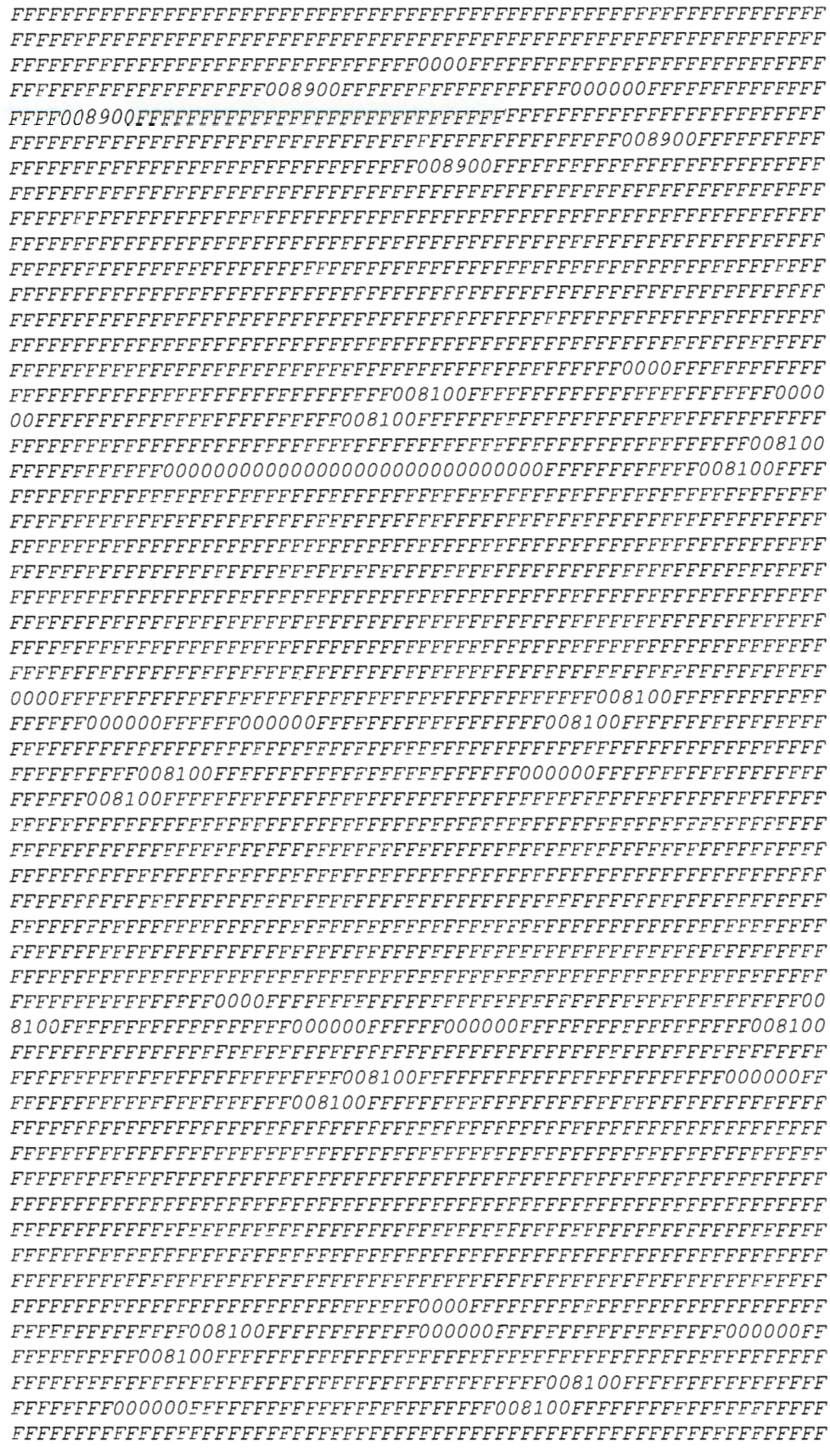
```
    end
    object N1: TMenuItem
      Caption = '-'
    end
    object FileExitItem: TMenuItem
      Action = FileExit1
      Caption = '&Salir'
    end
  end
  object Edit1: TMenuItem
    Caption = '&Editar'
    Hint = 'Edit commands'
    OnClick = Edit1Click
    object Deshacer1: TMenuItem
      Caption = 'Deshacer'
      Hint = 'Undo'
      ImageIndex = 3
      OnClick = Deshacer1Click
    end
    object Repetir1: TMenuItem
      Caption = 'Repetir'
      Hint = 'Redo'
      ImageIndex = 4
      OnClick = Repetir1Click
    end
    object N2: TMenuItem
      Caption = '-'
    end
    object CutItem: TMenuItem
      Caption = 'Cor&tar'
      Hint = 'Cut|Cuts the selection and puts it on the Clipboard'
      ImageIndex = 0
      ShortCut = 16472
      OnClick = CutItemClick
    end
    object CopyItem: TMenuItem
      Caption = '&Copiar'
      Hint = 'Copy|Copies the selection and puts it on the Clipboard'
      ImageIndex = 1
      ShortCut = 16451
      OnClick = CopyItemClick
    end
    object PasteItem: TMenuItem
      Caption = '&Pegar'
      Hint = 'Paste|Inserts Clipboard contents'
      ImageIndex = 2
      ShortCut = 16470
      OnClick = PasteItemClick
    end
    object Borrar1: TMenuItem
      Caption = 'Borrar'
      Hint = 'Delete the Selected Devices'
      ImageIndex = 5
      OnClick = Borrar1Click
    end
  end
```

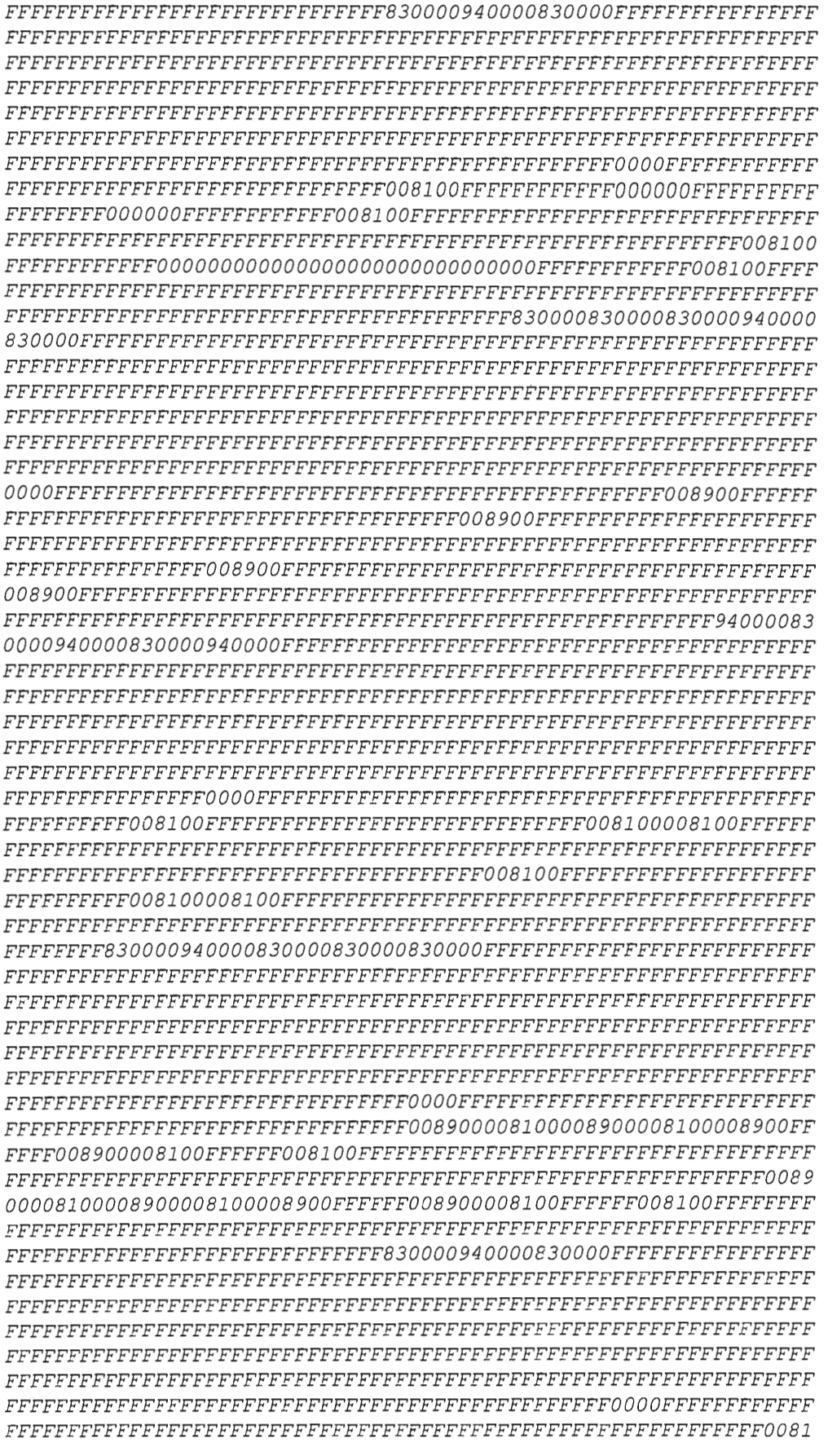
```
    object Window1: TMenuItem
      Caption = '&Ventanas'
      Hint = 'Window related commands'
      object TreeView1: TMenuItem
        Caption = 'T&reeView'
        Hint = 'Show the Objects Structure'
        ImageIndex = 18
        RadioItem = True
        OnClick = TreeView1Click
      end
      object Layers1: TMenuItem
        Caption = '&Layers'
        Hint = 'Show the Circuit Layers'
        ImageIndex = 17
        RadioItem = True
        OnClick = Layers1Click
      end
      object N3: TMenuItem
        Caption = '-'
      end
      object WindowTileItem: TMenuItem
        Action = WindowTile1
        Caption = '&Mosaico'
      end
      object WindowMinimizeItem: TMenuItem
        Action = WindowMinimizeAll1
        Caption = 'Mi&nimizar Todas'
      end
      object N4: TMenuItem
        Caption = '-'
      end
      object Modelos1: TMenuItem
        Caption = 'Mo&delos'
        ImageIndex = 20
        OnClick = Modelos1Click
      end
      object ConfigurarSimulacin1: TMenuItem
        Caption = 'Configurar &An'#225'lisis'
        OnClick = ConfigurarSimulacin1Click
      end
      object NetList1: TMenuItem
        Caption = 'Ver Ne&tlist'
        OnClick = NetList1Click
      end
    end
    object Help1: TMenuItem
      Caption = 'A&yuda'
      Hint = 'Help topics'
      object HelpAboutItem: TMenuItem
        Action = HelpAbout1
        Caption = 'A&cerca de...'
      end
    end
  end
  object DeviceList1: TImageList
    Height = 26
```

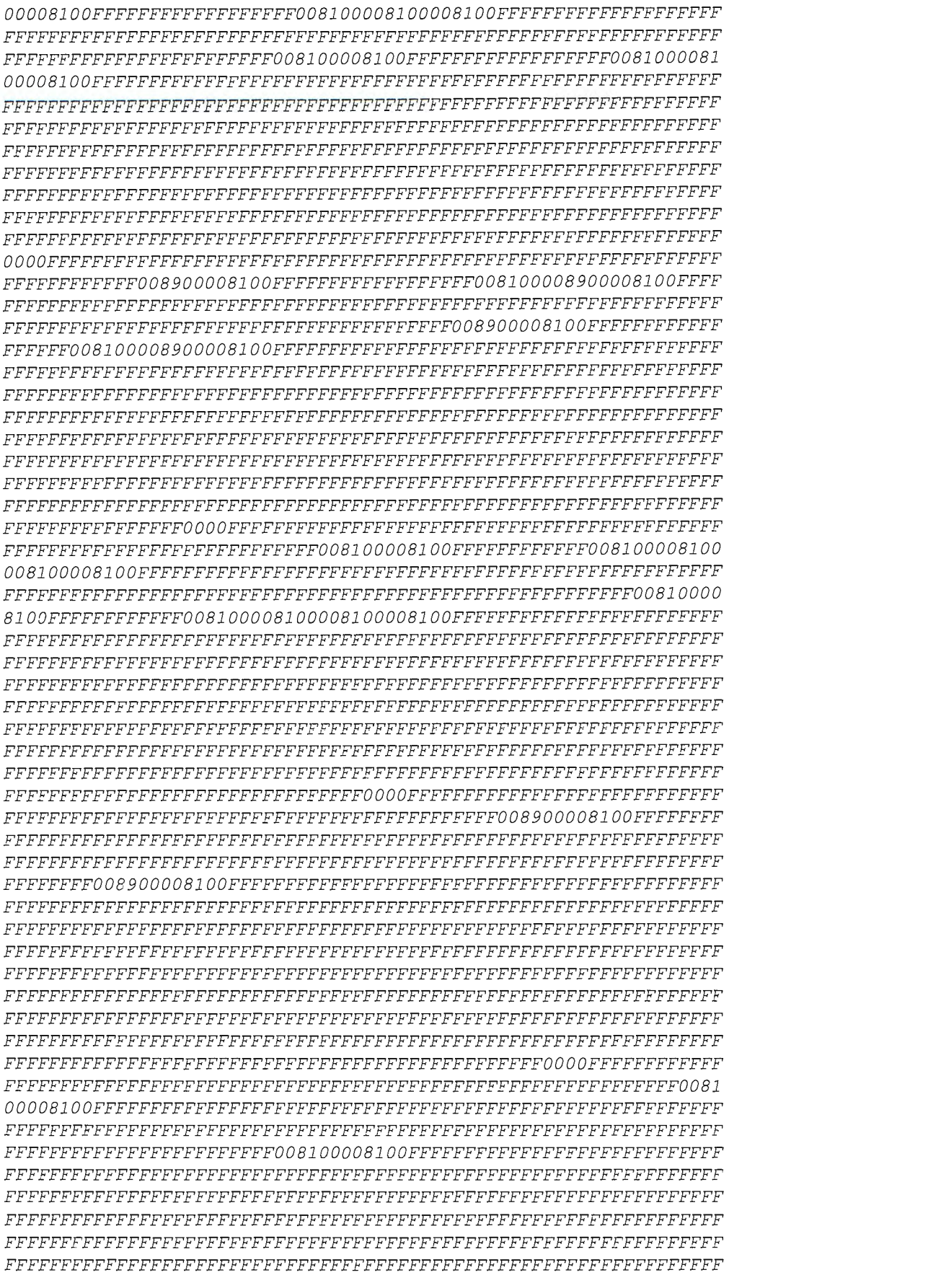
```
Width = 26
Left = 152
Top = 200
Bitmap = {
  494D474C010001001A0000001A00000012000000424D769F0000000000003600
  00002800000082000000680000000100180000000000409F0000120B0000120B
  00000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFF008900008100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008900008100FFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
  FFFFFFFFFFFFFFFFFFFFFFFFFFFF008100008100FFFFFFFFFFFFFFFFFFFFFFFF
```

