ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA MECANICA

Aplicaciones de autoLISP en el Diseño Mecánico

TESIS DE GRADO

PREVIA A LA OBTENCION DEL TITULO DE:

INGENIERO MECANICO

PRESENTADA POR:

RENATO VITTORIO PARODI ZAMBRANO

GUAYAQUIL - ECUADOR
1995

AGRADECIMIENTO

Agradezco a mi madre y a mis hermanos por todo el apoyo y comprensión que me brindaron durante toda mi carrera estudiantil.

Al Ing. Federico Camacho B. Director de Tesis por la ayuda y las facilidades prestadas.

DEDICATORIA

A mi hija
A mi esposa
A mis padres
A mis hermanos
A mis sobrinos
A mis amigos

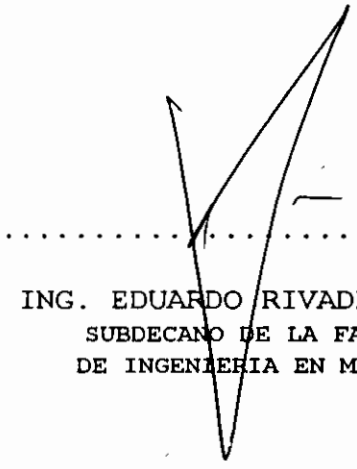
DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en ésta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

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

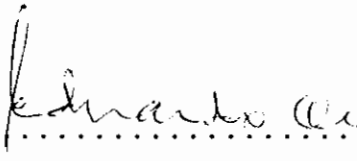
.....

RENATO VITTORIO PARODI ZAMBRANO

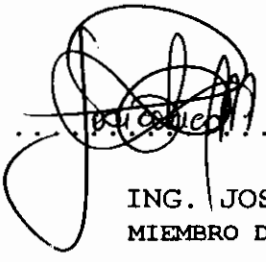


ING. EDUARDO RIVADENEIRA P.
SUBDECANO DE LA FACULTAD
DE INGENIERIA EN MECANICA

ING. FEDERICO CAMACHO B.
DIRECTOR DE TESIS



ING. EDUARDO ORCES
MIEMBRO DEL TRIBUNAL



ING. JOSE PACHECO
MIEMBRO DEL TRIBUNAL

RESUMEN

La elaboración de esta tesis es un puente que sirve de interface para la tesis de DISEÑO DE RESORTES HELICOIDALES y la tesis de DISEÑO DE LEVAS DE PLACA PLANA (ambas tesis de grado desarrollada en la FIM/ESPOL) con AutoCAD. En el caso de las levas éstas podrían ser importadas a MasterCAM para que sean manufacturadas. Con el mismo fin se ha desarrollado en ésta tesis un programa que gráfica automáticamente perfiles de ENGRANAJES DE DIENTES RECTOS. La interface mencionada se logra a través del LENGUAJE LISP.

Se parte detallando una breve reseña del inicio del lenguaje de programación en LISP estableciendo los objetivos para el cual fue creado, así como los principios básicos del manejo del lenguaje AutoLISP y definiendo las funciones que permiten el acceso a éste.

A continuación se elabora y detalla el programa de computadora, para DISEÑO GRAFICO DE RESORTES HELICOIDALES. Este permite automatizar la representación de resortes de tensión y compresión aplicando siempre las normas de dibujo técnico y diseño mecánico que las rigen.

Seguidamente se realiza el programa para el DISEÑO GRAFICO DE LEVAS DE PLACA PLANA, cuya finalidad es leer un archivo de texto el cual nos permite graficar dichas levas con un alto grado de precisión y rapidez.

Posteriormente se desarrolla el programa para el DISEÑO PARAMETRICO DE ENGRANAJES DE DIENTES RECTOS, el cual complementado a los programas de control numérico, permitiría el maquinado de piezas en serie de una manera eficiente y precisa.

Finalmente se diseña una rutina para la AUTOMATIZACION DEL ROTULADO DE PLANOS siguiendo las normas INEN/SI, lográndose rapidez en la elaboración del formato y la uniformidad de su rotulación.