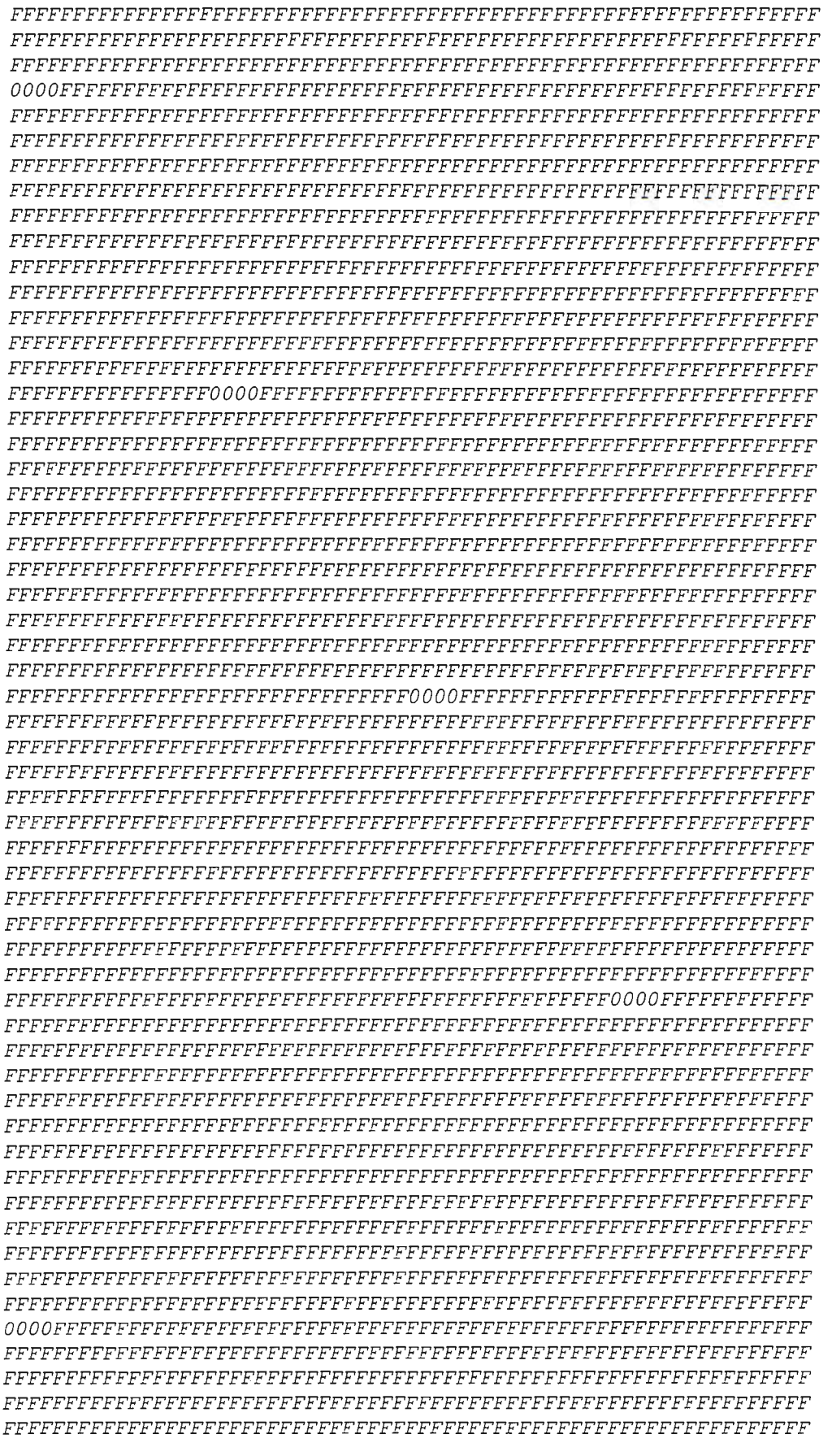
Source: CLXMain.dfm

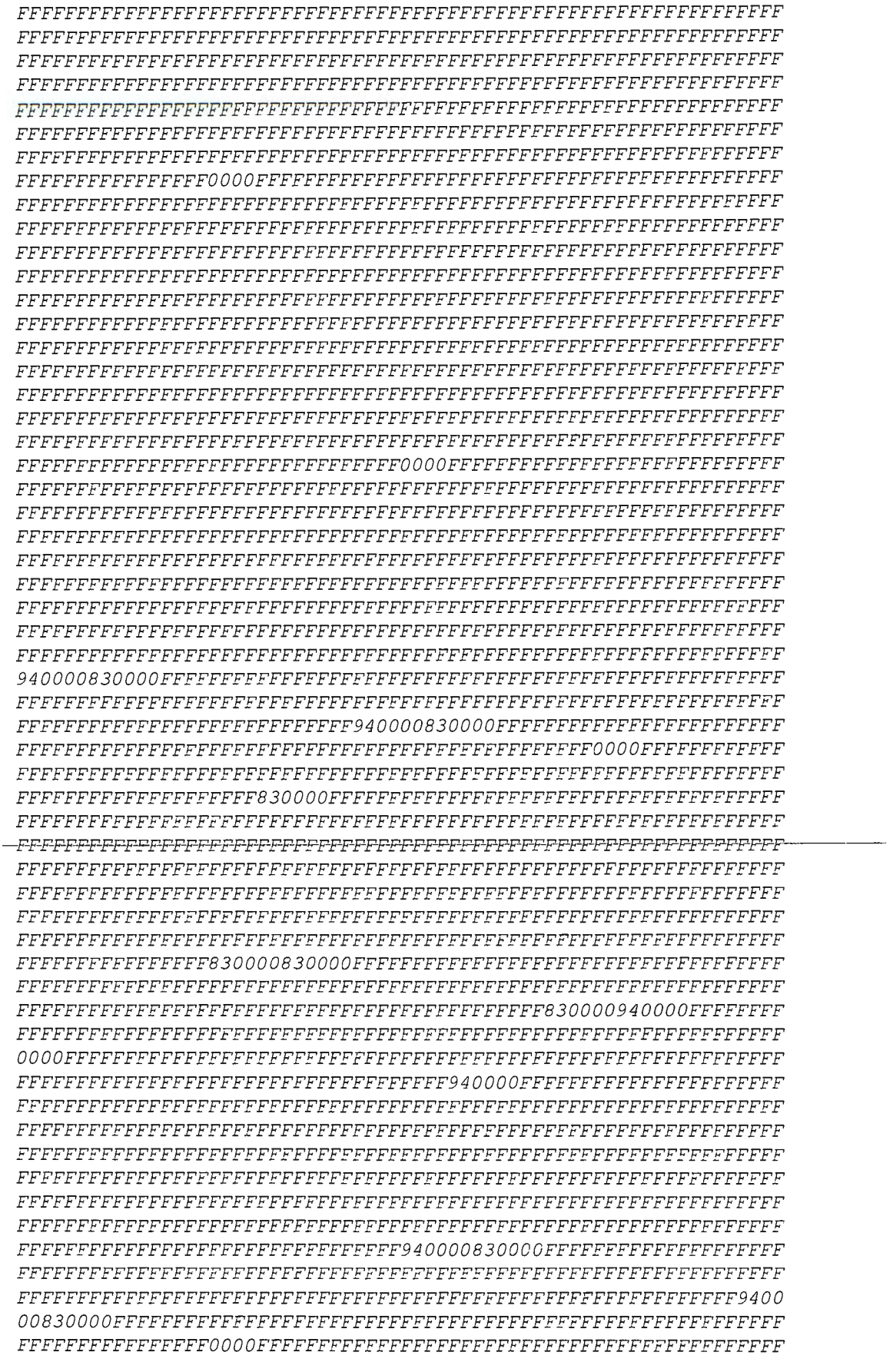
```
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00810000
8100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF008900FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008900008100FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008900FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF008900008100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF008100008100FFFFFFFFFFFFFFFFFFFFFFFF0081
00008100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008100008100
FFFFFFFFFFFFFFFFFFFFFFFF008100008100FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008100008900FFFFFF
008900008100008900008100008900FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF008100008900FFFFFF008900008100008900008100008900FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF008100008100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008100FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008100008100FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF008100FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

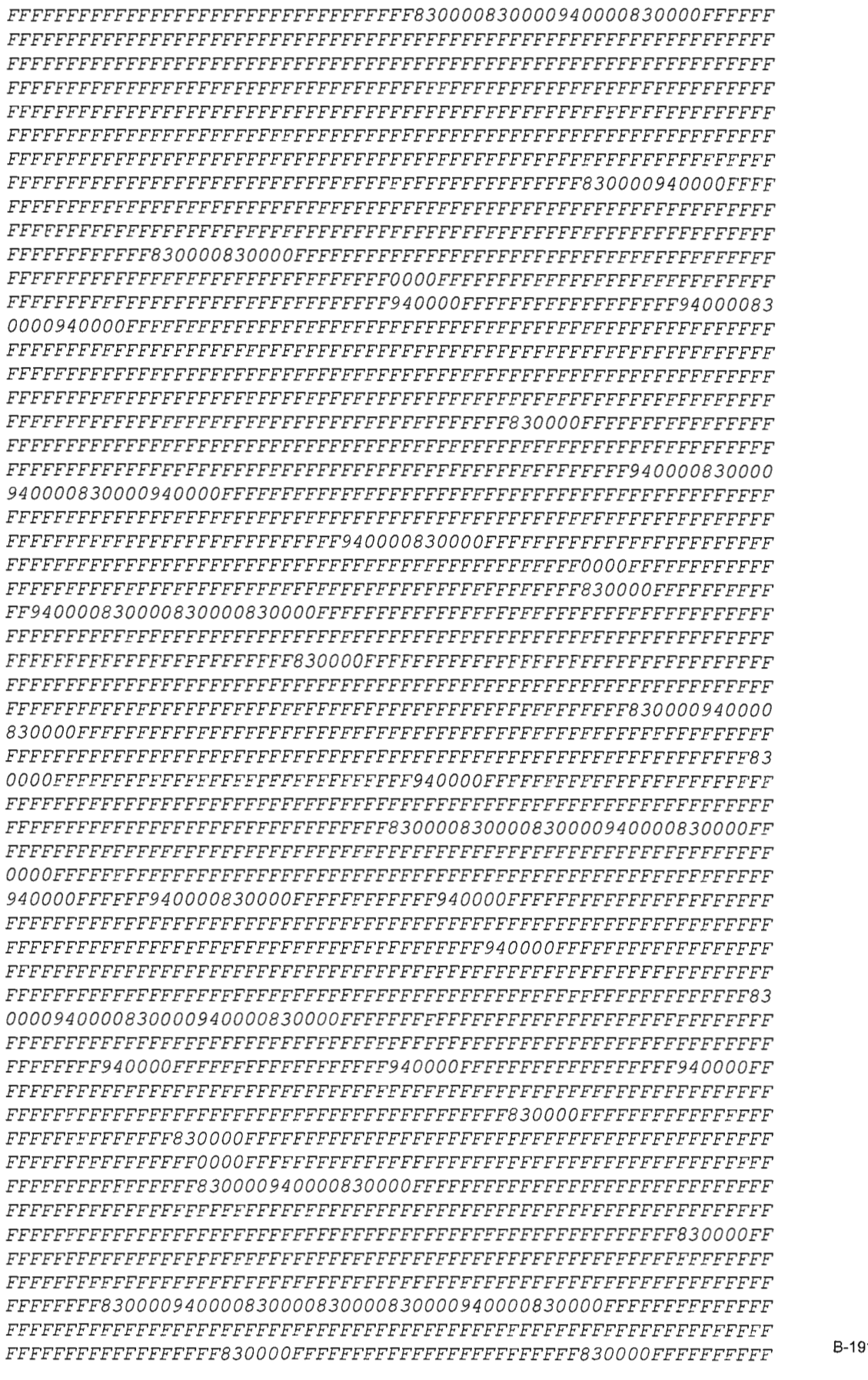


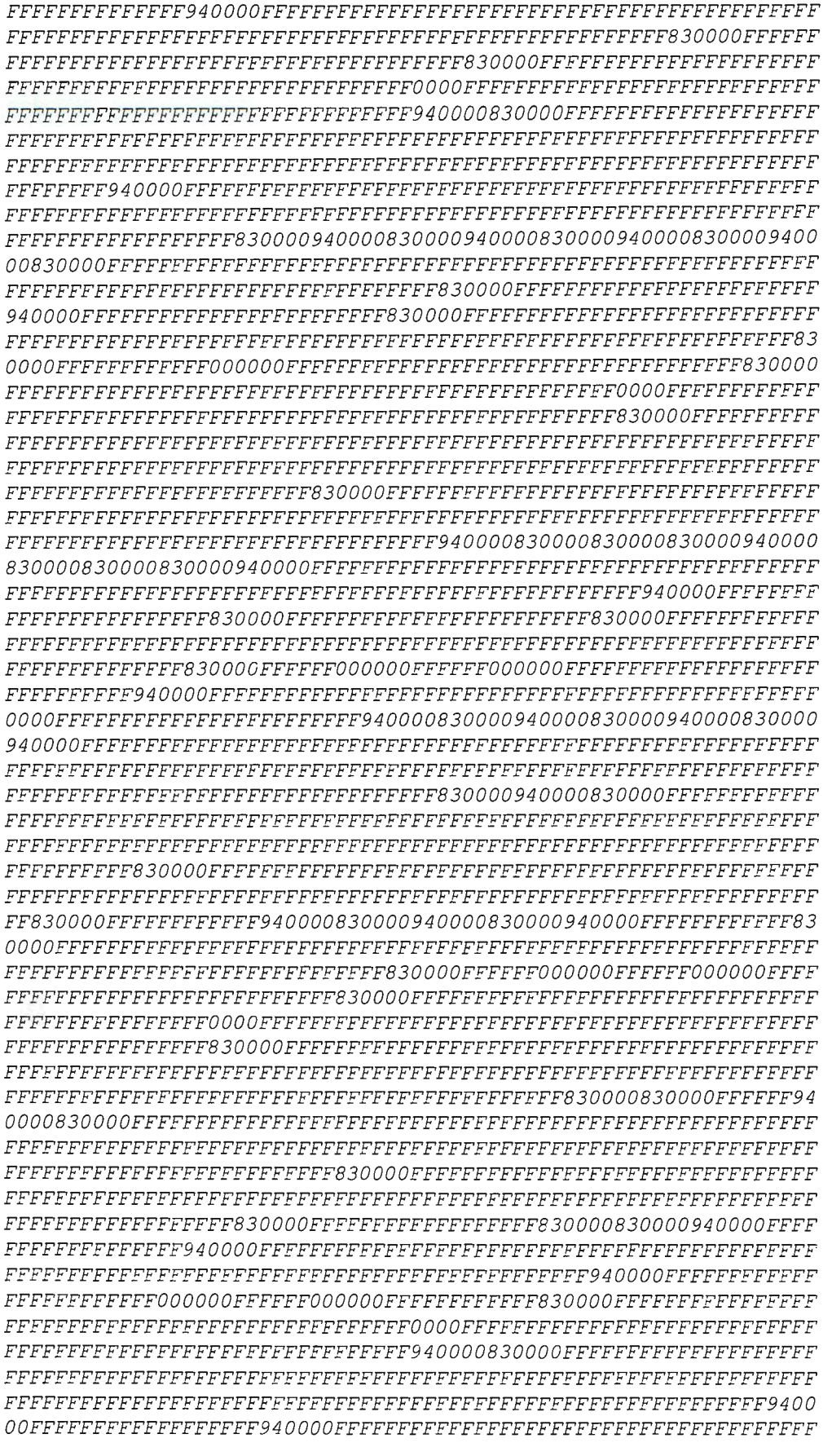




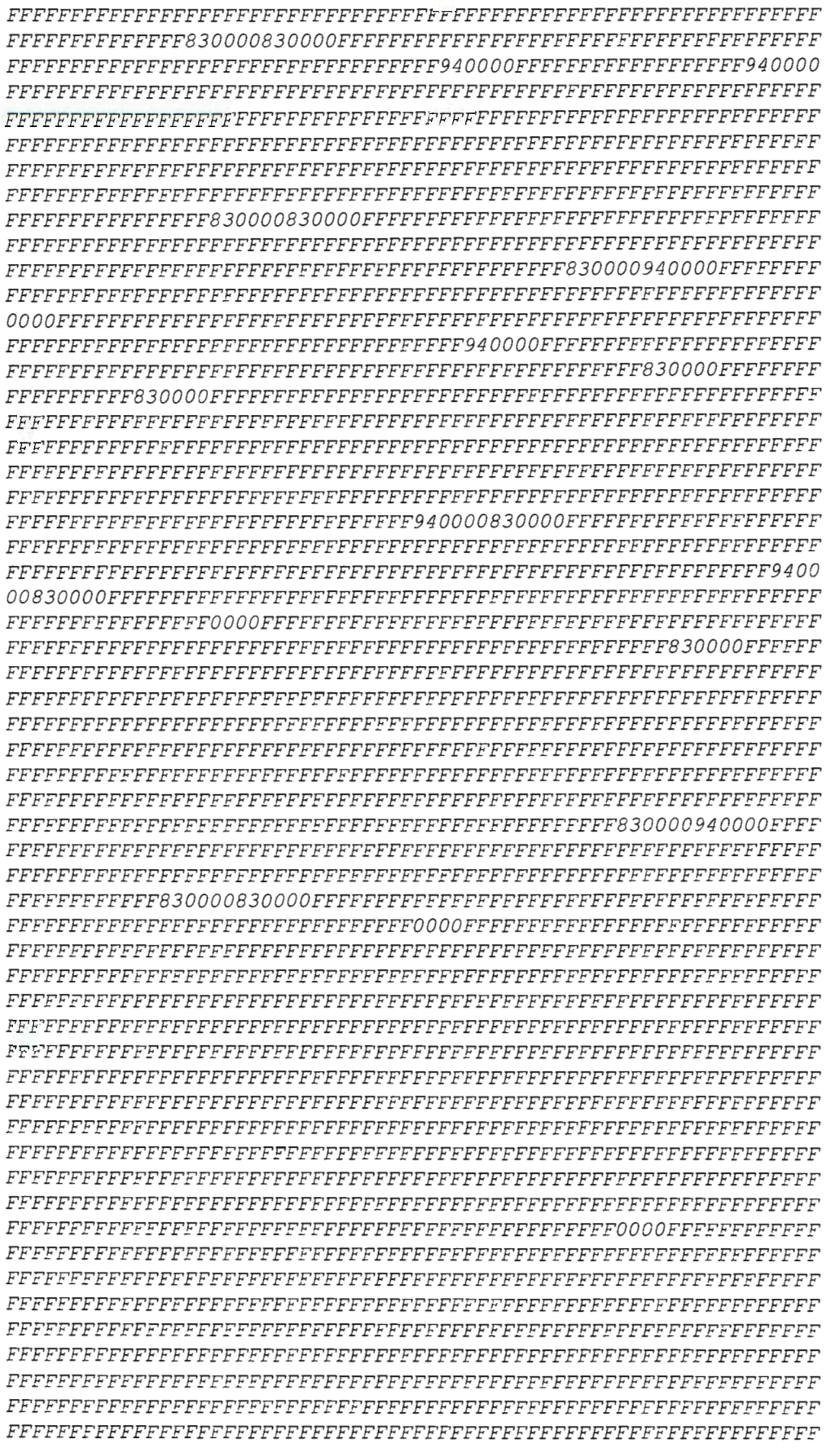


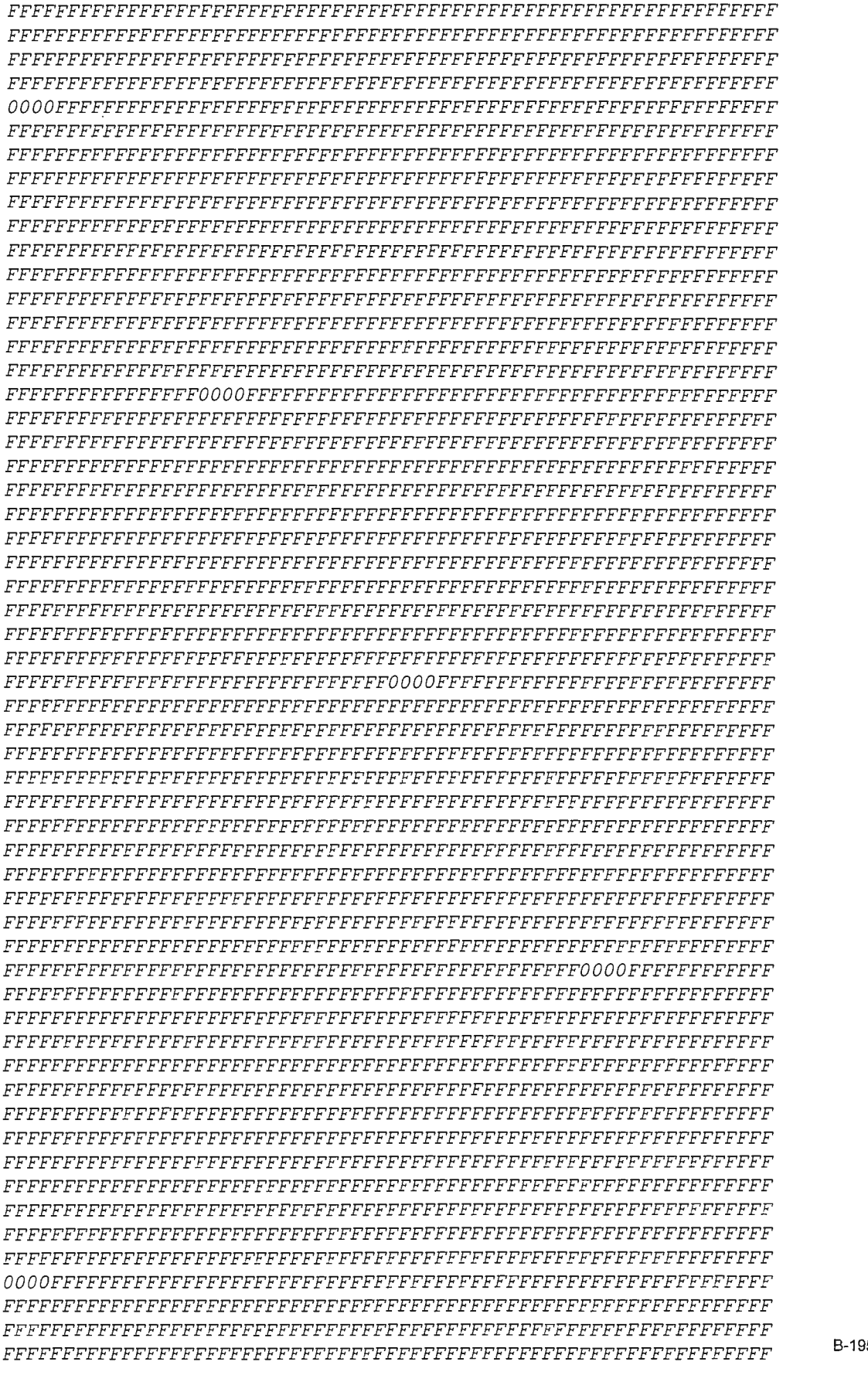


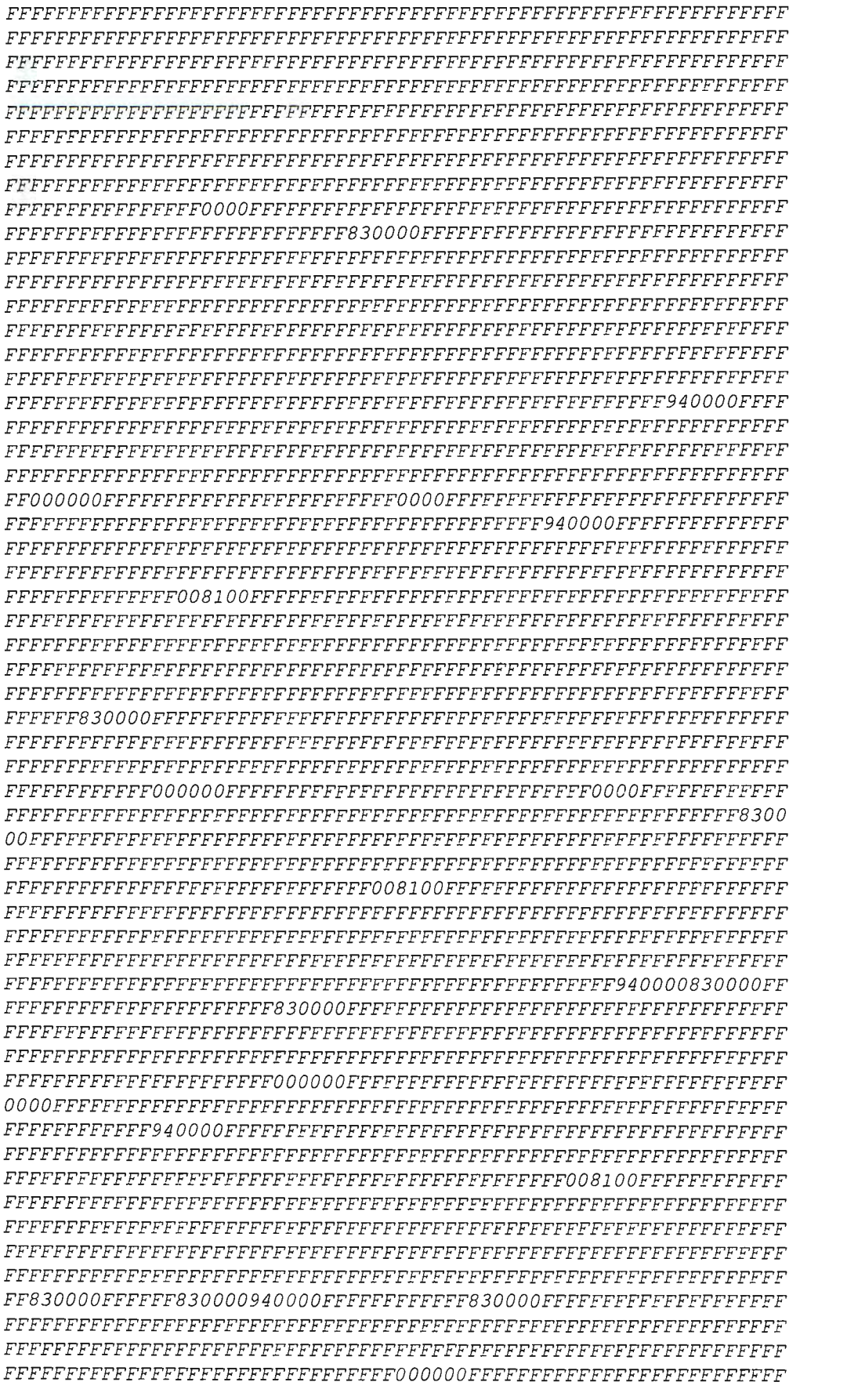


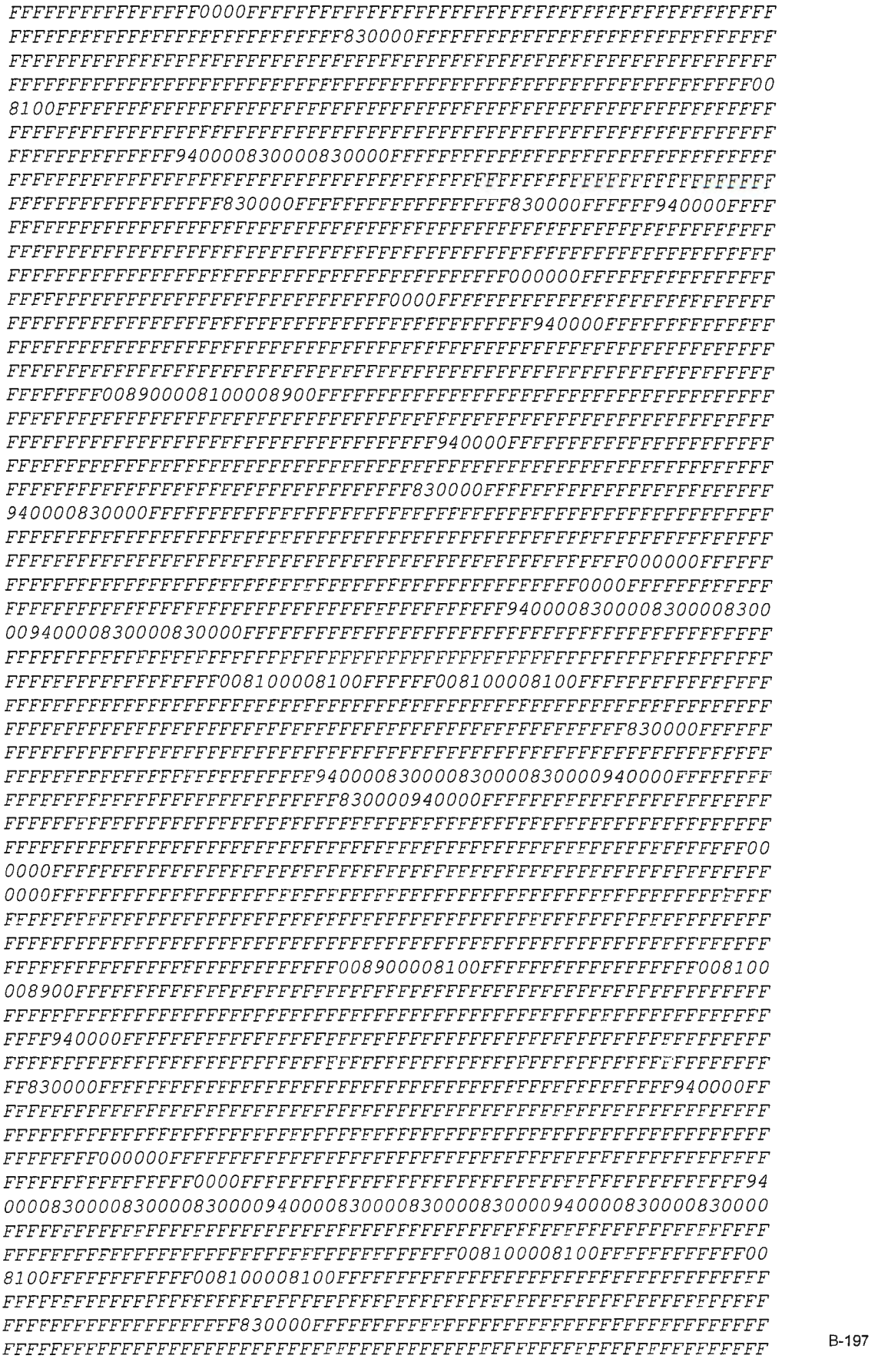


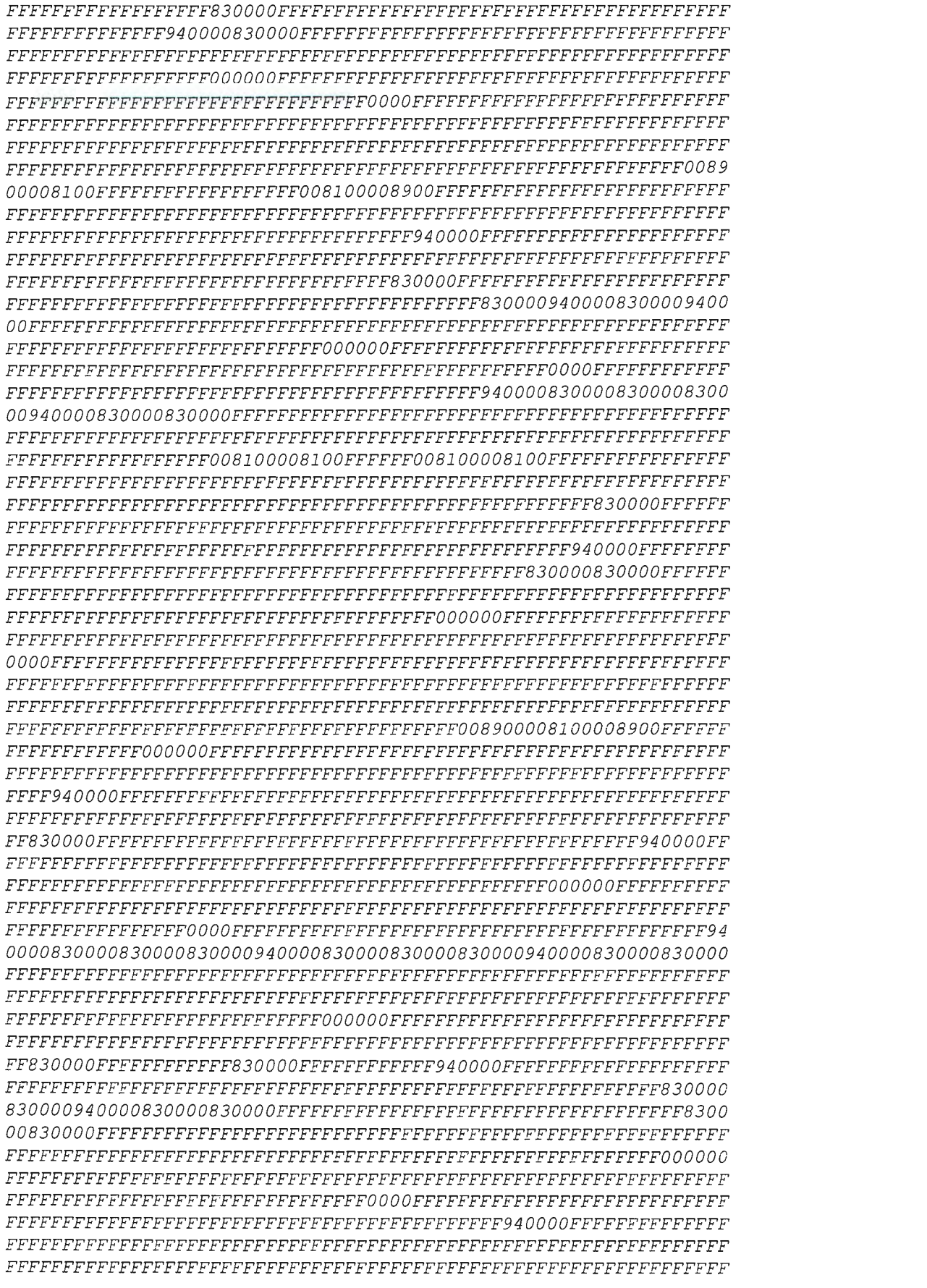
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFF
940000FFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFF000000FFFFFFFFFFFF830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8300008300008300
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFF
FFFFFFFFFFFF830000940000FFFFFF830000830000FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFF
FFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
940000FFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF83
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000940000830000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF940000830000940000830000940000FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF830000FFFFFFFFFFFF830000830000FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF830000FFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000940000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
830000940000830000830000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFF94000083
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFF940000FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFF

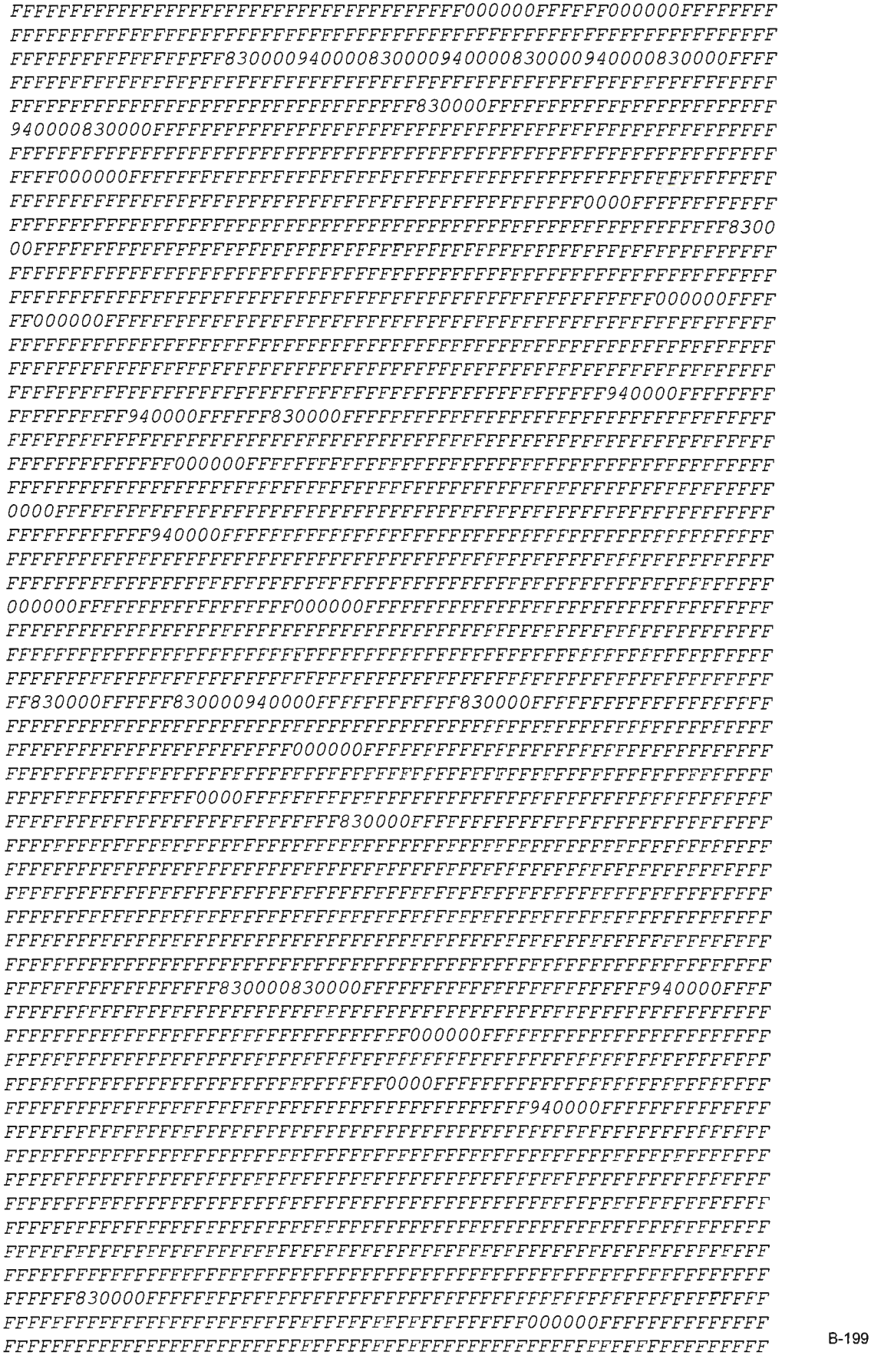


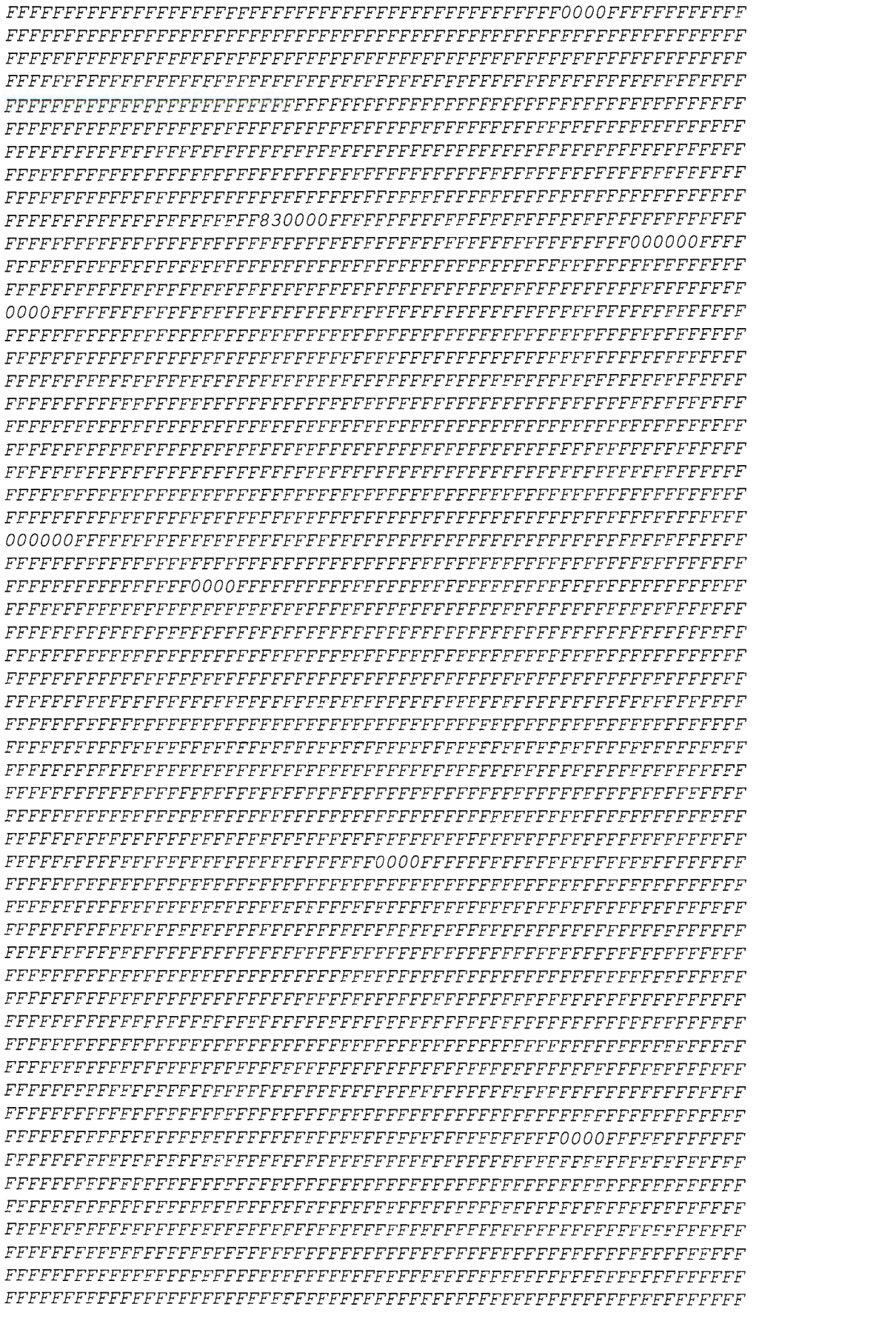


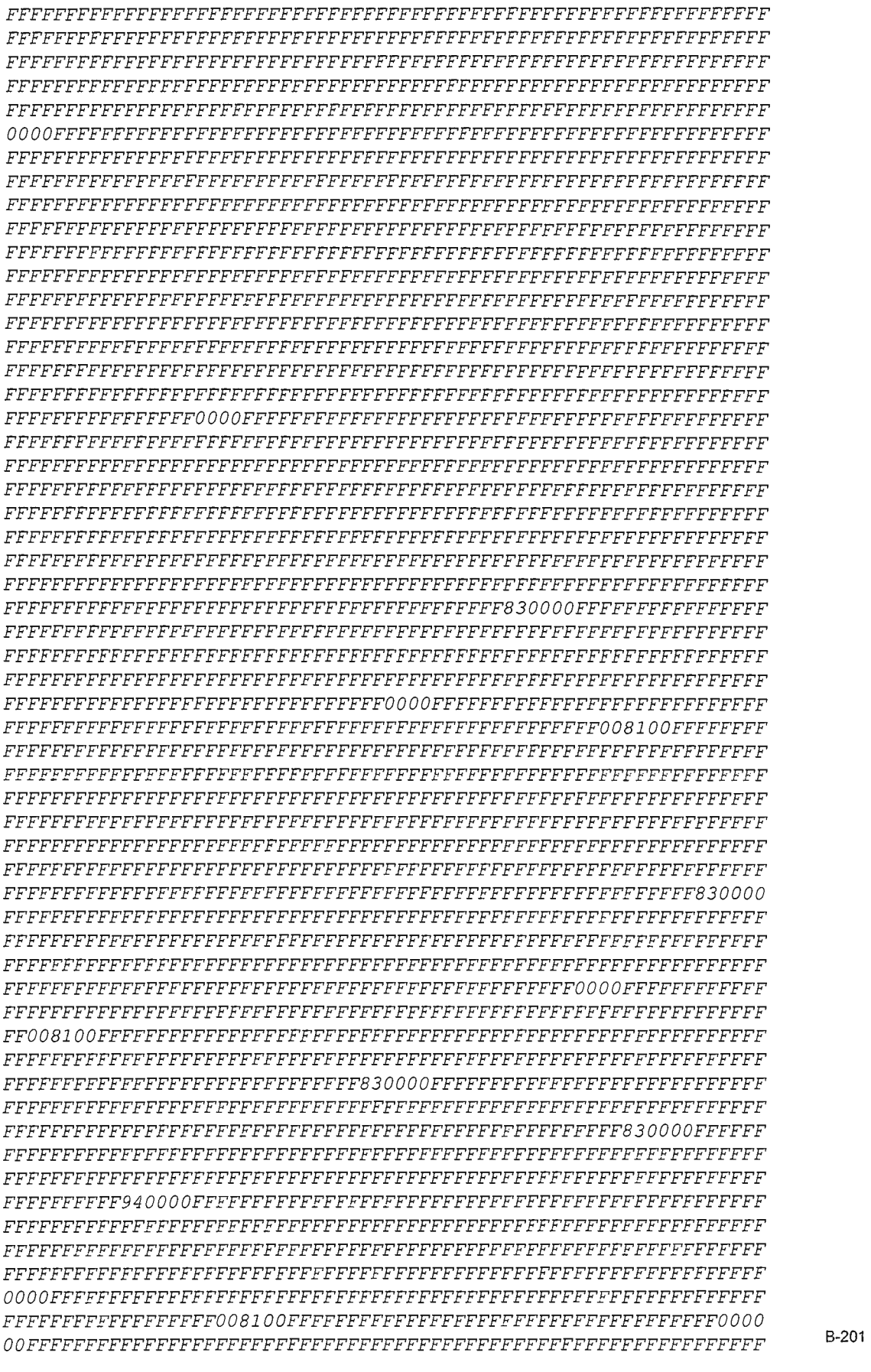




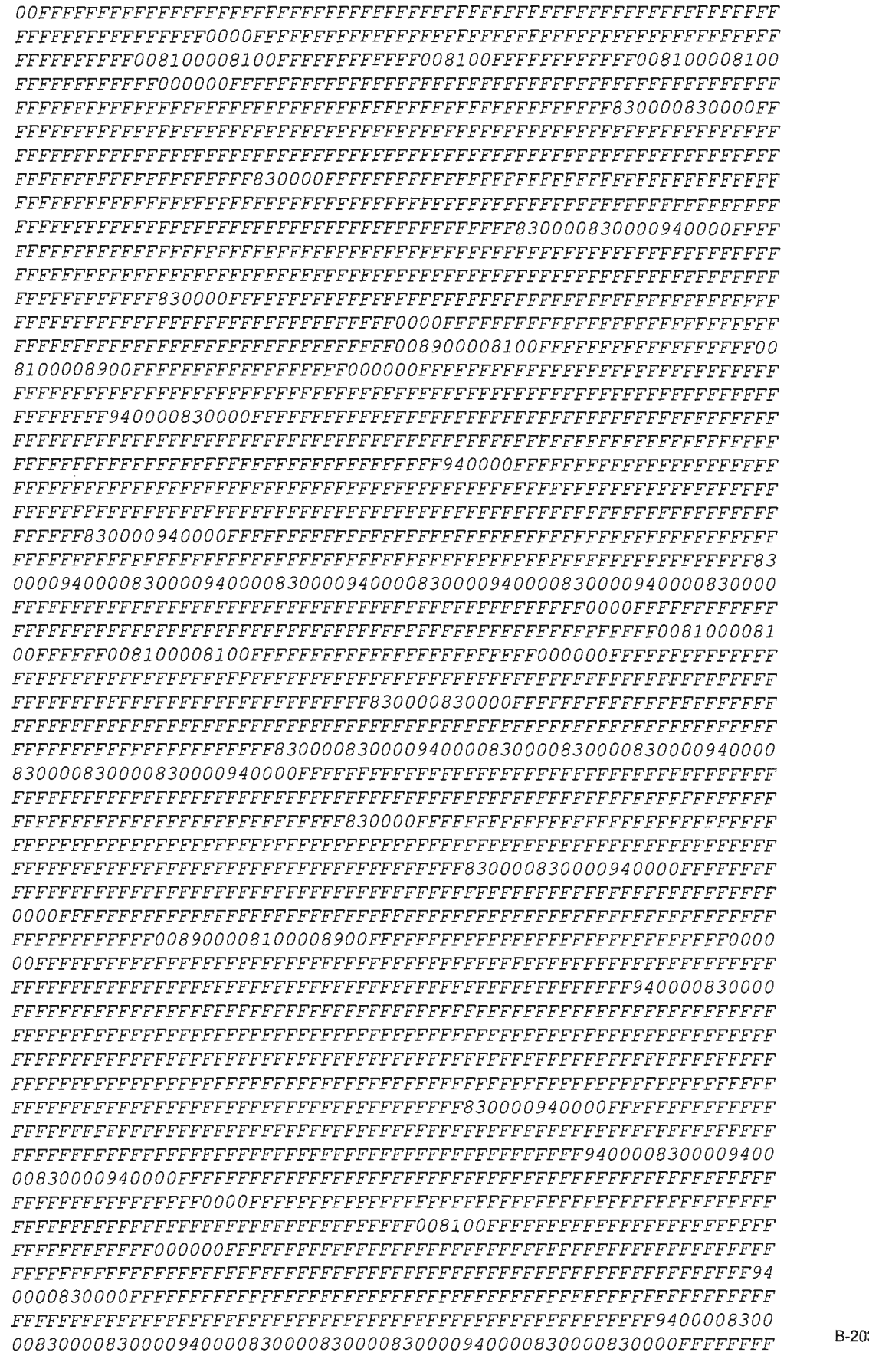

FFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0089
00008100FFFFFFFFFFFFFFFFFF008100008900FFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000940000830000940000
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000830000830000830000
009400008300008300 00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF008100008100FFFFFF008100008100FFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8300008300000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008900008100008900FFFFFF
FFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF94
0000830000830000830000940000830000830000830000940000830000830000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FF830000FFFFFFFFFFFFF830000FFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
8300009400008300008300 00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8300
00830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF000000
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

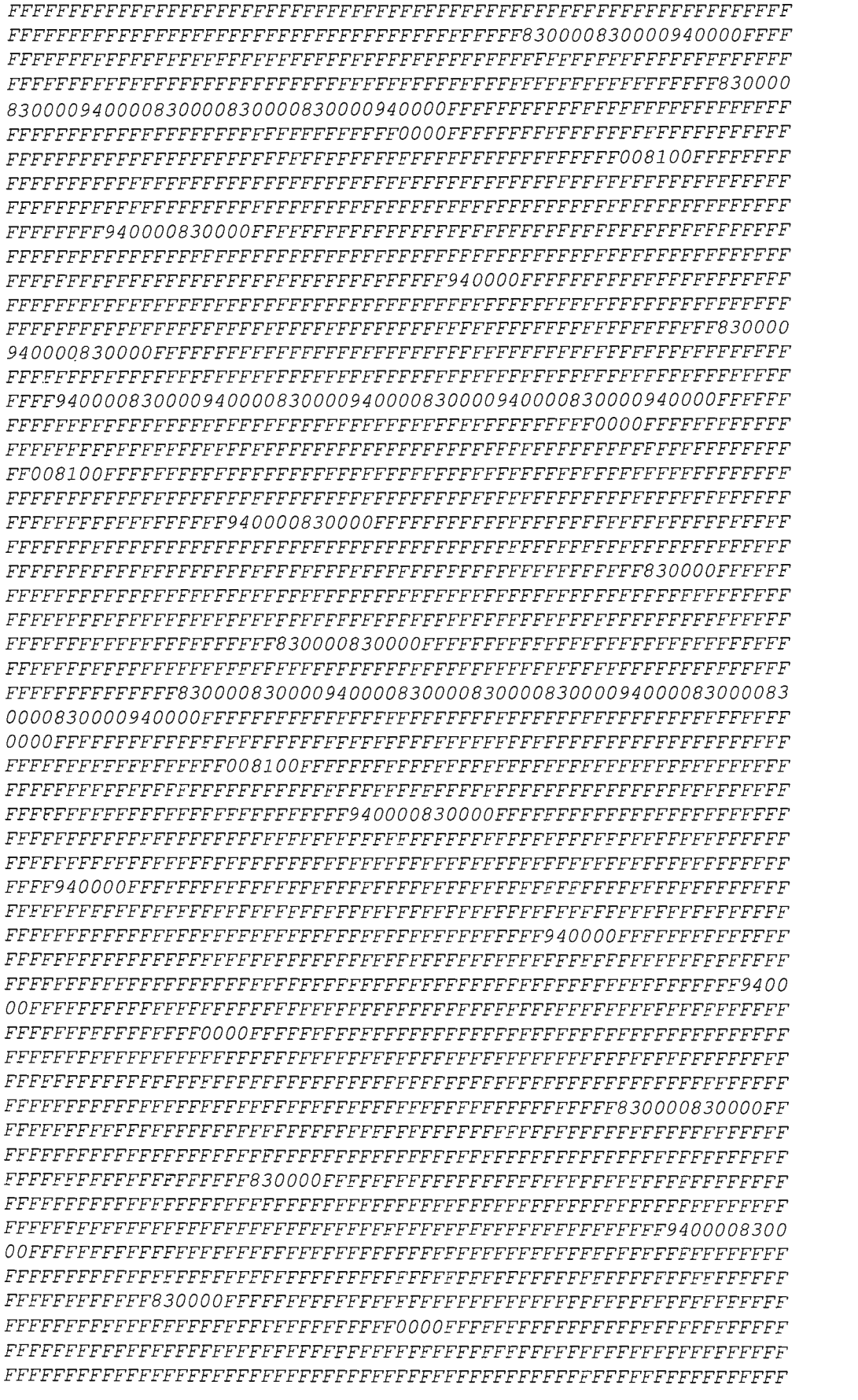






FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF830000940000830000FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9400
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008100FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF000000000000000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF94
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9400008300
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF008900008100008900FF
FFFFFFFFFFFFFFFF000000000000000000000000000000FFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0081000081
00FFFFFF008100008100FFFFFFFFFFFFFFFFFFFFFFFF000000FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFF830000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
008900008100FFFFFFFFFFFFFFFFFF008100008900FFFFFFFFFFFFFFFFFF0000
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF830000940000830000FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9400





Source: CLXMain.dfm

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000
940000830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFF940000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF9400
00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF94
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF830000FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF830000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

B-205

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0000FFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF0000424D5E080000000000003E0000002800000082000000
680000000100010000000000200800001 20B0000120B00000200000002000000
FFFFFF000000000000000000000000000003FFFFFFFFFFFFFFFFFFFF00000000
000000000003FFFFFFFFFFFFFFFFFFFF00000000000000000003FFFFFFFFFFFF
FFFFFFFF000C0000030000000003FFFFFFFFFFFFFFFFFFFF000C000003000000
0003FFFFFFFFFFFFFFFFFFFF020C0000830000000003FFFFFFFFFFFFFFFFFFFF
030C0000C30000000003FFFFFFFFFFFFFFFFFFFF01BE00006F8000000003FFFF
FFFFFFFFFFFFFFFF00C10000304000000003FFFFFFFFFFFFFFFFFFFF00888000
202000000003FFFFFFFFFFFFFFFFFFFF010840004F9000000003FFFFFFFFFFFF
FFFFFFFF01144000421000000003FFFFFFFFFFFFFFFFFFFF0114400042100000
0003FFFFFFFFFFFFFFFFFFFF0122400042100001C003FFFFFFFFFFFFFFFFFFFF
012240004F900003E003FFFFFFFFFFFFFFFFFFFF0080800020200003E003FFFF
FFFFFFFFFFFFFFFF0041800010600003E003FFFFFFFFFFFFFFFFFFFF003ED000
0FB40001C003FFFFFFFFFFFFFFFFFFFF000C7000031C00000003FFFFFFFFFFFF
FFFFFFFF000C7000031C00000003FFFFFFFFFFFFFFFFFFFF000CF000033C0000
0003FFFFFFFFFFFFFFFFFFFF000C0000030000000003FFFFFFFFFFFFFFFFFFFF
000C0000030000000003FFFFFFFFFFFFFFFFFFFF00000000000000000003FFFF
FFFFFFFFFFFFFFFF00000000000000000003FFFFFFFFFFFFFFFFFFFF00000000
000000000003FFFFFFFFFFFFFFFFFFFF00000000000000000000000000000000
3FFFFFFF000000000000000000000000000000003FFFFFFF0000000000000000
00000000000000003FFFFFFF00000000000000000000003000000C003FFFFFFF
00008000000000000000003000000C003FFFFFFF000080000000000000000030
00000C003FFFFFFF00078000000000000000003000000C003FFFFFFF00238000
00000000400000F800000C003FFFFFFF0027800002000000E000010400003E00
3FFFFFFF002C800002000001F0000222000041003FFFFFFF0038000002000003
F8000421000080803FFFFFFF0030000002000007FC000421000120403FFFFFFF
0020000002000007FC000421000150403FFFFFFF0FE0000007000000400004F9
000150403FFFFFFF002000000D8000004000047100010A403FFFFFFF00300000
088000004000022200010A403FFFFFFF003800010D8000004000010400008480
3FFFFFFF002C000107000000000000F8000041003FFFFFFF0026000280000000
0000003000003E003FFFFFFF00230002800000000000003000000C003FFFFFFF

```
      0001800440000000000000300000C003FFFFFFF000080044000000000000030
      00000C003FFFFFFF000080000000000000003000000C003FFFFFFF00000000
      0000000000000000000000003FFFFFFF000000000000000000000000000000000
      3FFFFFFF000000000000000000000000000000003FFFFFFF0000000000000000
      00000000000000003FFFFFFF000000000000000000000000000003FFFFFFF
      0000000000000000000000000000003FFFFFFF000000000000000000000000
      000000003FFFFFFF000800000000000000000100000000043FFFFFFF00080000
      0100000000000010000000083FFFFFFF000800000100000000000610000000010
      3FFFFFFF000800000100000000000590000000203FFFFFFF0008000001000001
      C000045000000040 3FFFFFFF000800000380000080000430000000803FFFFFFF
      007F000006C0000080007C0C000001003FFFFFFF000000000C60000080000402
      000002003FFFFFFF01FFC000193000008000040180000400 3FFFFFFF00000000
      0C60000080000400780008003FFFFFFF007F000006C000008000040180001000
      3FFFFFFF000000000388000080000402000020003FFFFFFF01FFC00000080004
      90007C0C000040003FFFFFFF000800000014000 7F000043000008000 3FFFFFFF
      000800000014000000000450000100003FFFFFFF000800000022000000000590
      000200003FFFFFFF000800000000000000000061000040000 3FFFFFFF00080000
      0000000000000010000800003FFFFFFF000000000000000000000001000100000
      3FFFFFFF000000000000000000000000002000003FFFFFFF0000000000000000
      00000000000000003FFFFFFF000000000000000000000000000000003FFFFFFF
      0000000000000000000000000000000003FFFFFFF000000000000000000000000
      000000003FFFFFFF000000000000000000000004000000003FFFFFFF00040000
      0000000000000040000000003FFFFFFF000400000100000080000040000000000
      3FFFFFFF000408000100000080000070000008003FFFFFFF00041C0001000000
      8000001800000 8003FFFFFFF000E3E000300000080000008000008003FFFFFFF
      001B080006000000800000180000008003FFFFFFF003188000C000000800000 70
      000008003FFFFFFF0064C8000600000080000070000008003FFFFFFF00318800
      0300000080000018000 1FFC03FFFFFFF001B08000180001FFC00000800001C00
      3FFFFFFF000E080000C000000000001800003E003FFFFFFF000408000180001F
      FC00007000007F003FFFFFFF000400000300000080000070000 0FF803FFFFFFF
      000400000600000080000018000 1FFC03FFFFFFF000400000C000000800000 8
      000008003FFFFFFF000000000600000080000018000008003FFFFFFF00000000
      0300000080000070000008003FFFFFFF000000000100000080000040000008 00
      3FFFFFFF000000000100000080000040000008003FFFFFFF0000000001000000
      0000004000000 8003FFFFFFF000000000000000000000000000000003FFFFFFF
      0000000000000000000000000000000003FFFFFFF000000000000000000000000
      000000003FFFFFFF}
  end
end
```

```
object FrmSetup: TFrmSetup
  Left = 232
  Top = 147
  Width = 367
  Height = 272
  VertScrollBar.Range = 265
  HorzScrollBar.Range = 361
  ActiveControl = Edit1
  Caption = 'Configurar An'#225'lisis'
  Color = clBackground
  Icon.Data = {
    0000010001002020100000000000E80200001600000028000000200000004000
    0000010004000000000008002000000000000000000000000000000000000000
    0000000080000080000000808000800000008000800080800000C0C0C0008080
    80000000FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF000000
    0000000000000000000000000000000000000000000000000000000000000888
    8888888888888888888888888800000000000000000000000000000080F88
    888888888888888888888888080F7F7F7F7F7F7F7F7F7F7F7F7F78080FF7
    F7F788888888888888F7F7F7F8080F7F7F70000000000000087F7F7F78080FF7
    F7F0FFFFFFFFFFFF08F7F7F7F8080F7F7F70FFFFFFFFFFFF087F7F7F78080FF7
    F000FFFFFFFFFFFF08F7F7F7F8080F7F70F0FFFFFFFFFFFF087F7F7F78080FF7
    F0F0FFFFFFFFFFFF08F7F7F7F8080F7000F0CCCCCC070707087F7F7F78080FF0
    F0F000000000000007F7F7F7F8080F70F0FFFFFFFFFFFF087F7F7F7F78080FF0
    F0CCCCCC07070708F7F7F7F7F8080F70F00000000000000F7F7F7F7F78080FF0
    FFFFFFFFFFFF08F7F7F7F7F7F8080F70CCCCCC070707087F7F7F7F7F78080FF0
    00000000000007F7F7F7F7F7F8080FFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080F7F7F7F7F7F7F7F7F7F7F7F7F7F080F00
    07F000F70007F7F7F7F7F7F7F7080FFFFFFFFFFFFFFFFFFFFFFFFFFFFF080000
    0000000000000000000000000080CCCCCCCCCCCCCCCCCCCCCCC070707080000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000FFFF
    FFFFFFFFFFFF8000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000001FFFFFFFFFFFFFFFFFFFFFFFF}
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
  TextWidth = 6
  object GroupBox2: TGroupBox
    Left = 24
    Top = 104
    Width = 257
    Height = 73
    TabOrder = 5
    Visible = False
    object Label7: TLabel
      Left = 16
      Top = 16
      Width = 40
      Height = 13
```

```
      Caption = 'timestep'
    end
    object Label9: TLabel
      Left = 136
      Top = 16
      Width = 54
      Height = 13
      Caption = 'tiempo final'
    end
    object Edit6: TEdit
      Left = 16
      Top = 32
      Width = 105
      Height = 21
      TabOrder = 2
    end
    object Edit8: TEdit
      Left = 136
      Top = 32
      Width = 105
      Height = 21
      TabOrder = 3
    end
  end
  object GroupBox1: TGroupBox
    Left = 8
    Top = 8
    Width = 353
    Height = 257
    TabOrder = 0
    object Label1: TLabel
      Left = 16
      Top = 96
      Width = 7
      Height = 13
      Caption = 'n'
    end
    object Label2: TLabel
      Left = 16
      Top = 136
      Width = 6
      Height = 13
      Caption = 's'
    end
    object Label3: TLabel
      Left = 16
      Top = 176
      Width = 7
      Height = 13
      Caption = 'k'
    end
    object Edit1: TEdit
```

```
    Left = 48
    Top = 96
    Width = 113
    Height = 21
    TabOrder = 3
  end
  object Edit2: TEdit
    Left = 48
    Top = 136
    Width = 113
    Height = 21
    TabOrder = 4
  end
  object Edit3: TEdit
    Left = 48
    Top = 176
    Width = 113
    Height = 21
    TabOrder = 5
  end
  object Label4: TLabel
    Left = 16
    Top = 216
    Width = 19
    Height = 13
    Caption = 'freq'
  end
  object Edit4: TEdit
    Left = 48
    Top = 216
    Width = 113
    Height = 21
    TabOrder = 7
  end
  object Edit5: TEdit
    Left = 192
    Top = 96
    Width = 153
    Height = 21
    TabOrder = 8
  end
  object Label5: TLabel
    Left = 192
    Top = 80
    Width = 63
    Height = 13
    Caption = 'Array of Ports'
  end
  object Label6: TLabel
    Left = 192
    Top = 152
    Width = 153
```

```
    Height = 57
    AutoSize = False
    Caption =
      'n = numero de puertos,                s = numero de fuentes,     ' +
      '          k = numero de armonicos,          freq = frecuencia fu' +
      'ndamental.'
    Font.Color = clBlack
    Font.Height = 11
    Font.Name = 'MS Shell Dlg'
    Font.Pitch = fpFixed
    Font.Style = []
    Font.Weight = 40
    ParentFont = False
    WordWrap = True
  end
end
object RadioGroup1: TRadioGroup
  Left = 24
  Top = 16
  Width = 161
  Height = 65
  Items.Strings = (
    'Transitivo'
    'Balance Arm'#243'nico')
  Caption = 'An'#225'lisis'
  Font.Color = clBlack
  Font.Height = 11
  Font.Name = 'MS Shell Dlg'
  Font.Pitch = fpVariable
  Font.Style = []
  Font.Weight = 40
  ItemIndex = 1
  ParentFont = False
  TabOrder = 1
  OnClick = RadioGroup1Click
end
object CheckBox1: TCheckBox
  Left = 200
  Top = 16
  Width = 101
  Height = 31
  Caption = 'Recrear Netlist'
  Checked = True
  State = cbChecked
  TabOrder = 2
end
object Button1: TButton
  Left = 200
  Top = 224
  Width = 65
  Height = 33
  Caption = 'Aceptar'
  TabOrder = 3
```

```
    OnClick = Button1Click
  end
  object Button2: TButton
    Left = 272
    Top = 224
    Width = 65
    Height = 33
    Caption = 'Cancelar'
    TabOrder = 4
    OnClick = Button2Click
  end
end
```

Source: CadLayer.dfm

```
object LayersDialog: TLayersDialog
  Left = 269
  Top = 190
  Width = 383
  Height = 224
  HorzScrollBar.Range = 189
  ActiveControl = ListBox1
  Caption = 'Layers'
  Color = clButton
  Font.Color = clText
  Font.Height = 11
  Font.Name = 'MS Sans Serif'
  Font.Pitch = fpVariable
  Font.Style = []
  Font.Weight = 40
  ParentFont = False
  OnClose = FormClose
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
  TextWidth = 6
  object Splitter1: TSplitter
    Left = 185
    Top = 0
    Width = 4
    Height = 224
  end
  object Panel1: TPanel
    Left = 0
    Top = 0
    Width = 185
    Height = 224
    Align = alLeft
    BevelOuter = bvNone
    Caption = 'Panel1'
    TabOrder = 0
    object Panel2: TPanel
      Left = 0
      Top = 0
      Width = 185
      Height = 17
      Align = alTop
      Alignment = taLeftJustify
```

```
      Caption = ' Combinations'
      TabOrder = 0
    end
    object Panel3: TPanel
      Left = 0
      Top = 17
      Width = 185
      Height = 207
      Align = alClient
      Caption = 'Panel3'
      TabOrder = 1
      object ListBox1: TListBox
        Left = 1
        Top = 1
        Width = 183
        Height = 205
        Align = alClient
        BorderStyle = bsNone
        ItemHeight = 13
        PopupMenu = PopupMenu1
        Sorted = True
        TabOrder = 0
        OnClick = ListBox1Click
        OnDragDrop = ListBox1DragDrop
        OnDragOver = ListBox1DragOver
      end
    end
  end
  object Panel7: TPanel
    Left = 189
    Top = 0
    Width = 194
    Height = 224
    Align = alClient
    BevelOuter = bvNone
    Caption = 'Panel1'
    TabOrder = 1
    object Panel8: TPanel
      Left = 0
      Top = 0
      Width = 194
      Height = 17
      Align = alTop
      Alignment = taLeftJustify
```

```
      Caption = ' Layers'
      TabOrder = 0
    end
    object Panel9: TPanel
      Left = 0
      Top = 17
      Width = 194
      Height = 207
      Align = alClient
      Caption = 'Panel3'
      TabOrder = 1
      object ListBox2: TListBox
        Left = 1
        Top = 1
        Width = 192
        Height = 205
        Align = alClient
        BorderStyle = bsNone
        DragMode = dmAutomatic
        ItemHeight = 13
        PopupMenu = PopupMenu2
        Sorted = True
        TabOrder = 0
      end
    end
  end
  object PopupMenu1: TPopupMenu
    Left = 97
    Top = 89
    object Delete1: TMenuItem
      Caption = 'Delete'
      OnClick = Delete1Click
    end
    object Edit1: TMenuItem
      Caption = 'Edit'
      OnClick = Edit1Click
    end
    object New1: TMenuItem
      Caption = 'New'
      OnClick = New1Click
    end
  end
  object PopupMenu2: TPopupMenu
    Left = 254
```

```
      Top = 89
      object Delete2: TMenuItem
        Caption = 'Delete'
        OnClick = Delete2Click
      end
      object Edit2: TMenuItem
        Caption = 'Edit'
        OnClick = Edit2Click
      end
      object New2: TMenuItem
        Caption = 'New'
        OnClick = New2Click
      end
    end
  end
```

Source: CadTree.dfm

```
object TreeDialog: TTreeDialog
  Left = 192
  Top = 118
  Width = 166
  Height = 167
  Caption = 'Tree'
  Color = clBackground
  OnClose = FormClose
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
  TextWidth = 6
  object TreeView1: TTreeView
    Left = 0
    Top = 0
    Width = 166
    Height = 167
    Align = alClient
    Columns = <>
    TabOrder = 0
  end
end
```

C/C++ Source Files

/

- fftn.cpp
- fftn.h
- hbdemo.cpp
- hbdemo.h
- main.cpp
- matrix.h

```
/*---------------------------------*-C-*---------------------------------*
 * File:
 *      fftn.c
 *
 * Public:
 *      fft_free ();
 *      fftn / fftnf ();
 *
 * Private:
 *      fftradix / fftradixf ();
 *
 * Descript:
 *      multivariate complex Fourier transform, computed in place
 *      using mixed-radix Fast Fourier Transform algorithm.
 *
 *      Fortran code by:
 *      RC Singleton, Stanford Research Institute, Sept. 1968
 *
 *      translated by f2c (version 19950721).
 *
 * Revisions:
 * 26 July 95   John Beale
 *      - added maxf and maxp as parameters to fftradix()
 *
 * 28 July 95   Mark Olesen <olesen@me.queensu.ca>
 *      - cleaned-up the Fortran 66 goto spaghetti, only 3 labels remain.
 *
 *      - added fft_free() to provide some measure of control over
 *        allocation/deallocation.
 *
 *      - added fftn() wrapper for multidimensional FFTs
 *
 *      - use -DFFT_NOFLOAT or -DFFT_NODOUBLE to avoid compiling that
 *        precision. Note suffix `f' on the function names indicates
 *        float precision.
 *
 *      - revised documentation
 *
 * 31 July 95   Mark Olesen <olesen@me.queensu.ca>
 *      - added GNU Public License
 *      - more cleanup
 *      - define SUN_BROKEN_REALLOC to use malloc() instead of realloc()
 *        on the first pass through, apparently needed for old libc
 *      - removed #error directive in favour of some code that simply
 *        won't compile (generate an error that way)
 *
 * 1 Aug 95     Mark Olesen <olesen@me.queensu.ca>
 *      - define FFT_RADIX4 to only have radix 2, radix 4 transforms
 *      - made fftradix /fftradixf () static scope, just use fftn()
 *        instead.  If you have good ideas about fixing the factors
 *        in fftn() please do so.
 *
 * ==========================================================================*
 * NIST Guide to Available Math Software.
 * Source for module FFT from package GO.
```

```
* Retrieved from NETLIB on Wed Jul  5 11:50:07 1995.
* ===========================================================================*
*
*---------------------------------------------------------------------------*
*
* int fftn (int ndim, const int dims[], REAL Re[], REAL Im[],
*           int iSign, double scaling);
*
* NDIM = the total number dimensions
* DIMS = a vector of array sizes
*      if NDIM is zero then DIMS must be zero-terminated
*
* RE and IM hold the real and imaginary components of the data, and return
* the resulting real and imaginary Fourier coefficients.  Multidimensional
* data *must* be allocated contiguously.  There is no limit on the number
* of dimensions.
*
* ISIGN = the sign of the complex exponential (ie, forward or inverse FFT)
*      the magnitude of ISIGN (normally 1) is used to determine the
*      correct indexing increment (see below).
*
* SCALING = normalizing constant by which the final result is *divided*
*      if SCALING == -1, normalize by total dimension of the transform
*      if SCALING <  -1, normalize by the square-root of the total dimension
*
* example:
* tri-variate transform with Re[n1][n2][n3], Im[n1][n2][n3]
*
*      int dims[3] = {n1,n2,n3}
*      fftn (3, dims, Re, Im, 1, scaling);
*
*---------------------------------------------------------------------------*
* int fftradix (REAL Re[], REAL Im[], size_t nTotal, size_t nPass,
*               size_t nSpan, int iSign, size_t max_factors,
*               size_t max_perm);
*
* RE, IM - see above documentation
*
* Although there is no limit on the number of dimensions, fftradix() must
* be called once for each dimension, but the calls may be in any order.
*
* NTOTAL = the total number of complex data values
* NPASS  = the dimension of the current variable
* NSPAN/NPASS = the spacing of consecutive data values while indexing the
*      current variable
* ISIGN - see above documentation
*
* example:
* tri-variate transform with Re[n1][n2][n3], Im[n1][n2][n3]
*
*      fftradix (Re, Im, n1*n2*n3, n1,       n1, 1, maxf, maxp);
*      fftradix (Re, Im, n1*n2*n3, n2,    n1*n2, 1, maxf, maxp);
*      fftradix (Re, Im, n1*n2*n3, n3, n1*n2*n3, 1, maxf, maxp);
*
* single-variate transform,
*    NTOTAL = N = NSPAN = (number of complex data values),
```

```
 *
 *      fftradix (Re, Im, n, n, n, 1, maxf, maxp);
 *
 * The data can also be stored in a single array with alternating real and
 * imaginary parts, the magnitude of ISIGN is changed to 2 to give correct
 * indexing increment, and data [0] and data [1] used to pass the initial
 * addresses for the sequences of real and imaginary values,
 *
 * example:
 *      REAL data [2*NTOTAL];
 *      fftradix ( &data[0], &data[1], NTOTAL, nPass, nSpan, 2, maxf, maxp);
 *
 * for temporary allocation:
 *
 * MAX_FACTORS  >= the maximum prime factor of NPASS
 * MAX_PERM     >= the number of prime factors of NPASS.  In addition,
 *      if the square-free portion K of NPASS has two or more prime
 *      factors, then MAX_PERM >= (K-1)
 *
 * storage in FACTOR for a maximum of 15 prime factors of NPASS. if NPASS
 * has more than one square-free factor, the product of the square-free
 * factors must be <= 210 array storage for maximum prime factor of 23 the
 * following two constants should agree with the array dimensions.
 *
 *----------------------------------------------------------------------*
 *
 * void fft_free (void);
 *
 * free-up allocated temporary storage after finished all the Fourier
 * transforms.
 *
 *----------------------------------------------------------------------*/
#ifndef _FFTN_C
#define _FFTN_C
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "fftn.h"

/* double precision routine */
static int
fftradix (double Re[], double Im[],
          size_t nTotal, size_t nPass, size_t nSpan, int isign,
          int max_factors, int max_perm);

/* float precision routine */
static int
fftradixf (float Re[], float Im[],
           size_t nTotal, size_t nPass, size_t nSpan, int isign,
           int max_factors, int max_perm);

/* parameters for memory management */

static size_t SpaceAlloced = 0;
static size_t MaxPermAlloced = 0;
```

```
/* temp space, (void *) since both float and double routines use it */
static void *Tmp0 = NULL;       /* temp space for real part */
static void *Tmp1 = NULL;       /* temp space for imaginary part */
static void *Tmp2 = NULL;       /* temp space for Cosine values */
static void *Tmp3 = NULL;       /* temp space for Sine values */
static int  *Perm = NULL;       /* Permutation vector */

#define NFACTOR 11
static int factor [NFACTOR];

void
fft_free (void)
{
   SpaceAlloced = MaxPermAlloced = 0;
   if (Tmp0 != NULL)     { free (Tmp0);  Tmp0 = NULL; }
   if (Tmp1 != NULL)     { free (Tmp1);  Tmp1 = NULL; }
   if (Tmp2 != NULL)     { free (Tmp2);  Tmp2 = NULL; }
   if (Tmp3 != NULL)     { free (Tmp3);  Tmp3 = NULL; }
   if (Perm != NULL)     { free (Perm);  Perm = NULL; }
}

#if !defined (__FILE__) && !defined (lint)
# define __FILE__="fftn.c"
#endif

#ifndef M_PI
# define M_PI   3.14159265358979323846264338327950288
#endif

#ifndef SIN60
# define SIN60  0.86602540378443865     /* sin(60 deg) */
# define COS72  0.30901699437494742     /* cos(72 deg) */
# define SIN72  0.95105651629515357     /* sin(72 deg) */
#endif

/* re-include this source file on the second pass through */
#undef REAL
#undef FFTN
#undef FFTNS
#undef FFTRADIX
#undef FFTRADIXS

#ifndef FFT_NOFLOAT
# define REAL           float
# define FFTN           fftnf           /* trailing 'f' for float */
# define FFTNS          "fftnf"         /* name for error message */
# define FFTRADIX       fftradixf       /* trailing 'f' for float */
# define FFTRADIXS      "fftradixf"     /* name for error message */
# include __FILE__                      /* include this file again */
#endif

#undef REAL
#undef FFTN
#undef FFTNS
```

```
#undef FFTRADIX
#undef FFTRADIXS

#ifndef FFT_NODOUBLE
# define REAL           double
# define FFTN           fftn
# define FFTNS          "fftn"
# define FFTRADIX       fftradix
# define FFTRADIXS      "fftradix"
# include __FILE__                      /* include this file again */
#endif

#if defined (FFT_NOFLOAT) && defined (FFT_NODOUBLE) && !defined (lint)
Error: cannot have both -DFFT_NOFLOAT and -DFFT_NODOUBLE
#endif
#else   /* _FFTN_C */

/*
 *
 */
int
FFTN (int ndim,
         const int dims[],
     REAL Re [],
     REAL Im [],
     int iSign,
     double scaling)
{
   size_t nSpan, nPass, nTotal;
   int ret, i, max_factors, max_perm;

   /*
    * tally the number of elements in the data array
    * and determine the number of dimensions
    */
   nTotal = 1;
   if (ndim && dims [0])
     {
        for (i = 0; i < ndim; i++)
          {
             if (dims [i] <= 0)
               {
                  fputs ("Error: " FFTNS "() - dimension error\n", stderr);
                  fft_free ();  /* free-up memory */
                  return -1;
               }
             nTotal *= dims [i];
          }
     }
   else
     {
        ndim = 0;
        for (i = 0; dims [i]; i++)
          {
             if (dims [i] <= 0)
```

```
                {
                   fputs ("Error: " FFTNS "() - dimension error\n", stderr);
                   fft_free ();  /* free-up memory */
                   return -1;
                }
             nTotal *= dims [i];
             ndim++;
          }
      }

   /* determine maximum number of factors and permuations */
#if 1
   /*
    * follow John Beale's example, just use the largest dimension and don't
    * worry about excess allocation.  May be someone else will do it?
    */
   max_factors = max_perm = 1;
   for (i = 0; i < ndim; i++)
     {
        nSpan = dims [i];
        if (nSpan > max_factors) max_factors = nSpan;
        if (nSpan > max_perm) max_perm = nSpan;
     }
#else
   /* use the constants used in the original Fortran code */
   max_factors = 23;
   max_perm = 209;
#endif
   /* loop over the dimensions: */
   nPass = 1;
   for (i = 0; i < ndim; i++)
     {
        nSpan = dims [i];
        nPass *= nSpan;
        ret = FFTRADIX (Re, Im, nTotal, nSpan, nPass, iSign,
                        max_factors, max_perm);
        /* exit, clean-up already done */
        if (ret)
          return ret;
     }

   /* Divide through by the normalizing constant: */
   if (scaling && scaling != 1.0)
     {
        if (iSign < 0) iSign = -iSign;
        if (scaling < 0.0)
          {
             scaling = nTotal;
             if (scaling < -1.0)
               scaling = sqrt (scaling);
          }
        scaling = 1.0 / scaling;        /* multiply is often faster */
        for (i = 0; i < nTotal; i += iSign)
          {
```

```
                Re [i] *= scaling;
                Im [i] *= scaling;
            }
        }
    return 0;
}

/*
 * singleton's mixed radix routine
 *
 * could move allocation out to fftn(), but leave it here so that it's
 * possible to make this a standalone function
 */
static int
FFTRADIX (REAL Re[],
          REAL Im[],
          size_t nTotal,
          size_t nPass,
          size_t nSpan,
          int iSign,
          int max_factors,
          int max_perm)
{
    int ii, mfactor, kspan, ispan, inc;
    int j, jc, jf, jj, k, kl, k2, k3, k4, kk, kt, nn, ns, nt;

    REAL radf;
    REAL c1, c2, c3, cd, aa, aj, ak, ajm, ajp, akm, akp;
    REAL s1, s2, s3, sd, bb, bj, bk, bjm, bjp, bkm, bkp;

    REAL *Rtmp = NULL;    /* temp space for real part*/
    REAL *Itmp = NULL;    /* temp space for imaginary part */
    REAL *Cos = NULL;     /* Cosine values */
    REAL *Sin = NULL;     /* Sine values */

    REAL s60 = SIN60;             /* sin(60 deg) */
    REAL c72 = COS72;             /* cos(72 deg) */
    REAL s72 = SIN72;             /* sin(72 deg) */
    REAL pi2 = M_PI;              /* use PI first, 2 PI later */

    /* gcc complains about k3 being uninitialized, but I can't find out where
     * or why ... it looks okay to me.
     *
     * initialize to make gcc happy
     */
    k3 = 0;

    /* gcc complains about c2, c3, s2,s3 being uninitialized, but they're
     * only used for the radix 4 case and only AFTER the (s1 == 0.0) pass
     * through the loop at which point they will have been calculated.
     *
     * initialize to make gcc happy
     */
    c2 = c3 = s2 = s3 = 0.0;
```

```
    /* Parameter adjustments, was fortran so fix zero-offset */
    Re--;
    Im--;

    if (nPass < 2)
      return 0;