INDICE GENERAL

CAPITULO 6

AUTOMATIZACION DE ROTULADO DE PLANOS

INDICE DE FIGURAS

No. **Pág.**

INDICE DE TABLAS

No. **Pág.**

INTRODUCCION

Históricamente el diseño ha tenido una traslación de ideas o requerimientos técnicos al ir atravesando las etapas de concepción a realización. El diseño se fundamenta en la generación de bosquejos, dibujos, gráficos, etc. Los diseñadores tienen que usar varias herramientas para suministrar exactitud y veracidad a sus representaciones, tales como juegos de escuadras, reglas, compases, etc.

En el campo de la Ingenierí Mecánica los productos son más complejos y la fuerza de trabajo mucho más hábil en la manufactura, algunas veces no es suficiente las vistas en dos dimensiones, dando esto lugar a la utilización de vistas en tres dimensiones e isométricos.

Durante los años 80 se imponen los sistemas CAD dando lugar al desarrollo de sofisticados programas de dibujos. El Diseño Ayudado por Computadora le da al diseñador un potencial en exactitud, precisión, rapidez y economía en la elaboración gráfica de los diseños. Esta tesis le da al diseñador mecánico la potencia de poder graficar en forma automática tres elementos mecánicos importantes dentro de la maquinaria moderna como son el caso de los resortes

helicoidales, levas de placas planas, y engranajes de dientes rectos, dando resultados eficaces en el menor tiempo posible y un alto grado de precisión. Además contiene una rutina que automatiza el formato de rotulación utilizado por los ingenieros mecánicos.

Esta tesis, está basada en el lenguaje de programación AutoLISP, especialmente adaptado para obtener la potencia y presentaciones de AutoCAD. Hablar de AutoCAD es hablar del programa de CAD (Diseño Asistido por Computador) más utilizado dentro de los campos del diseño.

CAPITULO I

INTRODUCCION AL LENGUAJE LISP.

1.1 LENGUAJE DE PROGRAMACION EN LISP..

El lenguaje **LISP**[1] nació hacia 1960. Por esas fechas, se disponía de pocos lenguajes y prácticamente todos los existentes se encontraban en su infancia: el **FORTRAN** comenzaba a extenderse, el **BASIC** flotaba en una nebulosa. Desde entonces acá, importantes mejoras han hecho algo más que transformar estos lenguajes. Pero el LISP ha permanecido invariable y no se ha visto la necesidad de perfeccionarlo estando completo, desde su nacimiento. Esta "perfección" conseguida desde 1962 se debe a John Mc-CARHY, su inspirado creador. Desde entonces, el LISP ha vivido y prosperado, pero sobretodo se ha aplicado durante sus dos primeros lustros en el campo de la inteligencia artificial, en el que se obtuvieron los primeros éxitos y en que el **LISP** sobresalió del resto de los grandes lenguajes. El último decenio ha visto aparecer en volumen creciente

de trabajos tantos teóricos como prácticos sobre el **LISP**. Existe una implantación de **LISP** en prácticamente todos los ordenadores (desde los más potentes hasta los más humildes microprocesadores) y es, veinte años después de su nacimiento, uno de los mejores lenguajes de programación existentes.

1.2 AutoLISP, VERSION ESPECIFICA DE LISP.

AutoLISP[3] es un conjunto de lenguaje de programación **CommonLISP**[1]. Puesto que AutoLISP está diseñado para funcionar desde un dibujo de AutoCAD[2], se han seleccionado las características de LISP más adecuadas para este fin y además se han añadido otras nuevas, sobre todo en lo relativo a la manipulación de entidades de dibujo, acceso a la base de datos de AutoCAD.

Además se puede escribir directamente instrucciones en AutoLISP desde la línea de órdenes del dibujo en AutoCAD. AutoLISP evalúa inmediatamente esa lista y ofrece un resultado. Y si la expresión contiene definiciones de funciones o variables, quedan cargadas en la memoria para su utilización posterior.