    /*  allocate storage */
    if (SpaceAlloced < max_factors * sizeof (REAL))
      {
#ifdef SUN_BROKEN_REALLOC
        if (!SpaceAlloced)      /* first time */
          {
            SpaceAlloced = max_factors * sizeof (REAL);
            Tmp0 = (void *) malloc (SpaceAlloced);
            Tmp1 = (void *) malloc (SpaceAlloced);
            Tmp2 = (void *) malloc (SpaceAlloced);
            Tmp3 = (void *) malloc (SpaceAlloced);
          }
        else
          {
#endif
            SpaceAlloced = max_factors * sizeof (REAL);
            Tmp0 = (void *) realloc (Tmp0, SpaceAlloced);
            Tmp1 = (void *) realloc (Tmp1, SpaceAlloced);
            Tmp2 = (void *) realloc (Tmp2, SpaceAlloced);
            Tmp3 = (void *) realloc (Tmp3, SpaceAlloced);
#ifdef SUN_BROKEN_REALLOC
          }
#endif
      }
    else
      {
        /* allow full use of alloc'd space */
        max_factors = SpaceAlloced / sizeof (REAL);
      }
    if (MaxPermAlloced < max_perm)
      {
#ifdef SUN_BROKEN_REALLOC
        if (!MaxPermAlloced)    /* first time */
          Perm = (int *) malloc (max_perm * sizeof(int));
        else
#endif
          Perm = (int *) realloc (Perm, max_perm * sizeof(int));
        MaxPermAlloced = max_perm;
      }
    else
      {
        /* allow full use of alloc'd space */
        max_perm = MaxPermAlloced;
      }
    if (Tmp0 == NULL || Tmp1 == NULL || Tmp2 == NULL || Tmp3 == NULL
        || Perm == NULL)
```

```
    goto Memory_Error_Label;

  /* assign pointers */
  Rtmp = (REAL *) Tmp0;
  Itmp = (REAL *) Tmp1;
  Cos  = (REAL *) Tmp2;
  Sin  = (REAL *) Tmp3;

  /*
   * Function Body
   */
  inc = iSign;
  if (iSign < 0) {
     s72 = -s72;
     s60 = -s60;
     pi2 = -pi2;
     inc = -inc;                /* absolute value */
  }

  /* adjust for strange increments */
  nt = inc * nTotal;
  ns = inc * nSpan;
  kspan = ns;

  nn = nt - inc;
  jc = ns / nPass;
  radf = pi2 * (double) jc;
  pi2 *= 2.0;                   /* use 2 PI from here on */

  ii = 0;
  jf = 0;
  /*  determine the factors of n */
  mfactor = 0;
  k = nPass;
  while (k % 16 == 0) {
     mfactor++;
     factor [mfactor - 1] = 4;
     k /= 16;
  }
  j = 3;
  jj = 9;
  do {
     while (k % jj == 0) {
        mfactor++;
        factor [mfactor - 1] = j;
        k /= jj;
     }
     j += 2;
     jj = j * j;
  } while (jj <= k);
  if (k <= 4) {
     kt = mfactor;
     factor [mfactor] = k;
     if (k != 1)
       mfactor++;
```

```
} else {
   if (k - (k / 4 << 2) == 0) {
      mfactor++;
      factor [mfactor - 1] = 2;
      k /= 4;
   }
   kt = mfactor;
   j = 2;
   do {
      if (k % j == 0) {
         mfactor++;
         factor [mfactor - 1] = j;
         k /= j;
      }
      j = ((j + 1) / 2 << 1) + 1;
   } while (j <= k);
}
if (kt) {
   j = kt;
   do {
      mfactor++;
      factor [mfactor - 1] = factor [j - 1];
      j--;
   } while (j);
}

/* test that mfactors is in range */
if (mfactor > NFACTOR)
  {
     fputs ("Error: " FFTRADIXS "() - exceeded number of factors\n", stderr);
     goto Memory_Error_Label;
   }

/* compute fourier transform */
for (;;) {
   sd = radf / (double) kspan;
   cd = sin(sd);
   cd = 2.0 * cd * cd;
   sd = sin(sd + sd);
   kk = 1;
   ii++;

   switch (factor [ii - 1]) {
    case 2:
      /* transform for factor of 2 (including rotation factor) */
      kspan /= 2;
      k1 = kspan + 2;
      do {
         do {
            k2 = kk + kspan;
            ak = Re [k2];
            bk = Im [k2];
            Re [k2] = Re [kk] - ak;
            Im [k2] = Im [kk] - bk;
            Re [kk] += ak;
```

```
            Im [kk] += bk;
            kk = k2 + kspan;
         } while (kk <= nn);
         kk -= nn;
      } while (kk <= jc);
      if (kk > kspan)
        goto Permute_Results_Label;          /* exit infinite loop */
      do {
         c1 = 1.0 - cd;
         s1 = sd;
         do {
            do {
               do {
                  k2 = kk + kspan;
                  ak = Re [kk] - Re [k2];
                  bk = Im [kk] - Im [k2];
                  Re [kk] += Re [k2];
                  Im [kk] += Im [k2];
                  Re [k2] = c1 * ak - s1 * bk;
                  Im [k2] = s1 * ak + c1 * bk;
                  kk = k2 + kspan;
               } while (kk < nt);
               k2 = kk - nt;
               c1 = -c1;
               kk = k1 - k2;
            } while (kk > k2);
            ak = c1 - (cd * c1 + sd * s1);
            s1 = sd * c1 - cd * s1 + s1;
            c1 = 2.0 - (ak * ak + s1 * s1);
            s1 *= c1;
            c1 *= ak;
            kk += jc;
         } while (kk < k2);
         k1 += inc + inc;
         kk = (k1 - kspan) / 2 + jc;
      } while (kk <= jc + jc);
      break;

   case 4:                    /* transform for factor of 4 */
      ispan = kspan;
      kspan /= 4;

      do {
         c1 = 1.0;
         s1 = 0.0;
         do {
            do {
               k1 = kk + kspan;
               k2 = k1 + kspan;
               k3 = k2 + kspan;
               akp = Re [kk] + Re [k2];
               akm = Re [kk] - Re [k2];
               ajp = Re [k1] + Re [k3];
               ajm = Re [k1] - Re [k3];
               bkp = Im [kk] + Im [k2];
```

```
            bkm = Im [kk] - Im [k2];
            bjp = Im [k1] + Im [k3];
            bjm = Im [k1] - Im [k3];
            Re [kk] = akp + ajp;
            Im [kk] = bkp + bjp;
            ajp = akp - ajp;
            bjp = bkp - bjp;
            if (iSign < 0) {
               akp = akm + bjm;
               bkp = bkm - ajm;
               akm -= bjm;
               bkm += ajm;
            } else {
               akp = akm - bjm;
               bkp = bkm + ajm;
               akm += bjm;
               bkm -= ajm;
            }
            /* avoid useless multiplies */
            if (s1 == 0.0) {
               Re [k1] = akp;
               Re [k2] = ajp;
               Re [k3] = akm;
               Im [k1] = bkp;
               Im [k2] = bjp;
               Im [k3] = bkm;
            } else {
               Re [k1] = akp * c1 - bkp * s1;
               Re [k2] = ajp * c2 - bjp * s2;
               Re [k3] = akm * c3 - bkm * s3;
               Im [k1] = akp * s1 + bkp * c1;
               Im [k2] = ajp * s2 + bjp * c2;
               Im [k3] = akm * s3 + bkm * c3;
            }
            kk = k3 + kspan;
         } while (kk <= nt);

         c2 = c1 - (cd * c1 + sd * s1);
         s1 = sd * c1 - cd * s1 + s1;
         c1 = 2.0 - (c2 * c2 + s1 * s1);
         s1 *= c1;
         c1 *= c2;
         /* values of c2, c3, s2, s3 that will get used next time */
         c2 = c1 * c1 - s1 * s1;
         s2 = 2.0 * c1 * s1;
         c3 = c2 * c1 - s2 * s1;
         s3 = c2 * s1 + s2 * c1;
         kk = kk - nt + jc;
      } while (kk <= kspan);
      kk = kk - kspan + inc;
   } while (kk <= jc);
   if (kspan == jc)
     goto Permute_Results_Label;          /* exit infinite loop */
   break;
```

```
        default:
          /*  transform for odd factors */
#ifdef FFT_RADIX4
          fputs ("Error: " FFTRADIXS "(): compiled for radix 2/4 only\n", stderr);
          fft_free ();           /* free-up memory */
          return -1;
          break;
#else    /* FFT_RADIX4 */
          k = factor [ii - 1];
          ispan = kspan;
          kspan /= k;

          switch (k) {
           case 3:       /* transform for factor of 3 (optional code) */
             do {
                do {
                   k1 = kk + kspan;
                   k2 = k1 + kspan;
                   ak = Re [kk];
                   bk = Im [kk];
                   aj = Re [k1] + Re [k2];
                   bj = Im [k1] + Im [k2];
                   Re [kk] = ak + aj;
                   Im [kk] = bk + bj;
                   ak -= 0.5 * aj;
                   bk -= 0.5 * bj;
                   aj = (Re [k1] - Re [k2]) * s60;
                   bj = (Im [k1] - Im [k2]) * s60;
                   Re [k1] = ak - bj;
                   Re [k2] = ak + bj;
                   Im [k1] = bk + aj;
                   Im [k2] = bk - aj;
                   kk = k2 + kspan;
                } while (kk < nn);
                kk -= nn;
             } while (kk <= kspan);
             break;

           case 5:       /*  transform for factor of 5 (optional code) */
             c2 = c72 * c72 - s72 * s72;
             s2 = 2.0 * c72 * s72;
             do {
                do {
                   k1 = kk + kspan;
                   k2 = k1 + kspan;
                   k3 = k2 + kspan;
                   k4 = k3 + kspan;
                   akp = Re [k1] + Re [k4];
                   akm = Re [k1] - Re [k4];
                   bkp = Im [k1] + Im [k4];
                   bkm = Im [k1] - Im [k4];
                   ajp = Re [k2] + Re [k3];
                   ajm = Re [k2] - Re [k3];
                   bjp = Im [k2] + Im [k3];
                   bjm = Im [k2] - Im [k3];
```

```
            aa = Re [kk];
            bb = Im [kk];
            Re [kk] = aa + akp + ajp;
            Im [kk] = bb + bkp + bjp;
            ak = akp * c72 + ajp * c2 + aa;
            bk = bkp * c72 + bjp * c2 + bb;
            aj = akm * s72 + ajm * s2;
            bj = bkm * s72 + bjm * s2;
            Re [k1] = ak - bj;
            Re [k4] = ak + bj;
            Im [k1] = bk + aj;
            Im [k4] = bk - aj;
            ak = akp * c2 + ajp * c72 + aa;
            bk = bkp * c2 + bjp * c72 + bb;
            aj = akm * s2 - ajm * s72;
            bj = bkm * s2 - bjm * s72;
            Re [k2] = ak - bj;
            Re [k3] = ak + bj;
            Im [k2] = bk + aj;
            Im [k3] = bk - aj;
            kk = k4 + kspan;
         } while (kk < nn);
         kk -= nn;
      } while (kk <= kspan);
      break;

   default:
      if (k != jf) {
         jf = k;
         s1 = pi2 / (double) k;
         c1 = cos(s1);
         s1 = sin(s1);
         if (jf > max_factors)
           goto Memory_Error_Label;
         Cos [jf - 1] = 1.0;
         Sin [jf - 1] = 0.0;
         j = 1;
         do {
            Cos [j - 1] = Cos [k - 1] * c1 + Sin [k - 1] * s1;
            Sin [j - 1] = Cos [k - 1] * s1 - Sin [k - 1] * c1;
            k--;
            Cos [k - 1] = Cos [j - 1];
            Sin [k - 1] = -Sin [j - 1];
            j++;
         } while (j < k);
      }
      do {
         do {
            k1 = kk;
            k2 = kk + ispan;
            ak = aa = Re [kk];
            bk = bb = Im [kk];
            j = 1;
            k1 += kspan;
            do {
```

```
            k2 -= kspan;
            j++;
            Rtmp [j - 1] = Re [k1] + Re [k2];
            ak += Rtmp [j - 1];
            Itmp [j - 1] = Im [k1] + Im [k2];
            bk += Itmp [j - 1];
            j++;
            Rtmp [j - 1] = Re [k1] - Re [k2];
            Itmp [j - 1] = Im [k1] - Im [k2];
            k1 += kspan;
         } while (k1 < k2);
         Re [kk] = ak;
         Im [kk] = bk;
         k1 = kk;
         k2 = kk + ispan;
         j = 1;
         do {
            k1 += kspan;
            k2 -= kspan;
            jj = j;
            ak = aa;
            bk = bb;
            aj = 0.0;
            bj = 0.0;
            k = 1;
            do {
               k++;
               ak += Rtmp [k - 1] * Cos [jj - 1];
               bk += Itmp [k - 1] * Cos [jj - 1];
               k++;
               aj += Rtmp [k - 1] * Sin [jj - 1];
               bj += Itmp [k - 1] * Sin [jj - 1];
               jj += j;
               if (jj > jf) {
                  jj -= jf;
               }
            } while (k < jf);
            k = jf - j;
            Re [k1] = ak - bj;
            Im [k1] = bk + aj;
            Re [k2] = ak + bj;
            Im [k2] = bk - aj;
            j++;
         } while (j < k);
         kk += ispan;
      } while (kk <= nn);
      kk -= nn;
   } while (kk <= kspan);
   break;
}
/*  multiply by rotation factor (except for factors of 2 and 4) */
if (ii == mfactor)
  goto Permute_Results_Label;          /* exit infinite loop */
kk = jc + 1;
do {
```

```
            c2 = 1.0 - cd;
            s1 = sd;
            do {
               c1 = c2;
               s2 = s1;
               kk += kspan;
               do {
                  do {
                     ak = Re [kk];
                     Re [kk] = c2 * ak - s2 * Im [kk];
                     Im [kk] = s2 * ak + c2 * Im [kk];
                     kk += ispan;
                  } while (kk <= nt);
                  ak = s1 * s2;
                  s2 = s1 * c2 + c1 * s2;
                  c2 = c1 * c2 - ak;
                  kk = kk - nt + kspan;
               } while (kk <= ispan);
               c2 = c1 - (cd * c1 + sd * s1);
               s1 += sd * c1 - cd * s1;
               c1 = 2.0 - (c2 * c2 + s1 * s1);
               s1 *= c1;
               c2 *= c1;
               kk = kk - ispan + jc;
            } while (kk <= kspan);
            kk = kk - kspan + jc + inc;
         } while (kk <= jc + jc);
         break;
#endif   /* FFT_RADIX4 */
      }
   }

/*  permute the results to normal order---done in two stages */
/*  permutation for square factors of n */
Permute_Results_Label:
   Perm [0] = ns;
   if (kt) {
      k = kt + kt + 1;
      if (mfactor < k)
        k--;
      j = 1;
      Perm [k] = jc;
      do {
         Perm [j] = Perm [j - 1] / factor [j - 1];
         Perm [k - 1] = Perm [k] * factor [j - 1];
         j++;
         k--;
      } while (j < k);
      k3 = Perm [k];
      kspan = Perm [1];
      kk = jc + 1;
      k2 = kspan + 1;
      j = 1;
      if (nPass != nTotal) {
/*  permutation for multivariate transform */
```

```
Permute_Multi_Label:
         do {
            do {
               k = kk + jc;
               do {
                  /* swap Re [kk] <> Re [k2], Im [kk] <> Im [k2] */
                  ak = Re [kk]; Re [kk] = Re [k2]; Re [k2] = ak;
                  bk = Im [kk]; Im [kk] = Im [k2]; Im [k2] = bk;
                  kk += inc;
                  k2 += inc;
               } while (kk < k);
               kk += ns - jc;
               k2 += ns - jc;
            } while (kk < nt);
            k2 = k2 - nt + kspan;
            kk = kk - nt + jc;
         } while (k2 < ns);
         do {
            do {
               k2 -= Perm [j - 1];
               j++;
               k2 = Perm [j] + k2;
            } while (k2 > Perm [j - 1]);
            j = 1;
            do {
               if (kk < k2)
                 goto Permute_Multi_Label;
               kk += jc;
               k2 += kspan;
            } while (k2 < ns);
         } while (kk < ns);
      } else {
/*  permutation for single-variate transform (optional code) */
Permute_Single_Label:
         do {
            /* swap Re [kk] <> Re [k2], Im [kk] <> Im [k2] */
            ak = Re [kk]; Re [kk] = Re [k2]; Re [k2] = ak;
            bk = Im [kk]; Im [kk] = Im [k2]; Im [k2] = bk;
            kk += inc;
            k2 += kspan;
         } while (k2 < ns);
         do {
            do {
               k2 -= Perm [j - 1];
               j++;
               k2 = Perm [j] + k2;
            } while (k2 > Perm [j - 1]);
            j = 1;
            do {
               if (kk < k2)
                 goto Permute_Single_Label;
               kk += inc;
               k2 += kspan;
            } while (k2 < ns);
         } while (kk < ns);
```

```
    }
    jc = k3;
}

if ((kt << 1) + 1 >= mfactor)
  return 0;
ispan = Perm [kt];
/* permutation for square-free factors of n */
j = mfactor - kt;
factor [j] = 1;
do {
    factor [j - 1] *= factor [j];
    j--;
} while (j != kt);
kt++;
nn = factor [kt - 1] - 1;
if (nn > max_perm)
  goto Memory_Error_Label;
j = jj = 0;
for (;;) {
    k = kt + 1;
    k2 = factor [kt - 1];
    kk = factor [k - 1];
    j++;
    if (j > nn)
      break;                          /* exit infinite loop */
    jj += kk;
    while (jj >= k2) {
        jj -= k2;
        k2 = kk;
        k++;
        kk = factor [k - 1];
        jj += kk;
    }
    Perm [j - 1] = jj;
}
/*  determine the permutation cycles of length greater than 1 */
j = 0;
for (;;) {
    do {
        j++;
        kk = Perm [j - 1];
    } while (kk < 0);
    if (kk != j) {
        do {
            k = kk;
            kk = Perm [k - 1];
            Perm [k - 1] = -kk;
        } while (kk != j);
        k3 = kk;
    } else {
        Perm [j - 1] = -j;
        if (j == nn)
          break;                /* exit infinite loop */
    }
```

```
}
max_factors *= inc;
/*  reorder a and b, following the permutation cycles */
for (;;) {
   j = k3 + 1;
   nt -= ispan;
   ii = nt - inc + 1;
   if (nt < 0)
     break;                    /* exit infinite loop */
   do {
      do {
         j--;
      } while (Perm [j - 1] < 0);
      jj = jc;
      do {
         kspan = jj;
         if (jj > max_factors) {
            kspan = max_factors;
         }
         jj -= kspan;
         k = Perm [j - 1];
         kk = jc * k + ii + jj;
         k1 = kk + kspan;
         k2 = 0;
         do {
            k2++;
            Rtmp [k2 - 1] = Re [k1];
            Itmp [k2 - 1] = Im [k1];
            k1 -= inc;
         } while (k1 != kk);
         do {
            k1 = kk + kspan;
            k2 = k1 - jc * (k + Perm [k - 1]);
            k = -Perm [k - 1];
            do {
               Re [k1] = Re [k2];
               Im [k1] = Im [k2];
               k1 -= inc;
               k2 -= inc;
            } while (k1 != kk);
            kk = k2;
         } while (k != j);
         k1 = kk + kspan;
         k2 = 0;
         do {
            k2++;
            Re [k1] = Rtmp [k2 - 1];
            Im [k1] = Itmp [k2 - 1];
            k1 -= inc;
         } while (k1 != kk);
      } while (jj);
   } while (j != 1);
}
return 0;                      /* exit point here */
```

```
/*---------------------------------*-C-*---------------------------------*
 * File:
 *      fftn.h
 * ------------------------------------------------------------------------*
 * Re[]:        real value array
 * Im[]:        imaginary value array
 * nTotal:      total number of complex values
 * nPass:       number of elements involved in this pass of transform
 * nSpan:       nspan/nPass = number of bytes to increment pointer
 *              in Re[] and Im[]
 * isign:       exponent: +1 = forward  -1 = reverse
 * scaling:     normalizing constant by which the final result is *divided*
 *      scaling == -1, normalize by total dimension of the transform
 *      scaling <  -1, normalize by the square-root of the total dimension
 *
 * ------------------------------------------------------------------------
 * See the comments in the code for correct usage!
 */

#ifndef _FFTN_H
#define _FFTN_H

#ifdef __cplusplus
extern "C" {
#endif

void fft_free (void);

/* double precision routine */
int fftn (int ndim, const int dims[], double Re[], double Im[],
                int isign, double scaling);

/* float precision routine */
int fftnf (int ndim, const int dims[], float Re[], float Im[],
                 int isign, double scaling);

#ifdef __cplusplus
}
#endif



#endif  /* _FFTN_H */
```

```
//////////////////////////////////////////////////////////
// File: Hbdemo.cpp
//
// Purpose: Demonstrates how to use harmonic balance alghoritm
//

#include "matrix.h"
#include <time.h>
#ifdef WIN32
        #include <windows.h>
#endif
#include "fftn.h"

#ifdef _DEBUG
        #include <conio.h>
#endif

//////////////////////////////////
// Note: The following conditional compilation statements are included
//       so that you can (likely to) compile this sample, without any
//       modification, using a compiler which does not support any or
//       all of the ANSI C++ features like NAMEPACE, TEMPLATE, and
//       EXCEPTION, used in this class.
//
//       If you have a compiler, such as C++ Builder, Borland C++ 5.0,
//       MS Visual C++ 5.0 or higher, etc., which supports most of the ANSI
//       C++ features used in this class, you do not need to include these
//       statements in your program.
//
#ifndef _NO_NAMESPACE
using namespace std;
using namespace math;
#define STD std
#else
#define STD
#endif

#if !defined(_NO_TEMPLATE)
#  if defined(_MSC_VER)
#     if _MSC_VER > 1000
#        include <complex>
         typedef complex<double> type;
#     else
         typedef double type;
#     endif
#  elif defined(__BORLANDC__)
#     if defined(__WIN32__)
#        include <complex>
         typedef complex<double> type;
#     else
#        include <complex.h>
         typedef complex type;
#     endif
#  elif defined( __GNUG__ )
#       include <complex>
        typedef complex<double> type;
# elif defined( _SGI_BROKEN_STL )
# include <complex>
typedef std::complex<double> type;
#  endif
  typedef matrix<type> Matrix;
#else
   typedef matrix Matrix;
#endif

#ifndef _NO_EXCEPTION
#  define TRYBEGIN()    try {
#  define CATCHERROR()  } catch (const STD::exception& e) { \
                      cerr << "Error: " << e.what() << endl; }
#else
#  define TRYBEGIN()
#  define CATCHERROR()
#endif
```



```
#define  SWAP(type, a, b)   {type swapx; swapx = a; a = b; b = swapx;}
```

```
int portnmb[64];
int no_of_freqs;   //This must include DC and fund..so no_of_freqs >= 2
double pi = 3.1415926535;
double freq;

 // initialize some arrays and variables.

int no_of_srces;
int no_of_nodes;
int no_of_samples;
int no_of_elements;
int no_of_srcelmts;

Matrix Vs_array;
Matrix V_array;
Matrix Vs_array_old;
Matrix V_array_time;
Matrix Vs_array_time;
Matrix Vfinal_array;
Matrix Y_array;
Matrix Z_array;
Matrix Ys_array;
Matrix Ig_array;
Matrix Ig_array_old;
Matrix Ig_array_nonlineal;
Matrix Ig_array_time;
Matrix Err;
Matrix Err_old;
Matrix J;
Matrix tiempo;

int counter = 0;
int converged;
double err_sum, err_sum_old, error_margin;
double eps_iter, eps_sol;

double sum(Matrix x)
{

double s;
int i;
int no = x.RowNo();

    s = 0;
        for (i=0; i < no; i++)
         s += abs(x(i,0));

return s;
}

Matrix diode(Matrix input_signal, int voltage_in)
{

//function [result] = get_diode_current(input_signal,voltage_in)
//This function returns the diode current or voltage that's associated with the diode voltage or current input.

int converged;
double L;
double d,alpha;
double k,q,T,VT,Rr,Rb,Is,n;
double F,F_old_old;
double x, x_old, x_old_old;
double vIN;
double der;
double dither;
double f, f2, x2;
double F_old, f_old;
double delta_x;
double F2;
double iIN;
double itt_param_finish;

k = 1.38E-23;
q = 1.602E-19;
```

```
T = 300;
VT = k*T/q;
n = 1.5;
Is = exp(1-0.7/(n*VT));
Rb = 5;
Rr = 10E6;

L = input_signal.RowNo();

Matrix MResult(L,1);
MResult.Null();

int counter = 0;
F = 1E100;
x = 0.1;
x_old = 0;
alpha = 0.4;
if(voltage_in == 1)
{
        x = 1;
        converged = 1;
        for (d = 0;d<L;d++)
        {
                vIN = input_signal(d,0).real();
                F_old_old = 1E200;
                x_old_old = 0;
//              printf("x=%0.16e, F=%0.16e, F_old_old=%0.16e",x,F,F_old_old);
//              printf("\n");
                while(1)
                {
                        if((x*n*VT) > 0.1)
                        {
                                x = 0.1/(n*VT);
                        }
                        F = (vIN-x)/Rb-Is*(exp(x/(n*VT))-1);
                        der = -1/Rb-Is/(n*VT)*exp(x/(n*VT));
                        x = x-F/(der);
//                      printf("x=%0.16e, F=%0.16e, F_old_old=%0.16e,  alpha=%0.5e",x,F,F_old_old, alpha);
//                      printf("\n");
                        if((abs((x-x_old)/x)<1E-10)||(abs(F)<1E-13))
                        {
                                break;
                        }
                        counter = counter + 1;
                        if(counter > 300)
                        {
                                fprintf(stderr, "diode function is not converging..\n");
                        }
/*
                        if(abs(F) > abs(F_old_old))
                        {
                                alpha = alpha*0.5;
                                F = F_old_old;
                                x = x_old_old;
                        }
                        else
                        {
                                if(counter > 5)
                                {
                                        F_old_old = F;
                                        x_old_old = x;
                                        alpha = alpha*1.1;
                                        if(alpha > 1)
                                        {
                                                alpha = 1;
                                        }
                                }
                        }
*/
                        x_old = x;
                }
                MResult(d,0) = (vIN-x)/Rb + vIN/Rr;
        }
}
else
{
```

```
        dither = 0.0001;
        converged = 1;
        for (d = 0; d<L-1; d++);
        {

                iIN = input_signal(d,0).real();
                x = 1.0;                        //If you have convergence problems, change this to larger value.
                f_old = 0;
                F_old = 0;
                F_old_old = 1E200;
                x_old_old = 0;
                x_old = x;
                while(1)
                {
                        /*
                        if((n*n*VT) > 0.1)
                        {
                                x = 0.1/(n*VT);
                        }
                        */
                        f = Is*(exp(x/(n*VT))-1);
                        F = f - iIN*Rr/(Rr+Rb+x/f);
                        f_old = f;
                        F_old = F;
                        delta_x = x*dither;
                        x2 = x+delta_x;
                        f2 = Is*(exp(x2/(n*VT))-1);
                        F2 = f2 - iIN*Rr/(Rr+Rb+x2/f2);
                        der = (F2-F)/delta_x;
                        x = x-F/(der)*alpha;
//                      printf('x=//0.16e, F=//0.16e  f=//0.16e',x,F,f);
//                      printf("\n");
                        if(( abs((x-x_old)/x)<1E-10)|(abs(F)<1E-13))
                        {
                                break;
                        }
                        counter= counter + 1;

                        if(counter > 300)
                        {
                                cout << "initial_parameters" << endl;
                                cout << "x" << x << endl;
                                cout << "iIN" << iIN << endl;
                            fprintf(stderr, "diode_current function is not converging..\n");

                        }

                        if(abs(F) > abs(F_old_old))
                        {
                                alpha = alpha*0.5;
                                F = F_old_old;
                                x = x_old_old;
                        }
                        else
                        {
                                if(counter > 5)
                                {
                                        F_old_old = F;
                                        x_old_old = x;
                                        alpha = alpha*1.1;

                                        if(alpha > 1)
                                        {
                                                alpha = 1;
                                        }
                                }
                        }

                        x_old = x;
                }
                MResult(d,0) = (iIN-f)*Rr;
        }
}

itt_param_finish = x;
```

```
return MResult;

}

Matrix phasor_to_time(Matrix frequency_array, int oversample_exp)
{
//function [signal_array] = phasor_to_time(frequency_array,oversample_exp)
//The frequency array must include DC and the fundamental.

// we need to pick as sampling time that is high enough to prevent aliasing...
// I will oversample.  This helps prevent aliasing.  If there is a case where
// the non-linear device is saturating or whatever such that there are really
// lots of high frequency harmonics, one will want to set this oversample rate
// higher to accomodate or set the number of harmonics higher. This is set by
// setting oversample_exp (oversample exponent) thus if oversample_exp = 5, we
// will be oversampling by a factor of 2^5 = 32.
// Additionally,we want to include enough samples to get one full cycle of the
// fundamental.  Any more than this will be redundant information.

int L, i, k, f, f_highest,
    no_of_samples;
double T_fund, Ts;
double mag, phase;
double oversample_factor;                //The bigger the oversample factor, the less aliasing that will
                                         //occur, but the more time to computer stuff.
oversample_factor = 1 << oversample_exp; //make sure this is an even number.  Also, no smaller than 2.
L = frequency_array.RowNo();

if(L < 2)
{
        fprintf(stderr, "There must be at least two frequencies in the frequency_array\n");
}
f_highest = L-1;                         //since the first element is DC.
T_fund = 1;                              //period of the fundamental frequency which is normalized to 1.
Ts = 1.0/(f_highest*2.0*oversample_factor);  //Sample period.
no_of_samples = int(T_fund/Ts);         //This won't quite be a period of time.. One sample less.

Matrix mag_array(L,1);
Matrix phase_array(L,1);
Matrix signal_array(no_of_samples,1);
Matrix t_array(no_of_samples,1);
Matrix t2_array(no_of_samples,1);

signal_array.Null();

// create signal array                   //This block of code is faster than using the ifft method by something
on the order of 3.
for (i = 0;i<L;i++)
        mag_array(i,0) = abs(frequency_array(i,0));

for (i = 0;i<L;i++)
        phase_array(i,0) = arg(frequency_array(i,0));    //Note phase is in radians.

for (i = 0;i<no_of_samples;i++)
        t_array(i,0) = i;

t_array = complex <double> (Ts, 0)* t_array;

for (i = 0;i<L;i++)
{
        f = i;
        mag = mag_array(i,0).real();
        phase = phase_array(i,0).real();
    for (k = 0;k<no_of_samples;k++)
                t2_array(k,0) = mag*cos(2*pi*f*t_array(k,0).real() + phase);
        signal_array = signal_array + t2_array; //This uses about 30% of the total time.
}

        return signal_array;
}

Matrix time_to_phasor(Matrix signal_array, int freqs_size)
{
```

```
//function [frequency_array] = time_to_phasor(signal_array, no_of_freqs)
//Signal array is time domain row vector of the signal over one fundamental frequency.
//period.
//no_of_freqs includes DC and the fundamental frequency.  This number should at least be 2.
//The returned array contains the phasors of the various frequencies.  The first
//element is DC, the next is fundamental phasor, the next is second harmonic phasor..ect.
//Becareful about using this function.  You won't get the correct fourier series unless
//you have just the right number of time samples.  If you generated the signal array
//using pnasor_to_time, and you set no_of_freqs the same as the number of phasors you
//used at the time you made the time domain plot, you should get an exact representaion
//of the fourier series.

int no_of_samples,
    no_of_harmonics,
        step_size,
        b, a;

no_of_samples = signal_array.RowNo();           //number of points in one fundamental cycle
no_of_harmonics = freqs_size - 1;               //no_of_harmonics includes the fundamental but not DC.
step_size = ceil((no_of_samples-1.0)/(no_of_harmonics*4.0)); //this rounding thing makes sure we get an integer.
no_of_samples = floor(double(no_of_samples/step_size));

double *Re, *Im;
int dims[1] = {no_of_samples};

Re=(double *)calloc( no_of_samples, sizeof(double));
Im=(double *)calloc( no_of_samples, sizeof(double));

Matrix f(freqs_size,1);

b = 0;
a = 0;

for(a=0;a<no_of_samples;a++)
{
    Re[a] = signal_array(b,0).real();
        Im[a] = signal_array(b,0).imag();

        b = b + step_size;
}

fftn(1, dims, Re, Im, -1, 1);

for(a=0;a<freqs_size;a++)
{
        f(a,0) = complex <double> (Re[a], Im
[a]);
}

Matrix frequency_array(no_of_harmonics+1,1);

                                                                                                    //
The DC component isn't split in two to parts like the
frequency_array.Null();                                                     //rest of the dft is
frequency_array(0,0) = f(0,0);

for (a = 1;a<=no_of_harmonics;a++)
{
        frequency_array(a,0) = 2.0*f(a,0);                          //The factor of two is required
since we are pulling off of one of the phasors.
}

frequency_array = frequency_array/complex <double>(no_of_samples, 0); //This correct for the DFT scaling.

return frequency_array;
}

#ifdef WIN32

int exe(char *ExecuteFile, char *Parameters)
{
        static SHELLEXECUTEINFO ExecInfo;
        static unsigned long ExitCode;
        MSG msg;

    ExitCode = STILL_ACTIVE;
```

```
    ExecInfo.fMask = SEE_MASK_NOCLOSEPROCESS;
    ExecInfo.hwnd = NULL;
    ExecInfo.lpFile = ExecuteFile;
    ExecInfo.lpParameters = Parameters;
        ExecInfo.lpDirectory = NULL;
    ExecInfo.nShow = SW_HIDE;
        ExecInfo.hProcess = NULL;

    ExecInfo.cbSize = sizeof(ExecInfo);

   if (ShellExecuteEx(&ExecInfo))
   {
     while (ExitCode == STILL_ACTIVE)
         {
        if (GetExitCodeProcess(ExecInfo.hProcess, &ExitCode))
                {
                  if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
                  {
                        TranslateMessage(&msg);
                        DispatchMessage(&msg);
                  }
                }
                else ExitCode = 2;
     }
     if (ExitCode == 2) fprintf(stderr, "Cann't access process information\n");
         CloseHandle(ExecInfo.hProcess);
   }
   else fprintf(stderr, "file %s can't be started.\n",
                    ExecuteFile);

   return 0;
}

#endif

int save_intermediate(double freq)
{
Matrix V_array_node(no_of_freqs,1);
Matrix V_time_node;
int i, size, a, b, Err;
FILE  *pMatrixFile, *pModelsFile, *pDevicesFile;
char sInput[BUFSIZ];

size=V_array_time.RowNo();
V_time_node.SetSize(size, no_of_nodes);

V_array = Z_array*(-Ys_array*Vs_array - Ig_array);

for (a = 0;a<no_of_nodes;a++)
{
        for (b = 0;b<no_of_freqs;b++)
                V_array_node(b,0) = V_array(a*no_of_freqs+b,0);

        V_array_time = phasor_tc_time(V_array_node, 1);                 //  This uses about 40% of time.

        for (b = 0;b<size;b++)
                V_time_node(b,a) = V_array_time(b,0);

}

tiempo.SetSize(size, 1);

    for (i=0; i < size; i++)
                tiempo(i,0)=i/(freq*(size-1));

/* Open file in write mode. */
    if ((pMatrixFile = fopen("volt.cir","w")) == NULL)
        return 0;

        Err = fprintf(pMatrixFile, "Circuito No Lineal\n");
        Err = fprintf(pMatrixFile, ".tran %.15e %.15e\n",tiempo(1,0).real(),tiempo(size-1,0).real());

        for (a = 0; a < no_of_nodes; a++)
        {
                Err = fprintf(pMatrixFile, "V%d %d %d PWL(\n", portnmb[a+1], portnmb[a+1], portnmb[(no_of_nodes
```

```
+no_of_srces)*2+a+1]);

                for (b = 0; b< size; b++)
                {
                        Err = fprintf
                        (   pMatrixFile, "+\t%.15e\t%.15e\n",
                                tiempo(b,0).real(), V_time_node(b,a).real()
                        );
                }

                Err = fprintf(pMatrixFile, "+\t)\n");
    }

    pDevicesFile = fopen("devices.cir", "r");
        if (pMatrixFile == NULL)
        {   fprintf(stderr, "file %s was not found.\n",
                    "devices.cir");
            return 1;
        }

        while( !feof( pDevicesFile ) )
    {
        /* Read label. */
        fgets( sInput, BUFSIZ, pDevicesFile );
        if (!ferror( pDevicesFile ) && !feof( pDevicesFile ))
           Err = fprintf(pMatrixFile, sInput);
        }

        fclose(pDevicesFile);

        pModelsFile = fopen("models.cir", "r");
        if (pMatrixFile == NULL)
        {   fprintf(stderr, "file %s was not found.\n",
                    "models.cir");
            return 1;
        }

        while( !feof( pModelsFile ) )
    {
        /* Read label. */
        fgets( sInput, BUFSIZ, pModelsFile );
        if (!ferror( pModelsFile ) && !feof( pModelsFile ))
           Err = fprintf(pMatrixFile, sInput);
        }

        fclose(pModelsFile);

Err = fprintf(pMatrixFile, ".print tran");
for (a = 0; a <= no_of_nodes; a++)
{
        if (portnmb[a]!=0)
                Err = fprintf(pMatrixFile, " I(V%d)",portnmb[a]);
}
Err = fprintf(pMatrixFile, "\n");

Err = fprintf(pMatrixFile, ".end\n");

        if (Err < 0) return 0;

/* Close file. */
    if (fclose(pMatrixFile) < 0) return 0;

        return 0;
}

int load_intermediate(char *FileName)
{
        Matrix Ig_array_node(no_of_freqs,1);
    Matrix I_time_node;
        int size, Row;
        FILE  *pMatrixFile;
        char  sInput[BUFSIZ];
        double Real;

        I_time_node.SetSize(no_of_samples, no_of_nodes);
```

```
        //ReadVectorsFromFile("vectors.mat");

        pMatrixFile = fopen(FileName, "r");
        if (pMatrixFile == NULL)
        {   fprintf(stderr, "file %s was not found.\n",
                    FileName);
            return 1;
        }

        for (int a = 0;a<no_of_nodes;a++)
        {
    /* Read label. */
        fgets( sInput, BUFSIZ, pMatrixFile );

    /* Read size of matrix  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d", &size);

                for (int b = 0; b<size; b++)
                {
                        fgets( sInput, BUFSIZ, pMatrixFile );
                        sscanf( sInput,"%d%lf", &Row, &Real );

                        I_time_node(Row, a)= Real;
                }
    }

        fclose(pMatrixFile);

    for (a = 0;a<no_of_nodes;a++)
    {
        for (int b = 0; b<size; b++)
                Ig_array_time(b, 0) = -I_time_node(b, a);

        Ig_array_node = time_to_phasor(Ig_array_time, no_of_freqs);  // This uses about 22% of the time.

        for (b = 0;b<no_of_freqs;b++)
                Ig_array_nonlineal(a*no_of_freqs+b,0) = Ig_array_node(b,0);
    }

    Err = Ig_array_nonlineal - Ig_array;


        return 0;
    }

    int calculate_error_harm()
    {
    #ifndef WIN32

    Matrix V_array_node(no_of_freqs,1);
    Matrix Ig_array_node(no_of_freqs,1);

    V_array = Z_array*(-Ys_array*Vs_array - Ig_array);

    for (int a = 0;a<no_of_nodes;a++)
    {
        for (int b = 0;b<no_of_freqs;b++)
                V_array_node(b,0) = V_array(a*no_of_freqs+b,0);

        V_array_time = phasor_to_time(V_array_node, 1);                 // This uses about 40% of time.
        Ig_array_time = diode(V_array_time, 1);                                 // This uses about 10% of the
    time.
        Ig_array_node = time_to_phasor(Ig_array_time, no_of_freqs);  // This uses about 22% of the time.

        for (b = 0;b<no_of_freqs;b++)
                Ig_array_nonlineal(a*no_of_freqs+b,0) = Ig_array_node(b,0);
    }

    Err = Ig_array_nonlineal - Ig_array;

    #else
            save_intermediate(freq);
```

```
        exe("spice.exe","volt.cir");
        load_intermediate("vectors.mat");

#endif

        return 0;

}

int harmonic_balance_circuit1()
{

int a,b;
time_t cpu_time, cputime;
complex <double> delta_x, delta_y, der;
int success;

double alpha = 1;
int flag = 0;
double dither_factor = 0.001;

time( &cpu_time );
calculate_error_harm();

converged = 0;

Ig_array_old = Ig_array;
Err_old = Err;

err_sum_old = sum(Err);

if (counter==0)
   printf("initial values --> step_size=%0.4e err_sum=%4.3e\n\n", alpha, err_sum_old);

while(~flag)
{
        counter = counter + 1;
        for (a = 0;a<no_of_elements;a++)
        {
                delta_x = abs(Ig_array(a,0))*dither_factor;    //we change the independent variables.  We are
                Ig_array(a,0) = Ig_array(a,0)+delta_x;         //dithering each frequency component of Ig_array.
                                                          //Here we are saying delta_x = V_new - V_old;
                calculate_error_harm();                       //we now have calculated a new f2 (dependent
variable)

                for (b = 0;b<no_of_elements;b++)                 //We compute the change in error do to this one
input variable we varied.
                {
                        delta_y = Err(b,0)-Err_old(b,0);        //Here we are saying delta_y = Err_new - Err_old.
                        der = delta_y/delta_x;
                        J(b,a) = der;                                                    //The derivative are then
placed in the jacobian matrix.
                }
                Err = Err_old;
                Ig_array = Ig_array_old;
        }

        Ig_array = Ig_array_old - (J.Inv()*Err_old)*complex <double> (alpha,0);
        calculate_error_harm();

        err_sum = sum(Err);
        success = 0;
        if(err_sum < err_sum_old+1E-12)
        {
                alpha = alpha*1.1;
                if(alpha > 1)
                {
                        alpha = 0.95;
                }
                Err_old = Err;
                err_sum_old = err_sum;
                Ig_array_old = Ig_array;
                success = 1;
        }
        else
```

```
        {
                alpha = alpha*0.9;
                if(alpha > 1)
                {
                        alpha = alpha*0.9;
                }
                if(alpha < 1E-3)
                {
                        converged = 0;
                        break;
                }
                err_sum = err_sum_old;
                Err = Err_old;
                Ig_array = Ig_array_old;
                success = 0;
        }

//      if(rem(counter,5)==1)
//      {
//              printf('err_sum = %0.16e, alpha = %0.16e, success = %i',err_sum, alpha, success);
//              printf("\n");
//      }

        if(err_sum < error_margin)
        {
                converged = 1;
                break;
        }

}

time( &cputime );

printf( "The elapsed time is %5.2f secs\n", difftime( cputime, cpu_time ) );


        return 0;
}

int hd(Matrix signal_array)
{
//function hd(signal_array);
//This function outputs the harmonic distortion terms

int a, b, L;
double fund;

L = signal_array.RowNo();

for (b = 0; b<no_of_nodes; b++)
{
        printf("\nDC V(%i) = %0.15e\n", portnmb[b+1], abs(signal_array(b*no_of_freqs+1,0)));
        fund = abs(signal_array(b*no_of_freqs+2,0));
        printf("Fundamental V(%i) = %0.15e\n\n", portnmb[b+1], fund);
        printf("Harmonic distortion terms\n");

        for (a = 3; a<no_of_freqs; a++)
        {
                printf("HD%i = %0.15f",a-1,abs(signal_array(b*no_of_freqs+a,0))/fund);
                printf("\n");
        }
}

//cout << "\nV vector\n\n";
//cout << signal_array;

        return 0;
}

int spFileVector(char *File)
{

int i, j, Err;
FILE  *pMatrixFile;
B-250Matrix V_array_node;
Matrix V_time_node;
```

```
Matrix V_temp_time;

static char *Titulo = "Harmonic Balance Result";
static char *Grafico = "Plot in Time Domain";
static char *Banderas = "real";
static char *Version = "version 0.6";
int Variables = no_of_nodes+no_of_srces+1, Puntos;
time_t Fecha;

time( &Fecha );
Puntos = tiempo.RowNo();

/* Open File in write mode. */
    if ((pMatrixFile = fopen(File,"wb")) == NULL)
        return 0;

Err = fprintf(pMatrixFile, "Title: %s\n", Titulo);
Err = fprintf(pMatrixFile, "Date: %s", ctime(&Fecha));
Err = fprintf(pMatrixFile, "Plotname: %s\n", Grafico);
Err = fprintf(pMatrixFile, "Flags: %s\n", Banderas);
Err = fprintf(pMatrixFile, "No. Variables: %i\n", Variables);
Err = fprintf(pMatrixFile, "No. Points: %i\n", Puntos);
Err = fprintf(pMatrixFile, "Command: %s\n", Version);
Err = fprintf(pMatrixFile, "Variables:\n");
Err = fprintf(pMatrixFile, "\t0\ttime\ttime\n");
for (i = 1;i<Variables;i++)
        Err = fprintf(pMatrixFile, "\t%i\tV(%i)\tvoltage\n", i, portnmb[i]);
Err = fprintf(pMatrixFile, "Values:\n");

V_time_node.SetSize(Puntos, no_of_nodes);
V_array_node.SetSize(no_of_freqs, 1);

for (j = 0;j<no_of_nodes;j++)
{
        for (i = 0;i<no_of_freqs;i++)
                V_array_node(i,0) = V_array(j*no_of_freqs+i,0);

        V_temp_time = phasor_to_time(V_array_node,4);

        for (i = 0;i<Puntos;i++)
                V_time_node(i,j) = V_temp_time(i,0);

}

    for (j = 0; j< Puntos; j++)
    {
                Err = fprintf
        (   pMatrixFile, "%i\t\t%.15e\n",
            j, tiempo(j,0).real()
        );
                for (i = 0; i < no_of_nodes; i++)
                {
                        Err = fprintf
            (   pMatrixFile, "\t%.15e\n",
                                V_time_node(j,i).real()
            );
                }
                for (i = 0; i < no_of_srces; i++)
                {
                        Err = fprintf
                        (   pMatrixFile, "\t%.15e\n",
                                Vs_array_time(j,i).real()
                        );
                }
    }

        if (Err < 0) return 0;

/* Close file. */
    if (fclose(pMatrixFile) < 0) return 0;
    return 1;
}

int plot_time(double freq)
{
        int i, j, size;
```

```
        Matrix Vs_array_node;

    Vs_array_node.SetSize(no_of_freqs, 1);

        for (i=0; i < no_of_freqs; i++)
       Vs_array_node(i,0) = Vs_array(i,0);

        V_array_time = phasor_to_time(Vs_array_node,4);

    size=V_array_time.RowNo();

    Vs_array_time.SetSize(size, no_of_srces);
        Vs_array_time.Null();

        for (j = 0; j < no_of_srces; j++)
        {
                if (j>0)
                {
        for (i=0; i < no_of_freqs; i++)
                        Vs_array_node(i,0) = Vs_array(j*no_of_freqs+i,0);

            V_array_time = phasor_to_time(Vs_array_node, 4);
                }

            for (i=0; i < size; i++)
                        Vs_array_time(i,j) = V_array_time(i,0);
        }

    tiempo.SetSize(size, 1);

    for (i=0; i < size; i++)
                tiempo(i,0)=i/(freq*(size-1));

        //V_array_time = phasor_to_time(V_array,4);

        //plot_time(x,f,1);
        //title('V2 vs. Time (Harmonic Balance Results)');
        //ylabel('V2 (volts)');

        //plot_time(x,f,3);
        //title('V1 Signal Amplitude vs. Time')
        //ylabel('V1 (volts)');
        //clear x;
        //figure(2);
        //figure(1);

        spFileVector("rawspice.raw");

        return 0;
}

int circuit_harm(Matrix input, Matrix function1, Matrix function2, Matrix output)
{

printf("\n");

// initialize some arrays and variables.

time_t tttt, cpu_time;
int total_iterations;
int flag2;
int fail_flag;
double step_size;
double step_size_old;
double x1, x2, percentage, percentage_old;

Vs_array.SetSize(no_of_srcelmts,1);
V_array.SetSize(no_of_elements,1);
Vs_array_old.SetSize(no_of_elements,1);
V_array_time.SetSize(no_of_samples,1);
Vfinal_array.SetSize(no_of_srcelmts,1);
Y_array.SetSize(no_of_elements, no_of_elements);
Z_array.SetSize(no_of_elements, no_of_elements);
Ys_array.SetSize(no_of_elements, no_of_srcelmts);
```

```
Ig_array.SetSize(no_of_elements,1);
Ig_array_old.SetSize(no_of_elements,1);
Ig_array_nonlineal.SetSize(no_of_elements,1);
Ig_array_time.SetSize(no_of_samples,1);
Err.SetSize(no_of_elements,1);
Err_old.SetSize(no_of_elements,1);
J.SetSize(no_of_elements, no_of_elements);

int a, b, i;

Vs_array.Null();
V_array.Null();
Vfinal_array.Null();
Y_array.Null();
for (i=0; i < no_of_elements; i++)
        Ig_array(i,0) = input(i,0);
Ig_array_time.Null();
Err.Null();
J.Null();

converged = 0;

//******************** More user inputs ***************************
// These are the final input signal levels you desire.
for (i=0; i < no_of_srcelmts; i++)
        Vfinal_array(i,0) = output(i, 0);
//*****************************************************************
step_size = 1.0;            //Initial step size that simulator uses to step the signal levels.
                            //The simulator will set this automatically anyway.

// precalculate S for each frequency.

for (a = 0;a<no_of_elements;a++)
        for (b = 0;b<no_of_srcelmts;b++)
                Ys_array(a,b) = function1(a,b);

for (a = 0;a<no_of_elements;a++)
        for (b = 0;b<no_of_elements;b++)
                Y_array(a,b) = function2(a,b);

Z_array = Y_array.Inv();

time( &tttt );
printf( "The time is %s\n", ctime( &tttt ) );

//now we do the progressive loop that progressively increases the power of the harmonics.

error_margin = eps_iter;
Vs_array_old = Vs_array;
Ig_array_old = Ig_array;
total_iterations = 0;
flag2 = 1;
percentage_old = 0;
fail_flag = 0;
step_size_old = step_size;

while(flag2 == 1)
{
        Vs_array += complex <double> (step_size,0)*Vfinal_array;
        x1 = sum(Vs_array);
        x2 = sum(Vfinal_array);
        percentage = x1/x2*100;
            if(percentage >= 100)
                {
        Vs_array = Vfinal_array;
                flag2 = 0;
                percentage = 100;
                }

        harmonic_balance_circuit1();
        total_iterations = total_iterations + counter;

        if(converged != 1)
        {
                Vs_array = Vs_array_old;
                Ig_array = Ig_array_old;
```

```
                step_size = 0.1*step_size;

                if(step_size < 1E-8)
                {
                        fail_flag = 1;
                        break;
                }

                flag2 = 1;
        }
        else
        {
                step_size = step_size*1.05;
                Vs_array_old = Vs_array;
                Ig_array_old = Ig_array;
        }

        time(&cpu_time);

    if(converged)
        {
                printf("itterations=%3i  time=%5.2f secs step_size=%0.4e  percentage=%7.3f  convergence: PASS
err_sum=%4.3e",counter, difftime(cpu_time, tttt), step_size_old, percentage, err_sum);
                percentage_old = percentage;
        }
        else
        {
                printf("itterations=%3i  time=%5.2f secs step_size=%0.4e  percentage=%7.3f  convergence: FAIL
err_sum=%4.3e",counter, difftime(cpu_time, tttt), step_size_old, percentage_old, err_sum);
        }
                printf("\n");
                step_size_old = step_size;
}

    if(fail_flag)
        {
                printf("\n");
                printf("Simulation completely failed and was aborted...\n");
                printf("Increasing the number of harmonics should solve the problem.\n");
    }
        else
        {
                error_margin = eps_sol;
                harmonic_balance_circuit1();                    //This final run cleans up the results to
relatively accurate numbers.
                total_iterations = total_iterations + counter;
                printf("\n");
                printf("err_sum = %0.16e   Sum of currents = %0.16e",err_sum, (sum(Ig_array)+sum
(Ig_array_nonlineal)));
                printf("\n");

                time(&cpu_time);

                printf("Total expired system time %5.2f secs",difftime(cpu_time, tttt));
                printf("\n");

                printf("Total number of itterations is %i",total_iterations);
                printf("\n");

                hd(V_array);

                plot_time(freq);
          //plot_freq(V_array,f,2);
        }

   return 0;

}
```

```
/* Harmonic Balance Demo */

#include "matrix.h"

////////////////////////////////
// Note: The following conditional compilation statements are included
//       so that you can (likely to) compile this sample, without any
//       modification, using a compiler which does not support any or
//       all of the ANSI C++ features like NAMEPACE, TEMPLATE, and
//       EXCEPTION, used in this class.
//
//       If you have a compiler, such as C++ Builder, Borland C++ 5.0,
//       MS Visual C++ 5.0 or higher, etc., which supports most of the
ANSI
//       C++ features used in this class, you do not need to include these
//       statements in your program.
//
#ifndef _NO_NAMESPACE
using namespace std;
using namespace math;
#define STD std
#else
#define STD
#endif

#if !defined(_NO_TEMPLATE)
#  if defined(_MSC_VER)
#     if _MSC_VER > 1000
#        include <complex>
         typedef complex<double> type;
#        define __STDC__
#     else
         typedef double type;
#     endif
#  elif defined(__BORLANDC__)
#     if defined(__WIN32__)
#        include <complex>
         typedef complex<double> type;
#     else
#        include <complex.h>
         typedef complex type;
#     endif
#  elif defined( __GNUG__ )
#       include <complex>
        typedef complex<double> type;
# elif defined( _SGI_BROKEN_STL )
# include <complex>
typedef std::complex<double> type;
#  endif
  typedef matrix<type> Matrix;
#else
```

```
   typedef matrix Matrix;
#endif

#ifndef _NO_EXCEPTION
#  define TRYBEGIN()     try {
#  define CATCHERROR()   } catch (const STD::exception& e) { \
                      cerr << "Error: " << e.what() << endl; }
#else
#  define TRYBEGIN()
#  define CATCHERROR()
#endif

#ifdef __STDC__

double sum(Matrix);
Matrix diode(Matrix, int);
Matrix phasor_to_time(Matrix, int);
Matrix time_to_phasor(Matrix, int);
int calculate_error_harm();
int harmonic_balance_circuit1();
int hd(Matrix);
int circuit_harm(Matrix, Matrix, Matrix, Matrix);

#endif
```

```
//////////////////////////////////////////////////////
// File: main.cpp
//
// Purpose: Demonstrates how to use harmonic balance alghoritm
//

#include "hbdemo.h"
extern double pi;
extern double freq;
extern double eps_iter;
extern double eps_sol;
extern int no_of_freqs;
extern int no_of_nodes;
extern int no_of_srces;
extern int no_of_samples;
extern int no_of_elements;
extern int no_of_srcelmts;
extern int portnmb[];

#define  BOOLEAN  int
#define  NOT  !
#define  AND  &&
#define  OR  ||
#define  YES    1
#define  NO     0

static char  Message[BUFSIZ];
static BOOLEAN  Complex, SolutionOnly = NO;
static int  Size, MaxSize = 0;
static BOOLEAN ExpansionWarningGiven;
static complex <double> j=complex <double> (0,1);
Matrix Matriz;

int ReadMatrixFromFile(char *FileName)
{
long Reads;
FILE *pMatrixFile;
char  sInput[BUFSIZ], sType[BUFSIZ], *p;
int Row, Col, LineNumber, IntSize;
double Real, Imag = 0.0;
static char *EndOfFile = "%s: unexpected end of file `%s' at line %d.\n";
static char *Syntax = "%s: syntax error in file `%s' at line %d.\n";
static char *ProgramName = "hbsim";

/* Begin `ReadMatrixFromFile'. */

/* Open matrix file in read mode. */

    if (NOT SolutionOnly) putchar('\n');

    if (FileName == NULL)
    {   FileName = "standard input";
        pMatrixFile = stdin;
    }
    else
    {   pMatrixFile = fopen(FileName, "r");
        if (pMatrixFile == NULL)
        {   fprintf(stderr, "%s: file %s was not found.\n",
                    ProgramName, FileName);
            return 1;
```

```
            }
        }

    Complex = NO;
    LineNumber = 1;

    /* Read and print label. */
        if (NULL == fgets( Message, BUFSIZ, pMatrixFile ))
        {   fprintf(stderr, EndOfFile, ProgramName, FileName, LineNumber);
            return 1;
        }

    /* For compatibility with the old file syntax. */
        if (!strncmp( Message, "Starting", 8 ))
        {   /* Test for complex matrix. */
            if (strncmp( Message, "Starting complex", 15 ) == 0)
                Complex = YES;
            LineNumber++;
            if (NULL == fgets( Message, BUFSIZ, pMatrixFile ))
            {   fprintf(stderr, EndOfFile, ProgramName, FileName, LineNumber);
                return 1;
            }
        }

            if (NOT SolutionOnly) printf("%-s\n", Message);

    /* Read size of matrix and determine type of matrix. */
        LineNumber++;
        if (NULL == fgets( sInput, BUFSIZ, pMatrixFile ))
        {   fprintf(stderr, EndOfFile, ProgramName, FileName, LineNumber);
            return 1;
        }
        if ((Reads = sscanf( sInput,"%d %s", &Size, sType )) < 1)
        {   fprintf(stderr, Syntax, ProgramName, FileName, LineNumber);
            return 1;
        }
        if (Reads == 2)
        {   for (p = sType; *p != '\0'; p++)
                if (isupper(*p)) *p += 'a'-'A';
            if (strncmp( sType, "complex", 7 ) == 0)
                Complex = YES;
            else if (strncmp( sType, "real", 7 ) == 0)
                Complex = NO;
            else
            {   fprintf(stderr, Syntax, ProgramName, FileName, LineNumber);
                return 1;
            }
        }

    /* Read matrix elements. */
        do
        {
            LineNumber++;
            if (NULL == fgets( sInput, BUFSIZ, pMatrixFile ))
            {   fprintf(stderr, "%s: unexpected end of file `%s' at line %d.\n",
                        ProgramName, FileName, LineNumber);
                return 1;
            }

            if (Complex)
            {   Reads = sscanf( sInput,"%d%d%lf%lf", &Row, &Col, &Real, &Imag );
```

```
            if (Reads != 4)
            {   fprintf(stderr, "%s: syntax error in file `%s' at line %d.\n",
                        ProgramName, FileName, LineNumber);
                return 1;
            }
        }
        else
        {   Reads = sscanf( sInput,"%d%d%lf", &Row, &Col, &Real );
            if (Reads != 3)
            {   fprintf(stderr, "%s: syntax error in file `%s' at line %d.\n",
                        ProgramName, FileName, LineNumber);
                return 1;
            }
        }
        if(Row < 0 OR Col < 0)
        {   fprintf(stderr, "%s: index not positive in file `%s' at line %d.\n",
                        ProgramName, FileName, LineNumber);
            return 1;
        }
        if(Row > Size OR Col > Size)
        {   if (NOT ExpansionWarningGiven)
            {   fprintf( stderr,
             "%s: computed and given matrix size differ in file `%s' at line %d.\n",
                        ProgramName, FileName, LineNumber);
                ExpansionWarningGiven = YES;
            }
//           Size = MAX(Row, Col);
        }

 if (Row != 0 AND Col != 0) Matriz(Row-1,Col-1)= Real + Imag*j;


} while (Row != 0 AND Col != 0);


/* Print out the size and the type of the matrix. */
    if (NOT SolutionOnly)
    {   IntSize = Matriz.ColNo();
        printf("Matrix is %d x %d ", Size, IntSize);
        if (IntSize != Size)
            printf("(external size is %d x %d) ", Size, Size);
                IntSize = Matriz.RowNo();
        if (IntSize != Size)
            printf("(external size is %d x %d) ", IntSize, Size);
        if (Complex)
            printf("and complex.\n",Size,Size);
        else
            printf("and real.\n",Size,Size);
    }

    (void)fclose(pMatrixFile);
    return 0;

}

int main()
{

        Matrix input;
        Matrix function1;
        Matrix function2;
```

```
Matrix output;
//int a, b, c;
//double omega;

int n, k, s, Tipo;
double timestep, endtime;

int Row;
FILE  *pMatrixFile;
char  sInput[BUFSIZ], name[BUFSIZ], cktname[BUFSIZ];
int port;
double value, ivalue;

static char *FileName="analisis.opt";

        pMatrixFile = fopen(FileName, "r");
if (pMatrixFile == NULL)
{   fprintf(stderr, "file %s was not found.\n",
            FileName);
    return 1;
}


/* Read cktname  */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%s", &cktname);

/* Read n  */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%d%s", &n,&name);

/* Read s */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%d%s", &s, &name);

/* Read k */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%d%s", &k, &name);

/* Read freq  */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%lf%s", &freq, &name);

/* Read timestep  */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%lf%s", &timestep,&name);

/* Read endtime */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%lf%s", &endtime, &name);

/* Read Tipo */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%d%s", &Tipo, &name);

/* Read size of array of Ports  */
fgets( sInput, BUFSIZ, pMatrixFile );
sscanf( sInput,"%d%s", &Row, &name);

for (int i = 0;i<Row;i++)
```

```
    {
            fgets( sInput, BUFSIZ, pMatrixFile );
            sscanf( sInput,"%d", &port );

            portnmb[i]= port;
}

    output.SetSize(s*k,1);
    output.Null();

    for (int a = 0; a<s; a++)
    {

    /* Read size of array of VoltSrces  */
    fgets( sInput, BUFSIZ, pMatrixFile );
    sscanf( sInput,"%d%s", &Row, &name);

    //******************** More user inputs ***************************
    // These are the final input signal levels you desire.

    for (i = 0;i<(Row/2);i = i++)
    {
            fgets( sInput, BUFSIZ, pMatrixFile );
            sscanf( sInput,"%lf", &value );

            output(a*k+i,0) = value;

    fgets( sInput, BUFSIZ, pMatrixFile );
            sscanf( sInput,"%lf", &value );

            output(a*k+i,0) = output(a*k+i,0) + value*j;
}

    }

    fclose(pMatrixFile);

no_of_srces = s;
no_of_nodes = n-s;
    no_of_freqs = k;
no_of_samples = (no_of_freqs-1)*4;
no_of_elements = no_of_freqs*no_of_nodes;
no_of_srcelmts = no_of_freqs*no_of_srces;

    static char *FileSet="hbset.opt";

    pMatrixFile = fopen( strcat( cktname, ".opt" ), "r");
if (pMatrixFile == NULL)
    {
            pMatrixFile = fopen(FileSet, "r");
            if (pMatrixFile == NULL)
            {   fprintf(stderr, "file %s was not found.\n",
            FileName);
                    return 1;
            }
}

    /* Read eps_iter  */
    fgets( sInput, BUFSIZ, pMatrixFile );
    sscanf( sInput,"%lf%s", &eps_iter,&name);
```

```
    /* Read eps_sol  */
    fgets( sInput, BUFSIZ, pMatrixFile );
    sscanf( sInput,"%lf%s", &eps_sol, &name);

    /* Read size of array of CurrentInputs  */
    fgets( sInput, BUFSIZ, pMatrixFile );
    sscanf( sInput,"%d%s", &Row, &name);

    input.SetSize(no_of_elements,1);
    for (i=0; i < no_of_elements; i++)
            input(i,0) = 1E-6;

    //******************** More user inputs ****************************
    // These are the final input signal levels you desire.
    int ok = Row/(n-s);

    for (i = 0;i<Row;i = i++)
    {
            fgets( sInput, BUFSIZ, pMatrixFile );
            sscanf( sInput,"%lf%lf", &value, &ivalue );

            int no = i/ok;
            int ro = i - no*ok + no*k;

            input(ro,0) = value + ivalue*j;
/*
        fgets( sInput, BUFSIZ, pMatrixFile );
            sscanf( sInput,"%lf", &value );

            input(i,0) = input(i,0) + value*j;
    */
}

    fclose(pMatrixFile);

    function1.SetSize(no_of_elements, no_of_srcelmts);
    function2.SetSize(no_of_elements, no_of_elements);
    //output.SetSize(no_of_srcelmts,1);

    //******************** First set of user inputs ********************
    //double f = 1E3;                    ////fundamental frequency (Hz).
    //double R = 100;
    //double C = 0.1E-5;
    //*****************************************************************

    double w = 2*pi*freq;                //fundamental frequency (rads/sec).

    // precalculate Z for each frequency.