Incluso el usuario puede definir ordenes propias personalizadas para ejecutar programas en AutoLISP. Una vez que la nueva orden está definida, se añade a las existentes en AutoCAD y se utiliza como una más en el momento que interese. También existe la posibilidad de redefinir ordenes existentes de AutoCAD para que funcione de manera diferente de la habitual.

Una de las más importantes potencialidades de AutoLISP es el acceso directo a la Base de Datos de un dibujo en AutoCAD. Toda la información del dibujo se encuentra en ella: capas, estilos de texto, tipos de líneas, sistemas de coordenadas, etc.

Se puede utilizar AutoLISP para modificar esa base de datos o para extraer la información que interese de ella con objeto de importarla, por ejemplo, en una base de datos externa.

1.3 TIPOS DE OBJETOS EN AutoLISP.

Los objetos en AutoLISP representan todos los tipos de componentes de un programa. En esencia son dos, listas y símbolos. Además es posible incluir como

elementos de listas valores concretos, ya sean numéricos o textuales. Atendiendo a sus características se puede definir varios tipos de objetos en AutoLISP:

a) **Lista:** es un objeto compuesto de:

- Paréntesis de apertura.
- Uno o más elementos separados por al menos un espacio en blanco.
- paréntesis de cierre.

Los elementos de la lista pueden ser símbolos, valores concretos (constantes numéricas o textuales), o también otras listas incluidas.

b) **Elemento:** cualquiera de los componentes de una lista.

c) **Atomo:** representa una información indivisible. El valor concreto o un símbolo de variable que contiene un valor son átomos. Una lista o una función definidas por el usuario definida por el usuario no son átomos.

d) **Símbolo:** cualquier elemento de lista que no sea un valor concreto. El símbolo puede ser un nombre de variable, un nombre de función definida por el usuario o un nombre de comando AutoLISP.

e) **Entero:** valores numéricos enteros, ya sean explícitos o contenidos en variables.

f) **Reales:** valores numéricos con precisión de coma flotante.

g) **Cadenas:** valores textuales que contienen cadenas de caracteres (código ASCII[4]). Deben ir entre comillas.

h) **Descriptores de fichero o archivo:** valores que representan un archivo abierto para lectura o escritura desde un programa en AutoLISP. Por ejemplo File:#5c90.

i) **Nombre de entidades de AutoCAD:** valores que representan el nombre de una entidad en la Base de Datos. Por ejemplo Entity name 60000A56.

j) **Conjunto designado de AutoCAD:** valores que representan una designación de entidades de dibujos. Por ejemplo, Selection set: 3.

k) **Función de usuario:** es un símbolo que representa el nombre de una función definida por el usuario.

l) **Función inherente o Comando:** es un símbolo con el nombre de un comando predefinido de AutoLISP. A veces se les llama Subrutinas.

1.4 AJUSTE DE LA MEMORIA RAM.

Usted puede controlar la cantidad de memoria RAM[4] disponible para que AutoCAD realice sus cálculos. Algunos dibujos complejos, al ser combinados con ciertos comandos de tercera dimensión tal como **HIDE**[3], puede exceder el espacio de trabajo por definición, visualizándose en pantalla un mensaje del DOS que nos

indica que estamos fuera de memoria **RAM -[Out of RAM]**-con lo cual se cierra el archivo y salimos de AutoCAD.

Para que no suceda, use el comando **SET**[4] del DOS para ajustar la cantidad de memoria **RAM** requerida por AutoCAD, agregando la línea:

SET ACADFREERAM=25

en el archivo **AUTOEXEC.BAT**[4]. Esto establece la cantidad de memoria **RAM** en 25K. De todas maneras, el valor máximo que acepta **ACADFREERAM**[5] es 30K.