    Matriz.SetSize(no_of_elements, no_of_srcelmts);
    ReadMatrixFromFile("src.mat");
    function1 = Matriz;
    Matriz.SetSize(no_of_elements, no_of_elements);
ReadMatrixFromFile("mna.mat");
function2 = Matriz;

    //*****************************************************************
    //*** stuff for example 1 ***
    //f = 1E3;                     ////fundamental frequency (Hz).
```

```
        //no_of_freqs = 20;          ////This must include DC and fund..so no_of_freqs >= 2
        //R = 100;
        //C = 0.1*E-5;
        //Vfinal_array(1,0) = 2;
        //**************************

        //*** stuff for example 2 ***
        //f = 1E3;                   ////fundamental frequency (Hz).
        //no_of_freqs = 20;          ////This must include DC and fund..so no_of_freqs >= 2
        //R = 100;
        //C = 0.1E-5;
        //Vfinal_array(0,0) = 1.0;
        //Vfinal_array(1,0) = 2*exp(+j*pi/2);
        //Vfinal_array(2,0) = 2*exp(-j*pi/2);
        //**************************

        //**************************
        //*** stuff for example 3 ***
        //f = 1E3;                   ////fundamental frequency (Hz).
        //no_of_freqs = 20;          ////This must include DC and fund..so no_of_freqs >= 2
        //R = 100;
        //C = 0.1E-6;
        //Vfinal_array(0,0) = -1.0;
        //Vfinal_array(1,0) = 20;
        //Vfinal_array(2,0) = -5;
        //**************************

        /*
        for (a = 0;a<no_of_nodes;a++)
        {
                for (c = 0;c<no_of_srces;c++)
                {
                        for (b = 0;b<no_of_freqs;b++)
                        {
                                omega = w*b;
                                function1(a*no_of_freqs+b,c*no_of_freqs+b) = -1/R;
                        }
                }
        }

        for (a = 0;a<no_of_nodes;a++)
        {
                for (b = 0;b<no_of_freqs;b++)
                {
                        omega = w*b;
                        function2(a*no_of_freqs+b,a*no_of_freqs+b) = (1.0+(j*omega)*C*R)/R;
                }
        }
        */

        circuit_harm(input, function1, function2, output);

        return 0;
}
```

```
////////////////////////////////
// Matrix TCL Lite v1.13
// Copyright (c) 1997-2002 Techsoft Pvt. Ltd. (See License.Txt file.)
//
// Matrix.h: Matrix C++ template class include file
// Web: http://www.techsoftpl.com/matrix/
// Email: matrix@techsoftpl.com
//

////////////////////////////////
// Installation:
//
// Copy this "matrix.h" file into include directory of your compiler.
//

////////////////////////////////
// Note: This matrix template class defines majority of the matrix
// operations as overloaded operators or methods. It is assumed that
// users of this class is familiar with matrix algebra. We have not
// defined any specialization of this template here, so all the instances
// of matrix will be created implicitly by the compiler. The data types
// tested with this class are float, double, long double, complex<float>,
// complex<double> and complex<long double>. Note that this class is not
// optimized for performance.
//
// Since implementation of exception, namespace and template are still
// not standardized among the various (mainly old) compilers, you may
// encounter compilation error with some compilers. In that case remove
// any of the above three features by defining the following macros:
//
//  _NO_NAMESPACE:  Define this macro to remove namespace support.
//
//  _NO_EXCEPTION:  Define this macro to remove exception handling
//                  and use old style of error handling using function.
//
//  _NO_TEMPLATE:   If this macro is defined matrix class of double
//                  type will be generated by default. You can also
//                  generate a different type of matrix like float.
//
//  _SGI_BROKEN_STL: For SGI C++ v.7.2.1 compiler.
//
//  Since all the definitions are also included in this header file as
//  inline function, some compiler may give warning "inline function
//  can't be expanded". You may ignore/disable this warning using compiler
//  switches. All the operators/methods defined in this class have their
//  natural meaning except the followings:
//
//  Operator/Method                         Description
//  ---------------                         -----------
//   operator ()    :   This function operator can be used as a
//                      two-dimensional subscript operator to get/set
//                      individual matrix elements.
//
//   operator !     :   This operator has been used to calculate inversion
//                      of matrix.
//
//   operator ~     :   This operator has been used to return transpose of
//                      a matrix.
//
//   operator ^     :   It is used calculate power (by a scalar) of a matrix.
//                      When using this operator in a matrix equation, care
```

```
//                      must be taken by parenthesizing it because it has
//                      lower precedence than addition, subtraction,
//                      multiplication and division operators.
//
//   operator >>   :    It is used to read matrix from input stream as per
//                      standard C++ stream operators.
//
//   operator <<   :    It is used to write matrix to output stream as per
//                      standard C++ stream operators.
//
// Note that professional version of this package, Matrix TCL Pro 2.11
// is optimized for performance and supports many more matrix operations.
// It is available from our web site at <http://www.techsoftpl.com/matrix/>.
//

#ifndef __cplusplus
#error Must use C++ for the type matrix.
#endif

#if !defined(__STD_MATRIX_H)
#define __STD_MATRIX_H

//////////////////////////////
// First deal with various shortcomings and incompatibilities of
// various (mainly old) versions of popular compilers available.
//

#if defined(__BORLANDC__)
#pragma option -w-inl -w-pch
#endif

#if ( defined(__BORLANDC__) || _MSC_VER <= 1000 ) && !defined( __GNUG__ )
#  include <stdio.h>
#  include <stdlib.h>
#  include <math.h>
#  include <iostream.h>
#  include <string.h>
#else
#  include <cmath>
#  include <cstdio>
#  include <cstdlib>
#  include <string>
#  include <iostream>
#endif

#if defined(_MSC_VER) && _MSC_VER <= 1000
#  define _NO_EXCEPTION        // stdexception is not fully supported in MSVC++ 4.0
typedef int bool;
#  if !defined(false)
#    define false  0
#  endif
#  if !defined(true)
#    define true   1
#  endif
#endif

#if defined(__BORLANDC__) && !defined(__WIN32__)
#  define _NO_EXCEPTION        // std exception and namespace are not fully
#  define _NO_NAMESPACE        // supported in 16-bit compiler
#endif

#if defined(_MSC_VER) && !defined(_WIN32)
```

```
#  define _NO_EXCEPTION
#endif

#if defined(_NO_EXCEPTION)
#  define _NO_THROW
#  define _THROW_MATRIX_ERROR
#else
#  if defined(_MSC_VER)
#    if _MSC_VER >= 1020
#      include <stdexcept>
#    else
#      include <stdexcpt.h>
#    endif
#  elif defined(__MWERKS__)
#      include <stdexcept>
#  elif (__GNUC__ >= 2 || (__GNUC__ == 2 && __GNUC_MINOR__ >= 8))
#     include <stdexcept>
#  else
#     include <stdexcep>
#  endif
#  define _NO_THROW               throw ()
#  define _THROW_MATRIX_ERROR     throw (matrix_error)
#endif

#ifndef __MINMAX_DEFINED
#  define max(a,b)     (((a) > (b)) ? (a) : (b))
#  define min(a,b)     (((a) < (b)) ? (a) : (b))
#endif

#if defined(_MSC_VER)
#undef _MSC_EXTENSIONS      // To include overloaded abs function definitions!
#endif

#if ( defined(__BORLANDC__) || _MSC_VER ) && !defined( __GNUG__ )
inline float abs (float v) { return (float)fabs( v); }
inline double abs (double v) { return fabs( v); }
inline long double abs (long double v) { return fabsl( v); }
#endif

#if defined(__GNUG__) || defined(__MWERKS__) || (defined(__BORLANDC__) && (__BORLANDC__ >=
0x540))
#define FRIEND_FUN_TEMPLATE <>
#else
#define FRIEND_FUN_TEMPLATE
#endif

#if defined(_MSC_VER) && _MSC_VER <= 1020    // MSVC++ 4.0/4.2 does not
#  define _NO_NAMESPACE                      // support "std" namespace
#endif

#if !defined(_NO_NAMESPACE)
#if defined( _SGI_BROKEN_STL )               // For SGI C++ v.7.2.1 compiler
namespace std { }
#endif
using namespace std;
#endif

#ifndef _NO_NAMESPACE
namespace math {
#endif
```

```
#if !defined(_NO_EXCEPTION)
class matrix_error : public logic_error
{
    public:
        matrix_error (const string& what_arg) : logic_error( what_arg) {}
};
#define REPORT_ERROR(ErrormMsg)  throw matrix_error( ErrormMsg);
#else
inline void _matrix_error (const char* pErrMsg)
{
    cout << pErrMsg << endl;
    exit(1);
}
#define REPORT_ERROR(ErrormMsg)  _matrix_error( ErrormMsg);
#endif

#if !defined(_NO_TEMPLATE)
#  define MAT_TEMPLATE  template <class T>
#  define matrixT  matrix<T>
#else
#  define MAT_TEMPLATE
#  define matrixT  matrix
#  ifdef MATRIX_TYPE
     typedef MATRIX_TYPE T;
#  else
     typedef double T;
#  endif
#endif


MAT_TEMPLATE
class matrix
{
public:
   // Constructors
   matrix (const matrixT& m);
   matrix (size_t row = 6, size_t col = 6);

   // Destructor
   ~matrix ();

   // Assignment operators
   matrixT& operator = (const matrixT& m) _NO_THROW;

   // Value extraction method
   size_t RowNo () const { return _m->Row; }
   size_t ColNo () const { return _m->Col; }

   // Subscript operator
   T& operator () (size_t row, size_t col) _THROW_MATRIX_ERROR;
   T  operator () (size_t row, size_t col) const _THROW_MATRIX_ERROR;

   // Unary operators
   matrixT operator + () _NO_THROW { return *this; }
   matrixT operator - () _NO_THROW;

   // Combined assignment - calculation operators
   matrixT& operator += (const matrixT& m) _THROW_MATRIX_ERROR;
   matrixT& operator -= (const matrixT& m) _THROW_MATRIX_ERROR;
   matrixT& operator *= (const matrixT& m) _THROW_MATRIX_ERROR;
```

```
    matrixT& operator *= (const T& c) _NO_THROW;
    matrixT& operator /= (const T& c) _NO_THROW;
    matrixT& operator ^= (const size_t& pow) _THROW_MATRIX_ERROR;

    // Miscellaneous -methods
    void Null (const size_t& row, const size_t& col) _NO_THROW;
    void Null () _NO_THROW;
    void Unit (const size_t& row) _NO_THROW;
    void Unit () _NO_THROW;
    void SetSize (size_t row, size_t col) _NO_THROW;

    // Utility methods
    matrixT Solve (const matrixT& v) const _THROW_MATRIX_ERROR;
    matrixT Adj () _THROW_MATRIX_ERROR;
    matrixT Inv () _THROW_MATRIX_ERROR;
    T Det () const _THROW_MATRIX_ERROR;
    T Norm () _NO_THROW;
    T Cofact (size_t row, size_t col) _THROW_MATRIX_ERROR;
    T Cond () _NO_THROW;

    // Type of matrices
    bool IsSquare () _NO_THROW { return (_m->Row == _m->Col); }
    bool IsSingular () _NO_THROW;
    bool IsDiagonal () _NO_THROW;
    bool IsScalar () _NO_THROW;
    bool IsUnit () _NO_THROW;
    bool IsNull () _NO_THROW;
    bool IsSymmetric () _NO_THROW;
    bool IsSkewSymmetric () _NO_THROW;
    bool IsUpperTriangular () _NO_THROW;
    bool IsLowerTriangular () _NO_THROW;

private:
    struct base_mat
    {
        T **Val;
        size_t Row, Col, RowSiz, ColSiz;
        int Refcnt;

        base_mat (size_t row, size_t col, T** v)
        {
            Row = row; RowSiz = row;
            Col = col; ColSiz = col;
            Refcnt = 1;

            Val = new T* [row];
            size_t rowlen = col * sizeof(T);

            for (size_t i=0; i < row; i++)
            {
                Val[i] = new T [col];
                if (v) memcpy( Val[i], v[i], rowlen);
            }
        }
        ~base_mat ()
        {
            for (size_t i=0; i < RowSiz; i++)
                delete [] Val[i];
            delete [] Val;
        }
    };
```

```
    base_mat *_m;

    void clone ();
    void realloc (size_t row, size_t col);
    int pivot (size_t row);
};

#if defined(_MSC_VER) && _MSC_VER <= 1020
#  undef  _NO_THROW              // MSVC++ 4.0/4.2 does not support
#  undef  _THROW_MATRIX_ERROR    // exception specification in definition
#  define _NO_THROW
#  define _THROW_MATRIX_ERROR
#endif

// constructor
MAT_TEMPLATE inline
matrixT::matrix (size_t row, size_t col)
{
  _m = new base_mat( row, col, 0);
}

// copy constructor
MAT_TEMPLATE inline
matrixT::matrix (const matrixT& m)
{
    _m = m._m;
    _m->Refcnt++;
}

// Internal copy constructor
MAT_TEMPLATE inline void
matrixT::clone ()
{
    _m->Refcnt--;
    _m = new base_mat( _m->Row, _m->Col, _m->Val);
}

// destructor
MAT_TEMPLATE inline
matrixT::~matrix ()
{
   if (--_m->Refcnt == 0) delete _m;
}

// assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator = (const matrixT& m) _NO_THROW
{
    m._m->Refcnt++;
    if (--_m->Refcnt == 0) delete _m;
    _m = m._m;
    return *this;
}

//  reallocation method
MAT_TEMPLATE inline void
matrixT::realloc (size_t row, size_t col)
{
   if (row == _m->RowSiz && col == _m->ColSiz)
   {
      _m->Row = _m->RowSiz;
```

```
      _m->Col = _m->ColSiz;
      return;
   }

   base_mat *m1 = new base_mat( row, col, NULL);
   size_t colSize = min(_m->Col,col) * sizeof(T);
   size_t minRow = min(_m->Row,row);

   for (size_t i=0; i < minRow; i++)
      memcpy( m1->Val[i], _m->Val[i], colSize);

   if (--_m->Refcnt == 0)
       delete _m;
   _m = m1;

   return;
}

// public method for resizing matrix
MAT_TEMPLATE inline void
matrixT::SetSize (size_t row, size_t col) _NO_THROW
{
   size_t i,j;
   size_t oldRow = _m->Row;
   size_t oldCol = _m->Col;

   if (row != _m->RowSiz || col != _m->ColSiz)
      realloc( row, col);

   for (i=oldRow; i < row; i++)
      for (j=0; j < col; j++)
         _m->Val[i][j] = T(0);

   for (i=0; i < row; i++)
      for (j=oldCol; j < col; j++)
         _m->Val[i][j] = T(0);

   return;
}

// subscript operator to get/set individual elements
MAT_TEMPLATE inline T&
matrixT::operator () (size_t row, size_t col) _THROW_MATRIX_ERROR
{
   if (row >= _m->Row || col >= _m->Col)
      REPORT_ERROR( "matrixT::operator(): Index out of range!");
   if (_m->Refcnt > 1) clone();
   return _m->Val[row][col];
}

// subscript operator to get/set individual elements
MAT_TEMPLATE inline T
matrixT::operator () (size_t row, size_t col) const _THROW_MATRIX_ERROR
{
   if (row >= _m->Row || col >= _m->Col)
      REPORT_ERROR( "matrixT::operator(): Index out of range!");
   return _m->Val[row][col];
}

// input stream function
MAT_TEMPLATE inline istream&
```

```
operator >> (istream& istrm, matrixT& m)
{
   for (size_t i=0; i < m.RowNo(); i++)
      for (size_t j=0; j < m.ColNo(); j++)
      {
         T x;
         istrm >> x;
         m(i,j) = x;
      }
   return istrm;
}

// output stream function
MAT_TEMPLATE inline ostream&
operator << (ostream& ostrm, const matrixT& m)
{
   for (size_t i=0; i < m.RowNo(); i++)
   {
      for (size_t j=0; j < m.ColNo(); j++)
      {
         T x = m(i,j);
         ostrm << x << '\t';
      }
      ostrm << endl;
   }
   return ostrm;
}


// logical equal-to operator
MAT_TEMPLATE inline bool
operator == (const matrixT& m1, const matrixT& m2) _NO_THROW
{
   if (m1.RowNo() != m2.RowNo() || m1.ColNo() != m2.ColNo())
      return false;

   for (size_t i=0; i < m1.RowNo(); i++)
      for (size_t j=0; j < m1.ColNo(); j++)
            if (m1(i,j) != m2(i,j))
               return false;

   return true;
}

// logical no-equal-to operator
MAT_TEMPLATE inline bool
operator != (const matrixT& m1, const matrixT& m2) _NO_THROW
{
    return (m1 == m2) ? false : true;
}

// combined addition and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator += (const matrixT& m) _THROW_MATRIX_ERROR
{
   if (_m->Row != m._m->Row || _m->Col != m._m->Col)
      REPORT_ERROR( "matrixT::operator+= : Inconsistent matrix sizes in addition!");
   if (_m->Refcnt > 1) clone();
   for (size_t i=0; i < m._m->Row; i++)
      for (size_t j=0; j < m._m->Col; j++)
         _m->Val[i][j] += m._m->Val[i][j];
```

```
   return *this;
}

// combined subtraction and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator -= (const matrixT& m) _THROW_MATRIX_ERROR
{
   if (_m->Row != m._m->Row || _m->Col != m._m->Col)
      REPORT_ERROR( "matrixT::operator-= : Inconsistent matrix sizes in subtraction!");
   if (_m->Refcnt > 1) clone();
   for (size_t i=0; i < m._m->Row; i++)
      for (size_t j=0; j < m._m->Col; j++)
         _m->Val[i][j] -= m._m->Val[i][j];
   return *this;
}

// combined scalar multiplication and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator *= (const T& c) _NO_THROW
{
    if (_m->Refcnt > 1) clone();
    for (size_t i=0; i < _m->Row; i++)
        for (size_t j=0; j < _m->Col; j++)
            _m->Val[i][j] *= c;
    return *this;
}

// combined matrix multiplication and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator *= (const matrixT& m) _THROW_MATRIX_ERROR
{
   if (_m->Col != m._m->Row)
      REPORT_ERROR( "matrixT::operator*= : Inconsistent matrix sizes in multiplication!");

   matrixT temp(_m->Row,m._m->Col);

   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < m._m->Col; j++)
      {
         temp._m->Val[i][j] = T(0);
         for (size_t k=0; k < _m->Col; k++)
            temp._m->Val[i][j] += _m->Val[i][k] * m._m->Val[k][j];
      }
   *this = temp;

   return *this;
}

// combined scalar division and assignment operator
MAT_TEMPLATE inline matrixT&
matrixT::operator /= (const T& c) _NO_THROW
{
    if (_m->Refcnt > 1) clone();
    for (size_t i=0; i < _m->Row; i++)
        for (size_t j=0; j < _m->Col; j++)
            _m->Val[i][j] /= c;

    return *this;
}

// combined power and assignment operator
```

```
MAT_TEMPLATE inline matrixT&
matrixT::operator ^= (const size_t& pow) _THROW_MATRIX_ERROR
{
        matrixT temp(*this);

        for (size_t i=2; i <= pow; i++)
      *this = *this * temp;

        return *this;
}

// unary negation operator
MAT_TEMPLATE inline matrixT
matrixT::operator - () _NO_THROW
{
   matrixT temp(_m->Row,_m->Col);

   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
         temp._m->Val[i][j] = - _m->Val[i][j];

   return temp;
}

// binary addition operator
MAT_TEMPLATE inline matrixT
operator + (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
   matrixT temp = m1;
   temp += m2;
   return temp;
}

// binary subtraction operator
MAT_TEMPLATE inline matrixT
operator - (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
   matrixT temp = m1;
   temp -= m2;
   return temp;
}

// binary scalar multiplication operator
MAT_TEMPLATE inline matrixT
operator * (const matrixT& m, const T& no) _NO_THROW
{
   matrixT temp = m;
   temp *= no;
   return temp;
}


// binary scalar multiplication operator
MAT_TEMPLATE inline matrixT
operator * (const T& no, const matrixT& m) _NO_THROW
{
   return (m * no);
}

// binary matrix multiplication operator
MAT_TEMPLATE inline matrixT
```

```
operator * (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
   matrixT temp = m1;
   temp *= m2;
   return temp;
}

// binary scalar division operator
MAT_TEMPLATE inline matrixT
operator / (const matrixT& m, const T& no) _NO_THROW
{
    return (m * (T(1) / no));
}


// binary scalar division operator
MAT_TEMPLATE inline matrixT
operator / (const T& no, const matrixT& m) _THROW_MATRIX_ERROR
{
    return (!m * no);
}

// binary matrix division operator
MAT_TEMPLATE inline matrixT
operator / (const matrixT& m1, const matrixT& m2) _THROW_MATRIX_ERROR
{
    return (m1 * !m2);
}

// binary power operator
MAT_TEMPLATE inline matrixT
operator ^ (const matrixT& m, const size_t& pow) _THROW_MATRIX_ERROR
{
   matrixT temp = m;
   temp ^= pow;
   return temp;
}

// unary transpose operator
MAT_TEMPLATE inline matrixT
operator ~ (const matrixT& m) _NO_THROW
{
   matrixT temp(m.ColNo(),m.RowNo());

   for (size_t i=0; i < m.RowNo(); i++)
      for (size_t j=0; j < m.ColNo(); j++)
      {
         T x = m(i,j);
              temp(j,i) = x;
      }
   return temp;
}

// unary inversion operator
MAT_TEMPLATE inline matrixT
operator ! (const matrixT m) _THROW_MATRIX_ERROR
{
   matrixT temp = m;
   return temp.Inv();
}
```

```
// inversion function
MAT_TEMPLATE inline matrixT
matrixT::Inv () _THROW_MATRIX_ERROR
{
   size_t i,j,k;
   T a1,a2,*rowptr;

   if (_m->Row != _m->Col)
      REPORT_ERROR( "matrixT::operator!: Inversion of a non-square matrix");

   matrixT temp(_m->Row,_m->Col);
   if (_m->Refcnt > 1) clone();


   temp.Unit();
   for (k=0; k < _m->Row; k++)
   {
      int indx = pivot(k);
      if (indx == -1)
              REPORT_ERROR( "matrixT::operator!: Inversion of a singular matrix");

      if (indx != 0)
      {
              rowptr = temp._m->Val[k];
              temp._m->Val[k] = temp._m->Val[indx];
              temp._m->Val[indx] = rowptr;
      }
      a1 = _m->Val[k][k];
      for (j=0; j < _m->Row; j++)
      {
              _m->Val[k][j] /= a1;
              temp._m->Val[k][j] /= a1;
      }
      for (i=0; i < _m->Row; i++)
           if (i != k)
           {
              a2 = _m->Val[i][k];
              for (j=0; j < _m->Row; j++)
              {
                 _m->Val[i][j] -= a2 * _m->Val[k][j];
                 temp._m->Val[i][j] -= a2 * temp._m->Val[k][j];
              }
           }
   }
   return temp;
}

// solve simultaneous equation
MAT_TEMPLATE inline matrixT
matrixT::Solve (const matrixT& v) const _THROW_MATRIX_ERROR
{
   size_t i,j,k;
   T a1;

   if (!(_m->Row == _m->Col && _m->Col == v._m->Row))
      REPORT_ERROR( "matrixT::Solve():Inconsistent matrices!");

   matrixT temp(_m->Row,_m->Col+v._m->Col);
   for (i=0; i < _m->Row; i++)
   {
      for (j=0; j < _m->Col; j++)
         temp._m->Val[i][j] = _m->Val[i][j];
```

```
            for (k=0; k < v._m->Col; k++)
               temp._m->Val[i][_m->Col+k] = v._m->Val[i][k];
         }
         for (k=0; k < _m->Row; k++)
         {
            int indx = temp.pivot(k);
            if (indx == -1)
               REPORT_ERROR( "matrixT::Solve(): Singular matrix!");

            a1 = temp._m->Val[k][k];
            for (j=k; j < temp._m->Col; j++)
               temp._m->Val[k][j] /= a1;

            for (i=k+1; i < _m->Row; i++)
            {
               a1 = temp._m->Val[i][k];
               for (j=k; j < temp._m->Col; j++)
                 temp._m->Val[i][j] -= a1 * temp._m->Val[k][j];
            }
         }
         matrixT s(v._m->Row,v._m->Col);
         for (k=0; k < v._m->Col; k++)
            for (int m=int(_m->Row)-1; m >= 0; m--)
            {
               s._m->Val[m][k] = temp._m->Val[m][_m->Col+k];
               for (j=m+1; j < _m->Col; j++)
                  s._m->Val[m][k] -= temp._m->Val[m][j] * s._m->Val[j][k];
            }
         return s;
      }

      // set zero to all elements of this matrix
      MAT_TEMPLATE inline void
      matrixT::Null (const size_t& row, const size_t& col) _NO_THROW
      {
          if (row != _m->Row || col != _m->Col)
              realloc( row,col);

          if (_m->Refcnt > 1)
              clone();

          for (size_t i=0; i < _m->Row; i++)
              for (size_t j=0; j < _m->Col; j++)
                  _m->Val[i][j] = T(0);
          return;
      }

      // set zero to all elements of this matrix
      MAT_TEMPLATE inline void
      matrixT::Null() _NO_THROW
      {
          if (_m->Refcnt > 1) clone();
          for (size_t i=0; i < _m->Row; i++)
              for (size_t j=0; j < _m->Col; j++)
                      _m->Val[i][j] = T(0);
          return;
      }

      // set this matrix to unity
      MAT_TEMPLATE inline void
      matrixT::Unit (const size_t& row) _NO_THROW
```

```
{
    if (row != _m->Row || row != _m->Col)
        realloc( row, row);

    if (_m->Refcnt > 1)
        clone();

    for (size_t i=0; i < _m->Row; i++)
        for (size_t j=0; j < _m->Col; j++)
            _m->Val[i][j] = i == j ? T(1) : T(0);
    return;
}

// set this matrix to unity
MAT_TEMPLATE inline void
matrixT::Unit () _NO_THROW
{
    if (_m->Refcnt > 1) clone();
    size_t row = min(_m->Row,_m->Col);
    _m->Row = _m->Col = row;

    for (size_t i=0; i < _m->Row; i++)
        for (size_t j=0; j < _m->Col; j++)
            _m->Val[i][j] = i == j ? T(1) : T(0);
    return;
}

// private partial pivoting method
MAT_TEMPLATE inline int
matrixT::pivot (size_t row)
{
  int k = int(row);
  double amax,temp;

  amax = -1;
  for (size_t i=row; i < _m->Row; i++)
    if ( (temp = abs( _m->Val[i][row])) > amax && temp != 0.0)
     {
       amax = temp;
       k = i;
     }
  if (_m->Val[k][row] == T(0))
     return -1;
  if (k != int(row))
  {
     T* rowptr = _m->Val[k];
     _m->Val[k] = _m->Val[row];
     _m->Val[row] = rowptr;
     return k;
  }
  return 0;
}

// calculate the determinant of a matrix
MAT_TEMPLATE inline T
matrixT::Det () const _THROW_MATRIX_ERROR
{
   size_t i,j,k;
   T piv,detVal = T(1);

   if (_m->Row != _m->Col)
```

```
      REPORT_ERROR( "matrixT::Det(): Determinant a non-square matrix!");

   matrixT temp(*this);
   if (temp._m->Refcnt > 1) temp.clone();

   for (k=0; k < _m->Row; k++)
   {
      int indx = temp.pivot(k);
      if (indx == -1)
         return 0;
      if (indx != 0)
         detVal = - detVal;
      detVal = detVal * temp._m->Val[k][k];
      for (i=k+1; i < _m->Row; i++)
      {
         piv = temp._m->Val[i][k] / temp._m->Val[k][k];
         for (j=k+1; j < _m->Row; j++)
            temp._m->Val[i][j] -= piv * temp._m->Val[k][j];
      }
   }
   return detVal;
}

// calculate the norm of a matrix
MAT_TEMPLATE inline T
matrixT::Norm () _NO_THROW
{
   T retVal = T(0);

   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
         retVal += _m->Val[i][j] * _m->Val[i][j];
   retVal = sqrt( retVal);

   return retVal;
}

// calculate the condition number of a matrix
MAT_TEMPLATE inline T
matrixT::Cond () _NO_THROW
{
   matrixT inv = ! (*this);
   return (Norm() * inv.Norm());
}

// calculate the cofactor of a matrix for a given element
MAT_TEMPLATE inline T
matrixT::Cofact (size_t row, size_t col) _THROW_MATRIX_ERROR
{
   size_t i,i1,j,j1;

   if (_m->Row != _m->Col)
      REPORT_ERROR( "matrixT::Cofact(): Cofactor of a non-square matrix!");

   if (row > _m->Row || col > _m->Col)
      REPORT_ERROR( "matrixT::Cofact(): Index out of range!");

   matrixT temp (_m->Row-1, _m->Col-1);

   for (i=i1=0; i < _m->Row; i++)
   {
```

```
        if (i == row)
          continue;
        for (j=j1=0; j < _m->Col; j++)
        {
           if (j == col)
              continue;
           temp._m->Val[i1][j1] = _m->Val[i][j];
           j1++;
        }
        i1++;
     }
     T  cof = temp.Det();
     if ((row+col)%2 == 1)
        cof = -cof;

     return cof;
  }


  // calculate adjoin of a matrix
  MAT_TEMPLATE inline matrixT
  matrixT::Adj () _THROW_MATRIX_ERROR
  {
     if (_m->Row != _m->Col)
        REPORT_ERROR( "matrixT::Adj(): Adjoin of a non-square matrix.");

     matrixT temp(_m->Row,_m->Col);

     for (size_t i=0; i < _m->Row; i++)
        for (size_t j=0; j < _m->Col; j++)
            temp._m->Val[j][i] = Cofact(i,j);
     return temp;
  }

  // Determine if the matrix is singular
  MAT_TEMPLATE inline bool
  matrixT::IsSingular () _NO_THROW
  {
     if (_m->Row != _m->Col)
        return false;
     return (Det() == T(0));
  }

  // Determine if the matrix is diagonal
  MAT_TEMPLATE inline bool
  matrixT::IsDiagonal () _NO_THROW
  {
     if (_m->Row != _m->Col)
        return false;
     for (size_t i=0; i < _m->Row; i++)
       for (size_t j=0; j < _m->Col; j++)
          if (i != j && _m->Val[i][j] != T(0))
            return false;
     return true;
  }

  // Determine if the matrix is scalar
  MAT_TEMPLATE inline bool
  matrixT::IsScalar () _NO_THROW
  {
     if (!IsDiagonal())
```

```
      return false;
   T v = _m->Val[0][0];
   for (size_t i=1; i < _m->Row; i++)
      if (_m->Val[i][i] != v)
         return false;
   return true;
}

// Determine if the matrix is a unit matrix
MAT_TEMPLATE inline bool
matrixT::IsUnit () _NO_THROW
{
   if (IsScalar() && _m->Val[0][0] == T(1))
      return true;
   return false;
}

// Determine if this is a null matrix
MAT_TEMPLATE inline bool
matrixT::IsNull () _NO_THROW
{
   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
         if (_m->Val[i][j] != T(0))
            return false;
   return true;
}

// Determine if the matrix is symmetric
MAT_TEMPLATE inline bool
matrixT::IsSymmetric () _NO_THROW
{
   if (_m->Row != _m->Col)
      return false;
   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
         if (_m->Val[i][j] != _m->Val[j][i])
            return false;
   return true;
}

// Determine if the matrix is skew-symmetric
MAT_TEMPLATE inline bool
matrixT::IsSkewSymmetric () _NO_THROW
{
   if (_m->Row != _m->Col)
      return false;
   for (size_t i=0; i < _m->Row; i++)
      for (size_t j=0; j < _m->Col; j++)
         if (_m->Val[i][j] != -_m->Val[j][i])
            return false;
   return true;
}

// Determine if the matrix is upper triangular
MAT_TEMPLATE inline bool
matrixT::IsUpperTriangular () _NO_THROW
{
   if (_m->Row != _m->Col)
      return false;
   for (size_t i=1; i < _m->Row; i++)
```

```
      for (size_t j=0; j < i-1; j++)
         if (_m->Val[i][j] != T(0))
            return false;
   return true;
}