En contraposición, especificar un valor más bajo para **ACADFREERAM** aumentará la capacidad de memoria **RAM**, lo cual permite que AutoCAD trabaje más fijo. Por tal motivo, puede que convenga reducir este valor hasta el mínimo: **SET ACADFREERAM=5.**

En casos de que el mensaje: fuera de memoria **RAM -[Out of RAM]-** haya aparecido y que después de haber hecho ésta modificación al archivo **AUTOEXEC.BAT** persista, incremente el valor en una unidad (de 25 a 26) y así sucesivamente hasta que desaparezca. Afinar este valor a veces toma tiempo, pero las ventajas son apreciables evitará la repentina y perjudicial

suspensión de su trabajo por no disponer AutoCAD de suficiente memoria RAM.

1.4.1 MEMORIA USADA POR AutoCAD Y AutoLISP.

El comando **SET** del sistema operativo DOS permite ajustar la cantidad de memoria **RAM** que AutoCAD emplea como área de trabajo, así como la cantidad de memoria **RAM** que se aparta para las funciones de AutoLISP. También puede controlar la cantidad y ubicación de la **Memoria Extendida**[4] y (o) **Expandida**[4] que usa AutoCAD para trabajar con los archivos temporales[5].

Si se dispone de un computador con más de 640K de memoria **RAM** es posible aprovechar la capacidad de la ampliación de la memoria instalada para imprimir una mayor velocidad de operación al programa.

Creando un **DISCO VIRTUAL**[4], proceso que dependerá del tipo y versión del sistema operativo utilizado, se puede copiar los archivos **OVERLAY**[5] (**ACAD.OVL**, **ACAD0.OVL**,

ACAD2.OVL, ACAD3.OVL, ACADVS.OVL Y ACADL.OVL.). Como el acceso a la información directamente a **RAM** es mucho más rápido que el disco duro, de ésta forma se puede aumentar la velocidad de ejecución considerablemente.

Hay que tener en cuenta que la información en memoria **RAM** se pierde al apagar el computador; por lo tanto, la copia de los archivos anteriormente descritos habrá que hacerla cada vez que se quiera cargar el programa.

1.4.2 MEMORIA PARA FUNCIONES DE AutoLISP.

El comando **SET** del DOS también fija la memoria que AutoCAD necesita para trabajar con AutoLISP. En lo que concierne a este lenguaje de programación, AutoCAD usa dos tipos de memoria: la memoria **HEAP** y la memoria **STACK**. En términos muy generales la memoria **HEAP** es la que usa para almacenar funciones y variables que se conocen como **"espacio de nodos" -[node space]-**; mientras que la memoria **STACK** almacena argumentos y

resultados parciales durante la evaluación de expresiones.

El valor por definición para la memoria **LISPHEAP** es de 40K bytes, mientras que para **LISPSTACK** es de 3K bytes.

Por lo general, estos valores son suficientes. No obstante, en el caso de visualizar el mensaje:

Insufficient node space

-[Espacio de nodos insuficiente]-

mientras ejecuta un programa en AutoLISP aumenta estos valores de la misma manera como lo hizo con **ACADFFREERAM** tecleando:

SET LISPHEAP=42000

SET LISPSTACK=3000

Si el programa se torna complejo, los valores para la memoria **HEAP** y **STACK** pueden variarse, pero teniendo en cuenta que en ningún caso la suma de las dos áreas de memoria (**HEAP** y **STACK**) pueden pasar de 45000 bytes. (Estos valores pueden ser incluidos en un archivo **AUTOEXEC.BAT** en caso que se trabaje con frecuencia con

programas escritos en AutoLISP, combinados con AutoCAD).

1.5 PRINCIPIOS BASICOS

1.5.1 CARGA DE LOS EDITORES DE TEXTOS.

El programa AutoCAD contiene la opción de poder cargar directamente el Sistema Operativo sin necesidad de abandonar el trabajo que se estuviese haciendo en ese momento, pudiendo de ésta manera disponer del procesador de texto que tuviéramos configurando o, en su defecto, del **EDLIN**[4] del propio Sistema Operativo.

Al igual que se hizo con la zona de memoria, los procesadores de texto también deben ser configurados antes de ser utilizados, para lo que tendremos que editar el archivo ACAD.PGP contenido en el paquete AutoCAD.