// Determine if the matrix is lower triangular
MAT_TEMPLATE inline bool
matrixT::IsLowerTriangular () _NO_THROW
{
   if (_m->Row != _m->Col)
      return false;

   for (size_t j=1; j < _m->Col; j++)
      for (size_t i=0; i < j-1; i++)
         if (_m->Val[i][j] != T(0))
            return false;

   return true;
}

#ifndef _NO_NAMESPACE
}
#endif

#endif //__STD_MATRIX_H
```

C/C++ Source Files

/

- niaciter.cpp
- postcoms.cpp
- spsmp.cpp

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/

    /*
     * NIacIter(ckt)
     *
     *  This subroutine performs the actual numerical iteration.
     *  It uses the sparse matrix stored in the NIstruct by NIinit,
     *  along with the matrix loading program, the load data, the
     *  convergence test function, and the convergence parameters
     * - return value is non-zero for convergence failure
     */

#define BALANCE_ARMONICO

#include "spice.h"
#include <stdio.h>
#include "trandefs.h"
#include "cktdefs.h"
#include "util.h"
#include "sperror.h"
#include "suffix.h"

int
NIacIter(ckt)
    register CKTcircuit * ckt;
{
    int error;
    int ignore;
    double *temp;

#ifdef BALANCE_ARMONICO /* this lines were added by me */

        SMPmatrix **CKTmatriz, **balmny, **balmns;
        SMPmatrix *balmna, *balmnz;
        SMPmatrix *mny, *mns;
        double *Solution, *iSolution, *Currents, *iCurrents, *Voltajes, *iVoltajes;
        double *Srcurrents, *iSrcurrents, *SrcVolts, *iSrcVolts, *Temp;
        double Omega;

        int i, j, l;
        int Size, n, k, s, Tipo;
        double freq, timestep, endtime;
        int portnmb[64]; //={0,3,5,1,0,0,0};

        /* INFORMACION QUE DEBE PROVEERSE ANTES DE COMENZAR */

        int Row;
        FILE  *pMatrixFile;
        char  sInput[BUFSIZ], name[BUFSIZ];
        int port;
        static char *FileName="analisis.opt";

                pMatrixFile = fopen(FileName, "r");
        if (pMatrixFile == NULL)
        {   fprintf(stderr, "file %s was not found.\n",
                        FileName);
            return 1;
```

```
        }

        /* Read cktname  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%s", &name);

        /* Read n  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d%s", &n,&name);

        /* Read s */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d%s", &s, &name);

        /* Read k */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d%s", &k, &name);

        /* Read freq  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%lf%s", &freq, &name);

        /* Read timestep  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%lf%s", &timestep,&name);

        /* Read endtime */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%lf%s", &endtime, &name);

        /* Read Tipo */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d%s", &Tipo, &name);

        /* Read size of array of Ports  */
        fgets( sInput, BUFSIZ, pMatrixFile );
        sscanf( sInput,"%d%s", &Row, &name);


        for (i = 0;i<Row;i++)
        {
                fgets( sInput, BUFSIZ, pMatrixFile );
                sscanf( sInput,"%d", &port );

                portnmb[i]= port;
    }

        fclose(pMatrixFile);

        //n = 3; s = 1; k = 20;

        // n = numero de puertos, s = numero de fuentes, k = numero de armonicos

    Voltajes=NEWN(double, (n-s) * k);
        iVoltajes=NEWN(double, (n-s) * k);

        SrcVolts=NEWN(double, (n-s) * k);
        iSrcVolts=NEWN(double, (n-s) * k);

        CKTmatriz=NEWN(char*, k);
        balmny=NEWN(char*, k);
```

```
    balmns=NEWN(char*, k);

for (i=0;i<=(n-s);i++)
    {
            *(Voltajes+i)=1;
    *(iVoltajes+i)=1;
    }

    for (i=0;i<=(n-s);i++)
    {
            *(SrcVolts+i)=0;
    *(iSrcVolts+i)=0;
    }

SrcVolts[1] = 1;

    /* FIN DE LA INFORMACION QUE DEBE PROVEERSE ANTES DE COMENZAR */

    spSetComplex(ckt->CKTmatrix);

Omega = ckt->CKTomega;

    //Size = 8; SMPmatSize(ckt->CKTmatrix); ??????
    Size = SMPmatSize(ckt->CKTmatrix);

    Solution=NEWN(double, Size);
    iSolution=NEWN(double, Size);

    Currents=NEWN(double, (n-s) * k);
    iCurrents=NEWN(double, (n-s) * k);

    Srcurrents=NEWN(double, (n-s) * k);
    iSrcurrents=NEWN(double, (n-s) * k);

    Temp=NEWN(double, (n-s) * k);

    for (i = 1; i <=  (n-s) * k ; i++)
    {
            Currents[i]=0;
            iCurrents[i]=0;
            iSrcurrents[i]=0;
            Srcurrents[i]=0;
    }

    SMPnewMatrix(&balmnz);
    SMPnewMatrix(&balmna);

    SMPnewMatrix(&mny);
    SMPnewMatrix(&mns);

    for (l=1;l<k+1;l++)
    {

    SMPnewMatrix(&CKTmatriz[l]);
    SMPnewMatrix(&balmny[l]);
    SMPnewMatrix(&balmns[l]);
SMPcClear(balmnz);
    SMPcClear(balmna);

if (l!=1)
            ckt->CKTomega = Omega * (l-1);
    else
```

```
                ckt->CKTomega = 2 * M_PI * 1E-3;

            error = CKTacLoad(ckt);
        if(error) return(error);

            for (j=0;j<Size;j++)
            {
                    *(Solution+j)=0;
            *(iSolution+j)=0;
            }

            SMPcopy(ckt->CKTmatrix, CKTmatriz[l], 1, 1, 1, 1, Size, Size);

            for (j=0;j<s;j++)
            {
            spDeleteRowAndCol( CKTmatriz[l], Size-j, Size-j );
            }

            //if (Size-s>n)
            //{

        error = SMPcLUfac(CKTmatriz[l], 0);

        for (i = 1; i <=  n; i++)
            {
                    Solution[portnmb[i]] = 1;
                    if (portnmb[n+i]!=0)
                       Solution[portnmb[n+i]] = -1;

            SMPcSolve(CKTmatriz[l], Solution,
                iSolution, Temp, Temp);

            for (j = 1; j <=  n; j++)
            {
                            SMPaddcElt(balmnz, i, j, Solution[portnmb[j]], iSolution[portnmb
[j]]);
                    }

            for (j=0;j<Size;j++)
                    {
                            *(Solution+j)=0;
                *(iSolution+j)=0;
                    }

            }

            error = SMPcLUfac(balmnz,0);
            //}

            for (i = 1; i <=  n; i++)
            {
                    Solution[i] = 1;

        //    if (Size-s>n)
                            SMPcSolve(balmnz, Solution,
                iSolution, Temp,
                Temp);

            for (j = 1; j <=  n; j++)
            {
            //              if (Size-s>n)
                               SMPaddcElt(balmna, i, j, Solution[j], iSolution[j]);
```

```
        //              else
    //           SMPcopy(CKTmatrix[l], balmna, portnmb[i], portnmb[j], i, j, 1 );
                }

        for (j=0;j<Size;j++)
                {
                        *(Solution+j)=0;
            *(iSolution+j)=0;
                }

        }

        SMPcopy(balmna, balmny[l], 1, 1, 1, 1, n-s, n-s);
        SMPcopy(balmna, balmns[l], 1, n-s+1, 1, 1, n-s, s);

//      SMPcopy(balmny[l], mny, 1 , 1, (n-s)*(l-1)+1, (n-s)*(l-1)+1, n*k);
//      SMPcopy(balmns[l], mns, 1 , 1, (n-s)*(l-1)+1, 1, n*k);

        SMP3Dto2D(balmny[l], mny, l, k);
        SMP3Dto2D(balmns[l], mns, l, k);

        spMultiply( balmny[l], &(Currents[(l-1)*(n-s)]), Voltajes, &(iCurrents[(l-1)*(n-
s)]), iVoltajes);
        spMultiply( balmns[l], &(Srcurrents[(l-1)*(n-s)]), SrcVolts, &(iSrcurrents[(l-1)*(n-
s)]), iSrcVolts);

/*
        for (i = 1; i <  n ; i++)
        {
                Currents[i]+=Srcurrents[i];
                iCurrents[i]+=iSrcurrents[i];
        }
*/
}

spFileMatrix(mns, "src.mat", "src matrix", 0, -1, -1);
spFileMatrix(mny, "mna.mat", "mna matrix", 0, -1, -1);

#ifdef _DEBUG /* this lines were added by me */
/*
                for (i = 1; i <= (n-s) * k; i++)
                {
                        printf("%e %e \n",Currents[i], iCurrents[i]);
                }
                printf(" \n");

                for (i = 1; i <= (n-s) * k; i++)
                {
                        printf("%e %e \n",Srcurrents[i], iSrcurrents[i]);
                }
                printf(" \n");
*/
#endif

ckt->CKTomega = Omega;

#endif

retry:
    ckt->CKTnoncon=0;

    error = CKTacLoad(ckt);
```

```
    if(error) return(error);

#define _DEBUG

#ifdef _DEBUG /* this lines were added by me */

        Size = SMPmatSize(ckt->CKTmatrix);

        spSetComplex((char *) ckt->CKTmatrix );

    SMPprint(ckt->CKTmatrix, stdout);
/*
        for (i = 1; i <=  Size; i++)
        {
                printf("%e %e \n",ckt->CKTrhs[i], ckt->CKTirhs[i]);
        }
        printf(" \n");
*/
#endif

exit(0);

    if(ckt->CKTniState & NIACSHOULDREORDER) {
        error = SMPcReorder(ckt->CKTmatrix,ckt->CKTpivotAbsTol,
                ckt->CKTpivotRelTol,&ignore);
        ckt->CKTniState &= ~NIACSHOULDREORDER;
        if(error != 0) {
            /* either singular equations or no memory, in either case,
             * let caller handle problem
             */
            return(error);
        }
    } else {
        error = SMPcLUfac(ckt->CKTmatrix,ckt->CKTpivotAbsTol);
        if(error != 0) {
            if(error == E_SINGULAR) {
                /* the problem is that the matrix can't be solved with the
                 * current LU factorization.  Maybe if we reload and
                 * try to reorder again it will help...
                 */
                ckt->CKTniState |= NIACSHOULDREORDER;
                goto retry;
            }
            return(error); /* can't handle E_BADMATRIX, so let caller */
        }
    }
    SMPcSolve(ckt->CKTmatrix,ckt->CKTrhs,
            ckt->CKTirhs, ckt->CKTrhsSpare,
            ckt->CKTirhsSpare);

#ifdef _DEBUG /* this lines were added by me */

        for (i = 1; i <=  Size; i++)
        {
                printf("%e %e \n",ckt->CKTrhs[i], ckt->CKTirhs[i]);
        }
        printf(" \n");

#endif

B-288   *ckt->CKTrhs = 0;
        *ckt->CKTrhsSpare = 0;
```

```
        *ckt->CKTrhsOld = 0;
        *ckt->CKTirhs = 0;
        *ckt->CKTirhsSpare = 0;
        *ckt->CKTirhsOld = 0;

        temp = ckt->CKTirhsOld;
        ckt->CKTirhsOld = ckt->CKTirhs;
        ckt->CKTirhs = temp;

        temp = ckt->CKTrhsOld;
        ckt->CKTrhsOld = ckt->CKTrhs;
        ckt->CKTrhs = temp;
        return(OK);
    }
```

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Wayne A. Christopher, U. C. Berkeley CAD Group
**********/

/*
 * Various post-processor commands having to do with vectors.
 */

#include "spice.h"
#include "util.h"
#include "misc.h"
#include "cpdefs.h"
#include "ftedefs.h"
#include "fteparse.h"
#include "ftedata.h"
#include "suffix.h"

static int dcomp();
static void pvec();
static void killplot( );

void
com_let(wl)
    wordlist *wl;
{
    char *p, *q, *s;
    int indices[MAXDIMS];
    int numdims;
    wordlist fake_wl;
    int need_open;
    int offset, length;
    struct pnode *nn;
    struct dvec *n, *t;
    int i, cube;
    int j, depth;
    int newvec;
    char *rhs;

    fake_wl.wl_next = NULL;

    if (!wl) {
        com_display((wordlist *) NULL);
        return;
    }

    p = wl_flatten(wl);

    /* extract indices */
    numdims = 0;
    if (rhs = index(p, '=')) {
        *rhs++ = 0;
    } else {
```

```
        fprintf(cp_err, "Error: bad let syntax\n");
        return;
    }
    if (s = index(p, '[')) {
        need_open = 0;
        *s++ = 0;
        while (!need_open || *s == '[') {
            depth = 0;
            if (need_open)
                s++;
            for (q = s; *q && (*q != ']' && *q != ',' || depth > 0); q++) {
                switch (*q) {
                case '[':
                    depth += 1;
                    break;
                case ']':
                    depth -= 1;
                    break;
                }
            }

            if (depth != 0 || !*q) {
                printf("syntax error specifying index\n");
                return;
            }

            if (*q == ']')
                need_open = 1;
            else
                need_open = 0;
            if (*q)
                *q++ = 0;

            /* evaluate expression between s and q */
            fake_wl.wl_word = s;
            nn = ft_getpnames(&fake_wl, true);
            t = ft_evaluate(nn);

            free_pnode(nn);

            if (!isreal(t) || t->v_link2 || t->v_length != 1 || !t->v_realdata)
            {
                fprintf(cp_err, "Error: index is not a scalar.\n");
                tfree(p);
                return;
            }
            j = t->v_realdata[0]; /* ignore sanity checks for now */

            if (j < 0) {
                printf("negative index (%d) is not allowed\n", j);
                tfree(p);
                /* XXX free data */
                return;
            }
```

```
            indices[numdims++] = j;

            for (s = q; *s && isspace(*s); s++)
                ;
        }
    }
    /* vector name at p */

    for (q = p + strlen(p) - 1; *q <= ' ' && p <= q; q--)
        ;

    *++q = 0;

    /* sanity check */
    if (eq(p, "all") || index(p, '@')) {
        fprintf(cp_err, "Error: bad variable name %s\n", p);
        return;
    }

    /* evaluate rhs */
    fake_wl.wl_word = rhs;
    nn = ft_getpnames(&fake_wl, true);
    if (nn == NULL) {
        /* XXX error message */
        tfree(p);
        return;
    }
    t = ft_evaluate(nn);
    if (!t) {
        /* XXX error message */
        tfree(p);
        return;
    }

    if (t->v_link2)
        fprintf(cp_err, "Warning: extra wildcard values ignored\n");

    n = vec_get(p);

    if (n) {
        /* re-allocate? */
        /* vec_free(n); */
        newvec = 0;
    } else {
        if (numdims) {
            fprintf(cp_err, "Can't assign into a subindex of a new vector\n");
            tfree(p);
            return;
        }

        /* create and assign a new vector */
        n = alloc(struct dvec);
```

```
    ZERO(n, struct dvec);
    n->v_name = copy(p);
    n->v_type = t->v_type;
    n->v_flags = (t->v_flags | VF_PERMANENT);
    n->v_length = t->v_length;

    if (!t->v_numdims) {
        n->v_numdims = 1;
        n->v_dims[0] = n->v_length;
    } else {
        n->v_numdims = t->v_numdims;
        for (i = 0; i < t->v_numdims; i++)
            n->v_dims[i] = t->v_dims[i];
    }

    if (isreal(t))
        n->v_realdata = (double *) tmalloc(n->v_length * sizeof(double));
    else
        n->v_compdata = (complex *) tmalloc(n->v_length * sizeof(complex));
    newvec = 1;
    vec_new(n);
}

/* fix-up dimensions */
if (n->v_numdims < 1) {
    n->v_numdims = 1;
    n->v_dims[0] = n->v_length;
}

/* Compare dimensions */
offset = 0;
length = n->v_length;

cube = 1;
for (i = n->v_numdims - 1; i >= numdims; i--)
    cube *= n->v_dims[i];

for (i = numdims - 1; i >= 0; i--) {
    offset += cube * indices[i];
    if (i < n->v_numdims) {
        cube *= n->v_dims[i];
        length /= n->v_dims[i];
    }
}

/* length is the size of the unit refered to */
/* cube ends up being the length */

if (length > t->v_length) {
    fprintf(cp_err, "left-hand expression is too small (need %d)\n",
            length * cube);
    if (newvec)
        n->v_flags &= ~VF_PERMANENT;
    tfree(p);
```

```
            return;
        }
        if (isreal(t) != isreal(n)) {
            fprintf(cp_err,
                    "Types of vectors are not the same (real vs. complex)\n");
            if (newvec)
                n->v_flags &= ~VF_PERMANENT;
            tfree(p);
            return;
        } else if (isreal(t)) {
            bcopy((char *) t->v_realdata, (char *) (n->v_realdata + offset),
                    length * sizeof (double));
        } else {
            bcopy((char *) t->v_compdata, (char *) (n->v_compdata + offset),
                    length * sizeof (complex));
        }

        n->v_minsignal = 0.0; /* How do these get reset ??? */
        n->v_maxsignal = 0.0;

        n->v_scale = t->v_scale;

        if (newvec)
            cp_addkword(CT_VECTOR, n->v_name);

        /* XXXX Free t !?! */
        tfree(p);
        return;
    }

    /* Undefine vectors. */

    void
    com_unlet(wl)
        wordlist *wl;
    {
        while (wl) {
            vec_remove(wl->wl_word);
            wl = wl->wl_next;
        }
        return;
    }

    /* Load in a file. */

    void
    com_load(wl)
        wordlist *wl;
    {

        if (!wl)
            ft_loadfile(ft_rawfile);
        else
            while (wl) {
```

```
            ft_loadfile(cp_unquote(wl->wl_word));
            wl = wl->wl_next;
        }

    /* note: default is to display the vectors in the last (current) plot */
    com_display(NULL);

    return;
}

/* Print out the value of an expression. When we are figuring out what to
 * print, link the vectors we want with v_link2... This has to be done
 * because of the way temporary vectors are linked together with permanent
 * ones under the plot.
 */

void
com_print(wl)
    wordlist *wl;
{
    struct dvec *v, *lv, *bv, *nv, *vecs = NULL;
    int i, j, ll, width = DEF_WIDTH, height = DEF_HEIGHT, npoints, lineno;
    struct pnode *nn;
    struct plot *p;
    bool col = true, nobreak = false, noprintscale, plotnames = false;
    bool optgiven = false;
    char *s, buf[BSIZE_SP], buf2[BSIZE_SP];
    int ngood;

        //modificado aqui
        #define BALANCE_ARMONICO

        #ifdef BALANCE_ARMONICO /* this lines were added by me */

        int newlen;
    struct dvec *newv_scale;
    double *newdata, *newscale;
    double tstep, tstart, tstop, ttime;
        FILE  *pMatrixFile;
        static char *File="vectors.mat";

        #endif
        //modificado aqui

    if (wl == NULL)
        return;
    if (eq(wl->wl_word, "col")) {
        col = true;
        optgiven = true;
        wl = wl->wl_next;
    } else if (eq(wl->wl_word, "line")) {
        col = false;
        optgiven = true;
```

```
            wl = wl->wl_next;
        }

        ngood = 0;
        for (nn = ft_getpnames(wl, true); nn; nn = nn->pn_next) {
            if (!(v = ft_evaluate(nn)))
                continue;
            if (!vecs)
                vecs = lv = v;
            else
                lv->v_link2 = v;
            for (lv = v; lv->v_link2; lv = lv->v_link2)
                ;
            ngood += 1;
        }

        if (!ngood)
            return;

        /* See whether we really have to print plot names. */
        for (v = vecs; v; v = v->v_link2)
            if (vecs->v_plot != v->v_plot) {
                plotnames = true;
                break;
            }

        if (!optgiven) {
            /* Figure out whether col or line should be used... */
            col = false;
            for (v = vecs; v; v = v->v_link2)
                if (v->v_length > 1) {
                    col = true;
                    break;
                }
        }

        out_init();

    //* modificado aqui
            #ifdef BALANCE_ARMONICO /* this lines were added by me */

            if (!(!ft_curckt || !ft_curckt->ci_ckt ||
                strcmp(ft_curckt->ci_name, plot_cur->pl_title) ||
                !if_tranparams(ft_curckt, &tstart, &tstop, &tstep) ||
                ((tstop - tstart) * tstep <= 0.0) ||
                ((tstop - tstart) < tstep) ||
                !plot_cur || !plot_cur->pl_dvecs ||
                !plot_cur->pl_scale ||
                !isreal(plot_cur->pl_scale) ||
                !ciprefix("tran", plot_cur->pl_typename)))
            {

              newlen = (tstop - tstart) / tstep + 1.5;
```

```
        newscale = (double *) tmalloc(newlen * sizeof(double));

        newv_scale = alloc(struct dvec);
        newv_scale->v_flags = vecs->v_plot->pl_scale->v_flags;
        newv_scale->v_type = vecs->v_plot->pl_scale->v_type;
        newv_scale->v_gridtype = vecs->v_plot->pl_scale->v_gridtype;
        newv_scale->v_length = newlen;
        newv_scale->v_name = copy(vecs->v_plot->pl_scale->v_name);
        newv_scale->v_realdata = newscale;

        for (i = 0, ttime = tstart; i < newlen; i++, ttime += tstep)
          newscale[i] = ttime;

        for (v = vecs; v; v= v->v_link2) {
          newdata = (double *) tmalloc(newlen * sizeof (double));

          if (!ft_interpolate(v->v_realdata, newdata,
              v->v_plot->pl_scale->v_realdata, v->v_plot->pl_scale->v_length,
          newscale, newlen, 1)) {
          fprintf(cp_err,
              "Error: can't interpolate %s\n", v->v_name);
          return(false);
          }

          tfree(v->v_realdata);
          v->v_realdata = newdata;
                      v->v_length = newlen;

          // Why go to all this trouble if agraf ignores it?
        }

        vecs->v_plot->pl_scale = newv_scale;
                vecs->v_length = newlen;

    }

    /* Open File in append mode. */
if ((pMatrixFile = fopen(File,"w")) == NULL)
    return 0;

for (v = vecs; v; v= v->v_link2)
{
    fprintf(pMatrixFile, "%s.%s\n", v->v_plot->pl_typename,
                    vec_basename(v));
fprintf(pMatrixFile, "%i\n", newlen);

            for (j = 0; j < newlen; j++)
    if (isreal(v)) {
    fprintf(pMatrixFile, "%i\t%.15e\n",
    j, v->v_realdata[j]);
} else {
    fprintf(pMatrixFile, "%i\t%.15e,\t%.15e\n",
            j, realpart(&v->v_compdata[j]),
    imagpart(&v->v_compdata[j]));
```

```
}
    }

    /* Close file. */
if (fclose(pMatrixFile) < 0) return 0;
return 1;

    #endif
    // modificado aqui */

if (!col) {
    for (v = vecs; v; v = v->v_link2) {
        if (plotnames) {
            (void) sprintf(buf, "%s.%s", v->v_plot->pl_typename,
                    vec_basename(v));
        } else {
            (void) strcpy(buf, vec_basename(v));
        }
        for (s = buf; *s; s++)
            ;
        s--;
        while (isspace(*s)) {
            *s = '\0';
            s--;
        }
        ll = 10;
        if (v->v_length == 1) {
            if (isreal(v)) {
                out_printf("%s = %s\n", buf,
                    printnum(*v->v_realdata));
            } else {
                out_printf("%s = %s,%s\n", buf,
                    copy(printnum(realpart(v->v_compdata))),
                    copy(printnum(imagpart(v->v_compdata))));
            }
        } else {
            out_printf("%s = (  ", buf);
            for (i = 0; i < v->v_length; i++)
                if (isreal(v)) {
                    (void) strcpy(buf,
                       printnum(v->v_realdata[i]));
                    out_send(buf);
                    ll += strlen(buf);
                    ll = (ll + 7) / 8;
                    ll = ll * 8 + 1;
                    if (ll > 60) {
                        out_send("\n\t");
                        ll = 9;
                    } else
                        out_send("\t");
                } else {
                    (void) sprintf(buf, "%s,%s",
                        copy(printnum(realpart(&v->v_compdata[i]))),
                        copy(printnum(imagpart(&v->v_compdata[i]))));
```

```
                        out_send(buf);
                        ll += strlen(buf);
                        ll = (ll + 7) / 8;
                        ll = ll * 8 + 1;
                        if (ll > 60) {
                            out_send("\n\t");
                            ll = 9;
                        } else
                            out_send("\t");
                    }
                out_send(")\n");
            }
        }
    } else {    /* Print in columns. */
        if (cp_getvar("width", VT_NUM, (char *) &i))
            width = i;
        if (width < 40)
            width = 40;
        if (cp_getvar("height", VT_NUM, (char *) &i))
            height = i;
        if (height < 20)
            height = 20;
        if (!cp_getvar("nobreak", VT_BOOL, (char *) &nobreak) && !ft_nopage)
            nobreak = false;
        else
            nobreak = true;
        (void) cp_getvar("noprintscale", VT_BOOL, (char *)
                &noprintscale);
        bv = vecs;
nextpage:
        /* Make the first vector of every page be the scale... */
        /* XXX But what if there is no scale?  e.g. op, pz */
        if (!noprintscale && bv->v_plot->pl_ndims) {
            if (bv->v_plot->pl_scale && !vec_eq(bv, bv->v_plot->pl_scale)) {
                nv = vec_copy(bv->v_plot->pl_scale);
                vec_new(nv);
                nv->v_link2 = bv;
                bv = nv;
            }
        }
        ll = 8;
        for (lv = bv; lv; lv = lv->v_link2) {
            if (isreal(lv))
                ll += 16;   /* Two tabs for real, */
            else
                ll += 32;   /* 4 for complex. */
            /* Make sure we have at least 2 vectors per page... */
            if ((ll > width) && (lv != bv) && (lv != bv->v_link2))
                break;
        }

        /* Print the header on the first page only. */
        p = bv->v_plot;
```

```
        j = (width - (int) strlen(p->pl_title)) / 2;     /* Yes, keep "(int)" */
        if (j < 0)
                j = 0;
        for (i = 0; i < j; i++)
            buf2[i] = ' ';
        buf2[j] = '\0';
        out_send(buf2);
        out_send(p->pl_title);
        out_send("\n");
        out_send(buf2);
        (void) sprintf(buf, "%s  %s", p->pl_name, p->pl_date);
        j = (width - strlen(buf)) / 2;
        out_send(buf);
        out_send("\n");
        for (i = 0; i < width; i++)
            buf2[i] = '-';
        buf2[width] = '\n';
        buf2[width+1] = '\0';
        out_send(buf2);
        (void) sprintf(buf, "Index   ");
        for (v = bv; v && (v != lv); v = v->v_link2) {
            if (isreal(v))
                (void) sprintf(buf2, "%-16.15s", v->v_name);
            else
                (void) sprintf(buf2, "%-32.31s", v->v_name);
            (void) strcat(buf, buf2);
        }
        lineno = 3;
        j = 0;
        npoints = 0;
        for (v = bv; (v && (v != lv)); v = v->v_link2)
            if (v->v_length > npoints)
                npoints = v->v_length;
pbreak:     /* New page. */
        out_send(buf);
        out_send("\n");
        for (i = 0; i < width; i++)
            buf2[i] = '-';
        buf2[width] = '\n';
        buf2[width+1] = '\0';
        out_send(buf2);
        lineno += 2;
loop:
        while ((j < npoints) && (lineno < height)) {
/*              out_printf("%d\t", j); */
            sprintf(out_pbuf, "%d\t", j);
            out_send(out_pbuf);
            for (v = bv; (v && (v != lv)); v = v->v_link2) {
                if (v->v_length <= j) {
                    if (isreal(v))
                        out_send("\t\t");
                    else
                        out_send("\t\t\t\t");
                } else {
```

```
                    if (isreal(v)) {
                        sprintf(out_pbuf, "%e\t",
                                v->v_realdata[j]);
                        out_send(out_pbuf);
                    } else {
                        sprintf(out_pbuf, "%e,\t%e\t",
                                realpart(&v->v_compdata[j]),
                                imagpart(&v->v_compdata[j]));
                        out_send(out_pbuf);
                    }
                }
            }
            out_send("\n");
            j++;
            lineno++;
        }
        if ((j == npoints) && (lv == NULL)) /* No more to print. */
            goto done;
        if (j == npoints) { /* More vectors to print. */
            bv = lv;
            out_send("\f\n");    /* Form feed. */
            goto nextpage;
        }

        /* Otherwise go to a new page. */
        lineno = 0;
        if (nobreak)
            goto loop;
        else
            out_send("\f\n");    /* Form feed. */
        goto pbreak;
    }
done:
    /* Get rid of the vectors. */
    return;
}

/* Write out some data. write filename expr ... Some cleverness here is
 * required.  If the user mentions a few vectors from various plots,
 * probably he means for them to be written out seperate plots.  In any
 * case, we have to be sure to write out the scales for everything we
 * write...
 */

void
com_write(wl)
    wordlist *wl;
{
    char *file, buf[BSIZE_SP];
    struct pnode *n;
    struct dvec *d, *vecs = NULL, *lv = NULL, *end, *vv;
    static wordlist all = { "all", NULL, NULL } ;
    struct pnode *names;
```

```
    bool ascii = AsciiRawFile;
    bool scalefound, appendwrite;
    struct plot *tpl, newplot;

    if (wl) {
        file = wl->wl_word;
        wl = wl->wl_next;
    } else
        file = ft_rawfile;
    if (cp_getvar("filetype", VT_STRING, buf)) {
        if (eq(buf, "binary"))
            ascii = false;
        else if (eq(buf, "ascii"))
            ascii = true;
        else
            fprintf(cp_err, "Warning: strange file type %s\n", buf);
    }
    (void) cp_getvar("appendwrite", VT_BOOL, (char *) &appendwrite);

    if (wl)
        names = ft_getpnames(wl, true);
    else
        names = ft_getpnames(&all, true);
    if (names == NULL)
        return;
    for (n = names; n; n = n->pn_next) {
        d = ft_evaluate(n);
        if (!d)
            return;
        if (vecs)
            lv->v_link2 = d;
        else
            vecs = d;
        for (lv = d; lv->v_link2; lv = lv->v_link2)
            ;
    }

    /* Now we have to write them out plot by plot. */

    while (vecs) {
        tpl = vecs->v_plot;
        tpl->pl_written = true;
        end = NULL;
        bcopy((char *) tpl, (char *) &newplot, sizeof (struct plot));
        scalefound = false;

        /* Figure out how many vectors are in this plot. Also look
         * for the scale, or a copy of it, which may have a different
         * name.
         */
        for (d = vecs; d; d = d->v_link2) {
            if (d->v_plot == tpl) {
                vv = vec_copy(d);
```

```
            /* Note that since we are building a new plot
             * we don't want to vec_new this one...
             */
            vv->v_name = vec_basename(vv);

            if (end)
                end->v_next = vv;
            else
                end = newplot.pl_dvecs = vv;
            end = vv;

            if (vec_eq(d, tpl->pl_scale)) {
                newplot.pl_scale = vv;
                scalefound = true;
            }
        }
    }
    end->v_next = NULL;

    /* Maybe we shouldn't make sure that the default scale is
     * present if nobody uses it.
     */
    if (!scalefound) {
        newplot.pl_scale = vec_copy(tpl->pl_scale);
        newplot.pl_scale->v_next = newplot.pl_dvecs;
        newplot.pl_dvecs = newplot.pl_scale;
    }

    /* Now let's go through and make sure that everything that
     * has its own scale has it in the plot.
     */
    for (;;) {
        scalefound = false;
        for (d = newplot.pl_dvecs; d; d = d->v_next) {
            if (d->v_scale) {
                for (vv = newplot.pl_dvecs; vv; vv =
                        vv->v_next)
                    if (vec_eq(vv, d->v_scale))
                        break;
                /* We have to grab it... */
                vv = vec_copy(d->v_scale);
                vv->v_next = newplot.pl_dvecs;
                newplot.pl_dvecs = vv;
                scalefound = true;
            }
        }
        if (!scalefound)
            break;
        /* Otherwise loop through again... */
    }

    if (ascii)
        raw_write(file, &newplot, appendwrite, false);
```

```
            else
                raw_write(file, &newplot, appendwrite, true);

            /* Now throw out the vectors we have written already... */
            for (d = vecs, lv = NULL;  d; d = d->v_link2)
                if (d->v_plot == tpl) {
                    if (lv) {
                        lv->v_link2 = d->v_link2;
                        d = lv;
                    } else
                        vecs = d->v_link2;
                } else
                    lv = d;
            /* If there are more plots we want them appended... */
            appendwrite = true;
        }
        return;
    }

    /* If the named vectors have more than 1 dimension, then consider
     * to be a collection of one or more matrices.  This command transposes
     * each named matrix.
     */
    void
    com_transpose(wl)
        wordlist *wl;
    {
        struct dvec *d;
        char *s;

        while (wl) {
            s = cp_unquote(wl->wl_word);
            d = vec_get(s);
            if (d == NULL)
                fprintf(cp_err, "Error: no such vector as %s.\n",
                    wl->wl_word);
            else
                while (d) {
                    vec_transpose(d);
                    d = d->v_link2;
                }
            if (wl->wl_next == NULL)
                return;
            wl = wl->wl_next;
        }
    }

    /* Set the default scale to the named vector.  If no vector named,
     * find and print the default scale.
     */

    void
    com_setscale(wl)
        wordlist *wl;
```

```
{
    struct dvec *d;
    char *s;

    if (plot_cur) {
        if (wl) {
            s = cp_unquote(wl->wl_word);
            d = vec_get(s);
            if (d == NULL)
                fprintf(cp_err, "Error: no such vector as %s.\n",
                    wl->wl_word);
            else
                plot_cur->pl_scale = d;
        } else if (plot_cur->pl_scale) {
            pvec(plot_cur->pl_scale);
        }
    } else {
        fprintf(cp_err, "Error: no current plot.\n");
    }
}


/* Display vector status, etc.  Note that this only displays stuff from the
 * current plot, and you must do a setplot to see the rest of it.
 */

void
com_display(wl)
    wordlist *wl;
{
    struct dvec *d;
    struct dvec **dvs;
    int len = 0, i = 0;
    bool b;
    char *s;

    /* Maybe he wants to know about just a few vectors. */

    out_init();
    while (wl) {
        s = cp_unquote(wl->wl_word);
        d = vec_get(s);
        if (d == NULL)
            fprintf(cp_err, "Error: no such vector as %s.\n",
                wl->wl_word);
        else
            while (d) {
                pvec(d);
                d = d->v_link2;
            }
        if (wl->wl_next == NULL)
            return;
        wl = wl->wl_next;
    }
```

```
    if (plot_cur)
        for (d = plot_cur->pl_dvecs; d; d = d->v_next)
            len++;
    if (len == 0) {
        fprintf(cp_out, "There are no vectors currently active.\n");
        return;
    }
    out_printf("Here are the vectors currently active:\n\n");
    dvs = (struct dvec **) tmalloc(len * (sizeof (struct dvec *)));
    for (d = plot_cur->pl_dvecs, i = 0; d; d = d->v_next, i++)
        dvs[i] = d;
    if (!cp_getvar("nosort", VT_BOOL, (char *) &b))
        qsort((char *) dvs, len, sizeof (struct dvec *), dcomp);

    out_printf("Title: %s\n",  plot_cur->pl_title);
    out_printf("Name: %s (%s)\nDate: %s\n\n",
        plot_cur->pl_typename, plot_cur->pl_name,
        plot_cur->pl_date);
    for (i = 0; i < len; i++) {
        d = dvs[i];
        pvec(d);
    }
    return;
}

static void
pvec(d)
    struct dvec *d;
{
    int i;
    char buf[BSIZE_SP], buf2[BSIZE_SP];

    sprintf(buf, "    %-20s: %s, %s, %d long", d->v_name,
            ft_typenames(d->v_type), isreal(d) ? "real" :
            "complex", d->v_length);
    if (d->v_flags & VF_MINGIVEN) {
        sprintf(buf2, ", min = %lg", d->v_minsignal);
        strcat(buf, buf2);
    }
    if (d->v_flags & VF_MAXGIVEN) {
        sprintf(buf2, ", max = %lg", d->v_maxsignal);
        strcat(buf, buf2);
    }
    switch (d->v_gridtype) {

        case GRID_LOGLOG:
        strcat(buf, ", grid = loglog");
        break;

        case GRID_XLOG:
        strcat(buf, ", grid = xlog");
        break;

        case GRID_YLOG:
```

```
            strcat(buf, ", grid = ylog");
            break;

            case GRID_POLAR:
            strcat(buf, ", grid = polar");
            break;

            case GRID_SMITH:
            strcat(buf, ", grid = smith (xformed)");
            break;

            case GRID_SMITHGRID:
            strcat(buf, ", grid = smithgrid (not xformed)");
            break;
        }
        switch (d->v_plottype) {

            case PLOT_COMB:
            strcat(buf, ", plot = comb");
            break;

            case PLOT_POINT:
            strcat(buf, ", plot = point");
            break;

        }
        if (d->v_defcolor) {
            sprintf(buf2, ", color = %s", d->v_defcolor);
            strcat(buf, buf2);
        }
        if (d->v_scale) {
            sprintf(buf2, ", scale = %s", d->v_scale->v_name);
            strcat(buf, buf2);
        }
        if (d->v_numdims > 1) {
            sprintf(buf2, ", dims = [%s]", dimstring(d->v_dims, d->v_numdims));
            strcat(buf, buf2);
        }
        if (d->v_plot->pl_scale == d) {
            strcat(buf, " [default scale]\n");
        } else {
            strcat(buf, "\n");
        }
        out_send(buf);
        return;
    }

    #ifdef notdef

    /* Set the current working plot. */

    void
    com_splot(wl)
        wordlist *wl;
```

```
{
    struct plot *p;
    char buf[BSIZE_SP], *s;

    if (wl == NULL) {
        fprintf(cp_out, "\tType the name of the desired plot:\n\n");
        fprintf(cp_out, "\tnew\tNew plot\n");
        for (p = plot_list; p; p = p->pl_next) {
            if (plot_cur == p)
                fprintf(cp_out, "Current");
            fprintf(cp_out, "\t%s\t%s (%s)\n",
                p->pl_typename, p->pl_title, p->pl_name);
        }
        fprintf(cp_out, "? ");
        (void) fflush(cp_out);
        (void) fgets(buf, BSIZE_SP, cp_in);
        clearerr(cp_in);
        for (s = buf; *s && !isspace(*s); s++)
            ;
        *s = '\0';
    } else {
        (void) strcpy(buf, wl->wl_word);
    }
    if (prefix("new", buf)) {
        p = plot_alloc("unknown");
        p->pl_title = copy("Anonymous");
        p->pl_name = copy("unknown");
        p->pl_next = plot_list;
        plot_list = p;
    } else {
        for (p = plot_list; p; p = p->pl_next)
            if (plot_prefix(buf, p->pl_typename))
                break;
        if (!p) {
            fprintf(cp_err, "Error: no such plot.\n");
            return;
        }
    }
    plot_cur->pl_ccom = cp_kwswitch(CT_VECTOR, p->pl_ccom);
    plot_cur = p;
    plot_docoms(plot_cur->pl_commands);
    if (wl)
        fprintf(cp_out, "%s %s (%s)\n", p->pl_typename, p->pl_title,
                p->pl_name);
    return;
}

#endif

/* For the sort in display. */

static int
dcomp(v1, v2)
    struct dvec **v1, **v2;
```

```
{
    return (strcmp((*v1)->v_name, (*v2)->v_name));
}

#ifdef notdef

/* Figure out what the name of this vector should be (if it is a number,
 * then make it 'V' or 'I')... Note that the data is static.
 */

static char *
dname(d)
    struct dvec *d;
{
    static char buf[128];
    char *s;

    for (s = d->v_name; *s; s++)
        if (!isdigit(*s))
            return (d->v_name);
    switch (d->v_type) {
        case SV_VOLTAGE:
            (void) sprintf(buf, "V(%s)", d->v_name);
            return (buf);
        case SV_CURRENT:
            (void) sprintf(buf, "I(%s)", d->v_name);
            return (buf);
    }
    return (d->v_name);
}

#endif

/* Take a set of vectors and form a new vector of the nth elements of each. */

void
com_cross(wl)
    wordlist *wl;
{
    char *newvec, *s;
    struct dvec *n, *v, *vecs = NULL, *lv = NULL;
    struct pnode *pn;
    int i, ind;
    bool comp = false;
    double *d;

    newvec = wl->wl_word;
    wl = wl->wl_next;
    s = wl->wl_word;
    if (!(d = ft_numparse(&s, false))) {
        fprintf(cp_err, "Error: bad number %s\n", wl->wl_word);
        return;
    }
    if ((ind = *d) < 0) {
```

```
            fprintf(cp_err, "Error: bad index %d\n", ind);
            return;
        }
        wl = wl->wl_next;
        pn = ft_getpnames(wl, true);
        while (pn) {
            if (!(n = ft_evaluate(pn)))
                return;
            if (!vecs)
                vecs = lv = n;
            else
                lv->v_link2 = n;
            for (lv = n; lv->v_link2; lv = lv->v_link2)
                ;
            pn = pn->pn_next;
        }
        for (n = vecs, i = 0; n; n = n->v_link2) {
            if (iscomplex(n))
                comp = true;
            i++;
        }

        vec_remove(newvec);
        v = alloc(struct dvec);
        v->v_name = copy(newvec);
        v->v_type = vecs ? vecs->v_type : SV_NOTYPE;
        v->v_length = i;
        v->v_flags |= VF_PERMANENT;
        v->v_flags = comp ? VF_COMPLEX : VF_REAL;
        if (comp)
            v->v_compdata = (complex *) tmalloc(i * sizeof (complex));
        else
            v->v_realdata = (double *) tmalloc(i * sizeof (double));

        /* Now copy the ind'ths elements into this one. */
        for (n = vecs, i = 0; n; n = n->v_link2, i++)
            if (n->v_length > ind) {
                if (comp) {
                    realpart(&v->v_compdata[i]) =
                            realpart(&n->v_compdata[ind]);
                    imagpart(&v->v_compdata[i]) =
                            imagpart(&n->v_compdata[ind]);
                } else
                    v->v_realdata[i] = n->v_realdata[ind];
            } else {
                if (comp) {
                    realpart(&v->v_compdata[i]) = 0.0;
                    imagpart(&v->v_compdata[i]) = 0.0;
                } else
                    v->v_realdata[i] = 0.0;
            }
        vec_new(v);
        v->v_flags |= VF_PERMANENT;
        cp_addkword(CT_VECTOR, v->v_name);
```

```
    return;
}

void
com_destroy(wl)
    wordlist *wl;
{
    struct plot *pl, *npl = NULL;

    if (!wl)
        killplot(plot_cur);
    else if (eq(wl->wl_word, "all")) {
        for (pl = plot_list; pl; pl = npl) {
            npl = pl->pl_next;
            if (!eq(pl->pl_typename, "const"))
                killplot(pl);
        }
    } else {
        while (wl) {
            for (pl = plot_list; pl; pl = pl->pl_next)
                if (eq(pl->pl_typename, wl->wl_word))
                    break;
            if (pl)
                killplot(pl);
            else
                fprintf(cp_err, "Error: no such plot %s\n",
                        wl->wl_word);
            wl = wl->wl_next;
        }
    }
    return;
}

static void
killplot(pl)
    struct plot *pl;
{
    struct dvec *v, *nv = NULL;
    struct plot *op;

    if (eq(pl->pl_typename, "const")) {
        fprintf(cp_err, "Error: can't destroy the constant plot\n");
        return;
    }
    for (v = pl->pl_dvecs; v; v = nv) {
        nv = v->v_next;
        vec_free(v);
    }
    if (pl == plot_list) {
        plot_list = pl->pl_next;
        if (pl == plot_cur)
            plot_cur = plot_list;
    } else {
        for (op = plot_list; op; op = op->pl_next)
```

```
            if (op->pl_next == pl)
                break;
        if (!op)
            fprintf(cp_err,
                "Internal Error: kill plot -- not in list\n");
        op->pl_next = pl->pl_next;
        if (pl == plot_cur)
            plot_cur = op;
    }
    tfree(pl->pl_title);
    tfree(pl->pl_name);
    tfree(pl->pl_typename);
    wl_free(pl->pl_commands);

    /* Never mind about the rest... */

    return;
}

void
com_splot(wl)
    wordlist *wl;
{
    struct plot *pl;
    char buf[BSIZE_SP], *s, *t;

    if (wl) {
        plot_setcur(wl->wl_word);
        return;
    }
    fprintf(cp_out, "\tType the name of the desired plot:\n\n");
    fprintf(cp_out, "\tnew\tNew plot\n");
    for (pl = plot_list; pl; pl = pl->pl_next)
        fprintf(cp_out, "%s%s\t%s (%s)\n",
                (pl == plot_cur) ? "Current " : "\t",
                pl->pl_typename, pl->pl_title, pl->pl_name);

    fprintf(cp_out, "? ");
    if (!fgets(buf, BSIZE_SP, cp_in)) {
        clearerr(cp_in);
        return;
    }
    t = buf;
    if (!(s = gettok(&t)))
        return;

    plot_setcur(s);
    return;
}
```

```
/*
 *  Spice3 COMPATIBILITY MODULE
 *
 *  Author:                     Advising professor:
 *     Kenneth S. Kundert           Alberto Sangiovanni-Vincentelli
 *     UC Berkeley
 *
 *  This module contains routines that make Sparse1.3 a direct
 *  replacement for the SMP sparse matrix package in Spice3c1 or Spice3d1.
 *  Sparse1.3 is in general a faster and more robust package than SMP.
 *  These advantages become significant on large circuits.
 *
 *  >>> User accessible functions contained in this file:
 *  SMPaddElt
 *  SMPmakeElt
 *  SMPcClear
 *  SMPclear
 *  SMPcLUfac
 *  SMPluFac
 *  SMPcReorder
 *  SMPreorder
 *  SMPcaSolve
 *  SMPcSolve
 *  SMPsolve
 *  SMPmatSize
 *  SMPnewMatrix
 *  SMPdestroy
 *  SMPpreOrder
 *  SMPprint
 *  SMPgetError
 *  SMPcProdDiag
 *  LoadGmin
 *  SMPfindElt
 */

/*
 *  To replace SMP with Sparse, rename the file spSpice3.h to
 *  spMatrix.h and place Sparse in a subdirectory of SPICE called
 *  `sparse'.  Then on UNIX compile Sparse by executing `make spice'.
 *  If not on UNIX, after compiling Sparse and creating the sparse.a
 *  archive, compile this file (spSMP.c) and spSMP.o to the archive,
 *  then copy sparse.a into the SPICE main directory and rename it
 *  SMP.a.  Finally link SPICE.
 *
 *  To be compatible with SPICE, the following Sparse compiler options
 *  (in spConfig.h) should be set as shown below:
 *
 *      REAL                        YES
 *      EXPANDABLE                  YES
 *      TRANSLATE                   NO
 *      INITIALIZE                  NO or YES, YES for use with test prog.
 *      DIAGONAL_PIVOTING           YES
 *      ARRAY_OFFSET                YES
 *      MODIFIED_MARKOWITZ          NO
 *      DELETE                      NO
 *      STRIP                       NO
 *      MODIFIED_NODAL              YES
 *      QUAD_ELEMENT                NO
 *      TRANSPOSE                   YES
 *      SCALING                     NO
 *      DOCUMENTATION               YES
```

```
 *       MULTIPLICATION                  NO
 *       DETERMINANT                     YES
 *       STABILITY                       NO
 *       CONDITION                       NO
 *       PSEUDOCONDITION                 NO
 *       FORTRAN                         NO
 *       DEBUG                           YES
 *       spCOMPLEX                       1
 *       spSEPARATED_COMPLEX_VECTORS     1
 *       spCOMPATIBILITY                 0
 *
 *       spREAL  double
 */

/*
 *  Revision and copyright information.
 *
 *  Copyright (c) 1985,86,87,88,89,90
 *  by Kenneth S. Kundert and the University of California.
 *
 *  Permission to use, copy, modify, and distribute this software and its
 *  documentation for any purpose and without fee is hereby granted, provided
 *  that the above copyright notice appear in all copies and supporting
 *  documentation and that the authors and the University of California
 *  are properly credited.  The authors and the University of California
 *  make no representations as to the suitability of this software for
 *  any purpose.  It is provided `as is', without express or implied warranty.
 */

#ifdef notdef
static char copyright[] =
    "Sparse1.3: Copyright (c) 1985,86,87,88,89,90 by Kenneth S. Kundert";
static char RCSid[] =
    "@(#)$Header: spSMP.c,v 1.2 88/06/24 05:02:42 kundert Exp $";
#endif



/*
 *  IMPORTS
 *
 *  >>> Import descriptions:
 *  spMatrix.h
 *      Sparse macros and declarations.
 *  SMPdefs.h
 *      Spice3's matrix macro definitions.
 */

#include "spice.h"
#include <stdio.h>
#include "spmatrix.h"
#include "smpdefs.h"
#include "spdefs.h"

/* #define NO   0 */
/* #define YES  1 */

#define BALANCE_ARMONICO

#ifdef __STDC__
B-314 static void LoadGmin( char * /*eMatrix*/, double /*Gmin*/ );
```

```
#else
static void LoadGmin( );
#endif

/*
 * SMPaddElt()
 */
int
SMPaddElt( Matrix, Row, Col, Value )
SMPmatrix *Matrix;
int Row, Col;
double Value;
{
    *spGetElement( (char *)Matrix, Row, Col ) = Value;
    return spError( (char *)Matrix );
}

#ifdef BALANCE_ARMONICO

/*
 * SMPaddcElt() This lines was added by me
 */
int
SMPaddcElt( Matrix, Row, Col, Value, iValue )
SMPmatrix *Matrix;
int Row, Col;
double Value, iValue;
{
    ElementPtr Element;

    Element = (RealNumber*) spGetElement( (char *)Matrix, Row, Col );
        Element->Real = Value;
        Element->Imag = iValue;

    return spError( (char *)Matrix );
}

/*
 * SMPcopy() This lines was added by me
 */
int
SMPcopy( SMatrix, DMatrix, Row, Col, Frow, Fcol, EndR, EndC)
SMPmatrix *SMatrix, *DMatrix;
int Row, Col, EndR, EndC, Frow, Fcol;
{
    ElementPtr  e;
        MatrixPtr  Matrix = (MatrixPtr)SMatrix;
        int             i, j, k ,l;
        int             size = Matrix->Size;

        if (Row>size) { Row=size; }
        if (Col>size) { Col=size; }
        if (EndR>size) { EndR=size; }
        if (EndC>size) { EndC=size; }

        for (i = 1; i <= size; i++)
        {
                for (j = 1; j <= size; j++)
                {
                        if (j>=Row && j<Row+EndR && i>=Col && i<Col+EndC)
                        {
                                k = Matrix->ExtToIntRowMap[j];
```

```
                l = Matrix->ExtToIntColMap[i];

                                if ((k != l) OR ((e = (RealNumber *)Matrix->Diag[k]) ==
NULL))
                                {
                                        e = (RealNumber*) spcFindElementInCol( Matrix,
                                                &(Matrix->FirstInCol[l]),
                                                k, l, 0 );
                                }
                                if (e != NULL)
                                {
                                        SMPaddcElt( DMatrix, j-Row+Frow, i-Col+Fcol, e-
>Real, e->Imag );
                                }
                        }
                }
        }
    return spError( (char *)DMatrix );
}

/*
 * SMP3Dto2D() This lines was added by me
 */
int
SMP3Dto2D( SMatrix, DMatrix, Pos, End)
SMPmatrix *SMatrix, *DMatrix;
int Pos, End;
{
    ElementPtr  e;
        MatrixPtr  Matrix = (MatrixPtr)SMatrix;
        int             i, j, k ,l;
        int             size = Matrix->Size;

        for (i = 1; i <= size; i++)
        {
                for (j = 1; j <= size; j++)
                {
                        k = Matrix->ExtToIntRowMap[j];
            l = Matrix->ExtToIntColMap[i];

                        if ((k != l) OR ((e = (RealNumber *)Matrix->Diag[k]) == NULL))
                        {
                                e = (RealNumber*) spcFindElementInCol( Matrix,
                                                &(Matrix->FirstInCol[l]),
                                                k, l, 0 );
                        }
                        if (e != NULL)
                        {
                                SMPaddcElt( DMatrix, Pos+(j-1)*End, Pos+(i-1)*End, e->Real,
e->Imag );
                        }
                }
        }
    return spError( (char *)DMatrix );
}

/*
 * FFTIFFT() This lines was added by me
 */
int
FFTIFFT( num, data, isign )
B-316 double *data;
```

```
int num, isign;
{
// num - number of complex samples
// data[] - array containing the data samples,
// real and imaginary part in alternating order (length: 2*num)
// isign - is 1 to calculate FFT and -1 to calculate inverse FFT

int i, j, m, n;

int mmax, istep;
double wt, theta, wr, wi, wpr, wpi, tempi, tempr;

n = 2 * num;
j = 0;

// bit reversal method
// 1) index 0 need not to be swapped
//        -> start at i=2
// 2) swap scheme is symmetrical
//        -> swap first and second half in one iteration
for ( i =2; i<num; i +=2)
{
        m = n / 2 ;
        while (m >= 2 && j >= m)
        {       // calculate swap index
                j -= m;
                m >>= 1;
        }
        j += m;

        if ( j > i )
        {   // was index already swapped ?
                SWAP( double, data[j], data[i] ); // swap real part
                SWAP( double, data[j+1], data[i+1] ); // swap imaginary part

                if ( j < num )
                {   // swap second half ?
                        SWAP ( double, data[n-j-2] , data[n-i-2] ); // swap real part
                        SWAP ( double, data[n-j-1] , data[n-i-1] ); // swap imaginary part
                }
        }
}

// Danielson-Lanzcos algorithm

mmax = 2;
while ( n > mmax)
{   // each Danielson-Lanzcos step
        istep = mmax << 1;
        theta = isign*(2*M_PI/mmax);
        wpr = cos(theta);
        wpi = sin(theta);
        wr = 1.0;
        wi = 0.0;
        for (m=1; m<mmax ; m+=2)
        {
                for ( i =m; i<=n ; i += istep )
                {
                        j = i +mmax ;
                        tempr = wr*data[j-1] - wi*data[j];
                        tempi = wr*data[j] + wi*data[j-1];
```

```
                        data[j-1] = data[i-1] - tempr;
                        data[j] = data[i]-tempi;
                        data[i-1] += tempr;
                        data[i] += tempi;
                }
                wt = wr;
                wr = wt*wpr - wi*wpi;
                wi = wi*wpr + wt*wpi;
        }
        mmax = istep;
}

    return 0;
}

#endif

/*
 * SMPmakeElt()
 */
double *
SMPmakeElt( Matrix, Row, Col )
SMPmatrix *Matrix;
int Row, Col;
{
    return spGetElement( (char *)Matrix, Row, Col );
}

/*
 * SMPcClear()
 */
void
SMPcClear( Matrix )
SMPmatrix *Matrix;
{
    spClear( (char *)Matrix );
}

/*
 * SMPclear()
 */
void
SMPclear( Matrix )
SMPmatrix *Matrix;
{
    spClear( (char *)Matrix );
}

/*
 * SMPcLUfac()
 */
/*ARGSUSED*/
int
SMPcLUfac( Matrix, PivTol )
SMPmatrix *Matrix;
double PivTol;
{
    spSetComplex( (char *)Matrix );
    return spFactor( (char *)Matrix );
}
```

```
 * SMPluFac()
 */
/*ARGSUSED*/
int
SMPluFac( Matrix, PivTol, Gmin )
SMPmatrix *Matrix;
double PivTol, Gmin;
{
    spSetReal( (char *)Matrix );
    LoadGmin( (char *)Matrix, Gmin );
    return spFactor( (char *)Matrix );
}

/*
 * SMPcReorder()
 */
int
SMPcReorder( Matrix, PivTol, PivRel, NumSwaps )
SMPmatrix *Matrix;
double PivTol, PivRel;
int *NumSwaps;
{
    *NumSwaps = 1;
    spSetComplex( (char *)Matrix );
    return spOrderAndFactor( (char *)Matrix, (spREAL*)NULL,
                             (spREAL)PivRel, (spREAL)PivTol, YES );
}

/*
 * SMPreorder()
 */
int
SMPreorder( Matrix, PivTol, PivRel, Gmin )
SMPmatrix *Matrix;
double PivTol, PivRel, Gmin;
{
    spSetReal( (char *)Matrix );
    LoadGmin( (char *)Matrix, Gmin );
    return spOrderAndFactor( (char *)Matrix, (spREAL*)NULL,
                             (spREAL)PivRel, (spREAL)PivTol, YES );
}

/*
 * SMPcaSolve()
 */
void
SMPcaSolve( Matrix, RHS, iRHS, Spare, iSpare)
SMPmatrix *Matrix;
double RHS[], iRHS[], Spare[], iSpare[];
{
    spSolveTransposed( (char *)Matrix, RHS, RHS, iRHS, iRHS );
}

/*
 * SMPcSolve()
 */
void
SMPcSolve( Matrix, RHS, iRHS, Spare, iSpare)
SMPmatrix *Matrix;
double RHS[], iRHS[], Spare[], iSpare[];
{
    spSolve( (char *)Matrix, RHS, RHS, iRHS, iRHS );
```

```
}

/*
 * SMPsolve()
 */
void
SMPsolve( Matrix, RHS, Spare )
SMPmatrix *Matrix;
double RHS[], Spare[];
{
    spSolve( (char *)Matrix, RHS, RHS, (spREAL*)NULL, (spREAL*)NULL );
}

/*
 * SMPmatSize()
 */
int
SMPmatSize( Matrix )
SMPmatrix *Matrix;
{
    return spGetSize( (char *)Matrix, 1 );
}

/*
 * SMPnewMatrix()
 */
int
SMPnewMatrix( pMatrix )
SMPmatrix **pMatrix;
{
int Error;
    *pMatrix = (SMPmatrix *)spCreate( 0, 1, &Error );
    return Error;
}

/*
 * SMPdestroy()
 */
void
SMPdestroy( Matrix )
SMPmatrix *Matrix;
{
    spDestroy( (char *)Matrix );
}

/*
 * SMPpreOrder()
 */
int
SMPpreOrder( Matrix )
SMPmatrix *Matrix;
{
    spMNA_Preorder( (char *)Matrix );
    return spError( (char *)Matrix );
}

/*
 * SMPprint()
 */
/*ARGSUSED*/
void
SMPprint( Matrix, File )
```

```
SMPmatrix *Matrix;
FILE *File;
{
    spPrint( (char *)Matrix, 0, 1, 1 );
}

/*
 * SMPgetError()
 */
void
SMPgetError( Matrix, Col, Row)
SMPmatrix *Matrix;
int *Row, *Col;
{
    spWhereSingular( (char *)Matrix, Row, Col );
}

/*
 * SMPcProdDiag()
 *    note: obsolete for Spice3d2 and later
 */
int
SMPcProdDiag( Matrix, pMantissa, pExponent)
SMPmatrix *Matrix;
SPcomplex *pMantissa;
int *pExponent;
{
    spDeterminant( (char *)Matrix, pExponent, &(pMantissa->real),
                                              &(pMantissa->imag) );
    return spError( (char *)Matrix );
}

/*
 * SMPcDProd()
 */
int
SMPcDProd( Matrix, pMantissa, pExponent)
SMPmatrix *Matrix;
SPcomplex *pMantissa;
int *pExponent;
{
    double      re, im, x, y, z;
    int         p;

    spDeterminant( (char *)Matrix, &p, &re, &im);

#ifndef M_LN2
#define M_LN2   0.69314718055994530942
#endif
#ifndef M_LN10
#define M_LN10  2.30258509299404568402
#endif

#ifdef debug_print
    printf("Determinant 10: (%20g,%20g)^%d\n", re, im, p);
#endif

    /* Convert base 10 numbers to base 2 numbers, for comparison */
    y = p * M_LN10 / M_LN2;
    x = (int) y;
    y -= x;
```

```
    /* ASSERT
     *  x = integral part of exponent, y = fraction part of exponent
     */

    /* Fold in the fractional part */
#ifdef debug_print
    printf(" ** base10 -> base2 int =  %g, frac = %20g\n", x, y);
#endif
    z = pow(2.0, y);
    re *= z;
    im *= z;
#ifdef debug_print
    printf(" ** multiplier = %20g\n", z);
#endif

    /* Re-normalize (re or im may be > 2.0 or both < 1.0 */
    if (re != 0.0) {
        y = logb(re);
        if (im != 0.0)
            z = logb(im);
        else
            z = 0;
    } else if (im != 0.0) {
        z = logb(im);
        y = 0;
    } else {
        /* Singular */
        /*printf("10 -> singular\n");*/
        y = 0;
        z = 0;
    }

#ifdef debug_print
    printf(" ** renormalize changes = %g,%g\n", y, z);
#endif
    if (y < z)
        y = z;

    *pExponent = x + y;
    x = scalb(re, (int) -y);
    z = scalb(im, (int) -y);
#ifdef debug_print
    printf(" ** values are: re %g, im %g, y %g, re' %g, im' %g\n",
            re, im, y, x, z);
#endif
    pMantissa->real = scalb(re, (int) -y);
    pMantissa->imag = scalb(im, (int) -y);

#ifdef debug_print
    printf("Determinant 10->2: (%20g,%20g)^%d\n", pMantissa->real,
        pMantissa->imag, *pExponent);
#endif
    return spError( (char *)Matrix );
}



/*
 *  The following routines need internal knowledge of the Sparse data
 *  structures.
 */
```

```
/*
 *  LOAD GMIN
 *
 *  This routine adds Gmin to each diagonal element.  Because Gmin is
 *  added to the current diagonal, which may bear little relation to
 *  what the outside world thinks is a diagonal, and because the
 *  elements that are diagonals may change after calling spOrderAndFactor,
 *  use of this routine is not recommended.  It is included here simply
 *  for compatibility with Spice3.
 */

static void
LoadGmin( eMatrix, Gmin )
char *eMatrix;
register double Gmin;
{
MatrixPtr Matrix = (MatrixPtr)eMatrix;
register int I;
register ArrayOfElementPtrs Diag;
register ElementPtr diag;

/* Begin `LoadGmin'. */
    ASSERT( IS_SPARSE( Matrix ) );

    if (Gmin != 0.0) {
        Diag = Matrix->Diag;
        for (I = Matrix->Size; I > 0; I--) {
            if (diag = Diag[I])
                diag->Real += Gmin;
        }
    }
    return;
}




/*
 *  FIND ELEMENT
 *
 *  This routine finds an element in the matrix by row and column number.
 *  If the element exists, a pointer to it is returned.  If not, then NULL
 *  is returned unless the CreateIfMissing flag is true, in which case a
 *  pointer to the new element is returned.
 */

SMPelement *
SMPfindElt( eMatrix, Row, Col, CreateIfMissing )

SMPmatrix *eMatrix;
int Row, Col;
int CreateIfMissing;
{
MatrixPtr Matrix = (MatrixPtr)eMatrix;
ElementPtr Element;

/* Begin `SMPfindElt'. */
    ASSERT( IS_SPARSE( Matrix ) );
        Row = Matrix->ExtToIntRowMap[Row];
        Col = Matrix->ExtToIntColMap[Col];
        Element = Matrix->FirstInCol[Col];
    Element = spcFindElementInCol(Matrix, &Element, Row, Col, CreateIfMissing);
```

```
    return (SMPelement *)Element;
}

/* XXX The following should probably be implemented in spUtils */

/*
 * SMPcZeroCol()
 */
int
SMPcZeroCol( Matrix, Col )
MatrixPtr Matrix;
int Col;
{
    ElementPtr  Element;

    Col = Matrix->ExtToIntColMap[Col];

    for (Element = Matrix->FirstInCol[Col];
        Element != NULL;
        Element = Element->NextInCol)
    {
        Element->Real = 0.0;
        Element->Imag = 0.0;
    }

    return spError( (char *)Matrix );
}

/*
 * SMPcAddCol()
 */
int
SMPcAddCol( Matrix, Accum_Col, Addend_Col )
MatrixPtr Matrix;
int Accum_Col, Addend_Col;
{
    ElementPtr  Accum, Addend, *Prev;

    Accum_Col = Matrix->ExtToIntColMap[Accum_Col];
    Addend_Col = Matrix->ExtToIntColMap[Addend_Col];

    Addend = Matrix->FirstInCol[Addend_Col];
    Prev = &Matrix->FirstInCol[Accum_Col];
    Accum = *Prev;;

    while (Addend != NULL) {
        while (Accum && Accum->Row < Addend->Row) {
            Prev = &Accum->NextInCol;
            Accum = *Prev;
        }
        if (!Accum || Accum->Row > Addend->Row) {
            Accum = spcCreateElement(Matrix, Addend->Row, Accum_Col, Prev, 0);
        }
        Accum->Real += Addend->Real;
        Accum->Imag += Addend->Imag;
        Addend = Addend->NextInCol;
    }

    return spError( (char *)Matrix );
}
```

```
 * SMPzeroRow()
 */
int
SMPzeroRow( Matrix, Row )
MatrixPtr Matrix;
int Row;
{
    ElementPtr  Element;

    Row = Matrix->ExtToIntColMap[Row];

    if (Matrix->RowsLinked == NO)
        spcLinkRows(Matrix);

#if spCOMPLEX
    if (Matrix->PreviousMatrixWasComplex OR Matrix->Complex) {
        for (Element = Matrix->FirstInRow[Row];
            Element != NULL;
            Element = Element->NextInRow)
        {
            Element->Real = 0.0;
            Element->Imag = 0.0;
        }
    } else
#endif
    {
        for (Element = Matrix->FirstInRow[Row];
            Element != NULL;
            Element = Element->NextInRow)
        {
            Element->Real = 0.0;
        }
    }

    return spError( (char *)Matrix );
}
```

- **Código Fuente**

Esta tesis utiliza el codigo de Spice3f5 de Berkeley como base para su módulo de análisis, para consultar dicho código consultar documentación en formato digital.

Puede ser bajado directamente de la pagina oficial de SPICE de Berkeley aquí:

http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/