Este archivo está compuesto de las siguientes líneas que se detallan a continuación:

```
CATALOG,DIR/W,24000,*Files:0
DEL,DEL,24000,File to delete:,0
DIR,DIR,24000,File specification:,0
EDIT,EDLIN,40000,File to edit:,0
SH,,24000,*DOS Command:,0
SHELL,,125000,*DOS Command:,0
TYPE,TYPE,24000,File to list:,0
```

TABLA 1.1 Funciones externas de AutoCAD[3]

Como se observará, el archivo está compuesto por líneas que contienen comandos del DOS. Estas a su vez, constan de cinco elementos de información, separados por comas, que nos determinan:

1. El comando del DOS que es usado desde el editor de dibujo.
2. El nombre del programa que se va a utilizar.
3. El espacio de memoria requerido para ejecutar el comando.
4. La respuesta que nos daría el comando al ser ejecutado (Si se desea que ésta no aparezca, simplemente se quitará, pero dejaremos las dos comas separadoras.)

5. El número de código usado por AutoCAD para retornar al editor de dibujo.

De todas las líneas anteriores, la que nos interesa en este apartado es la que comienza con la función **EDIT,** ya que es ésta la línea usada por AutoCAD para llamar al procesador de texto (en este caso el del sistema operativo).

Si quisiéramos trabajar con otro procesador de texto diferente al **EDLIN**, editaríamos esa línea sustituyendo **EDLIN** por el nombre del nuevo procesador que se desea utilizar, adaptando asimismo el espacio de memoria.

Así, para introducir el WORDSTAR, situaríamos:

EDIT,WS,80000,,0

y para el WORDPERFECT:

EDIT,WP,145000,,0

Otros procesadores requerirán diferentes espacios de memoria, dependiendo de la computadora en que se trabaje y de la configuración que se haya hecho.

1.5.2 CONVENIOS SINTACTICOS DE AutoLISP.

Como todos los lenguajes de programación, AutoLISP tienen una regla sintácticas que deben cumplirse cuando se hacen un programa. Estas, aunque mínimas son imprescindibles y se relacionan a continuación.

EL PUNTO Y COMA.- Cuando una cadena va precedida del punto y coma ésta línea no se ejecuta, aunque se guarda como parte del programa sirviendo solamente para documentar o informar sobre él (trabaja exactamente igual que la orden REM de BASIC).

Cuando el programa es ejecutado, todo lo escrito a continuación del punto y coma se ignora, pudiéndose acceder a ellos solamente a través de un editor de texto. En el ejemplo:

```
(PROMPT "Esto será escrito") ;Esto se ignora
```

Solamente será ejecutable lo encerrado entre paréntesis, ignorándose lo escrito a continuación del punto y coma.

LAS COMILLAS.- Como se ha visto en el ejemplo anterior, cuando se precisa de una cadena sea escrita literalmente, se debe encerrar entre comillas, puesto que de ésta forma se comunicara al interprete AutoLISP que no es un procesamiento.

EL APÓSTROFO.- Se utiliza para introducir información literal de AutoLISP de forma que todo lo que valla a continuación sea tomado literalmente para su procesamiento (trabaja igual que la función QUOTE del interprete AutoLISP). En el ejemplo:

(GETANGLE `(0,0) "Entre ángulo")

Se tomaran 0,0 como coordenadas de un punto base y luego se pedirá las coordenadas de otro punto.

LOS PARENTESIS.- Son esenciales en escritura de un programa en AutoLISP, puesto que las ordenes deben ir encerradas entre ellas.

Usándolos de forma anidada, permiten escribir secuencias de ordenes relacionadas entre sí y

como en cualquier otro lenguaje, el número de paréntesis de cierre deben coincidir con los de aperturas. Por ejemplo:

```
(*** (***** (**)))
```

Si por cualquier causa no se situaran los paréntesis de cierre correctamente, AutoLISP daría un mensaje de error en la línea de ordenes de AutoCAD.

1.5.3 ELABORACION DE UN PROGRAMA EN AutoLISP.

Una vez hechas todas las operaciones para situar las zonas de memoria anteriores y conociendo las normas sintácticas del intérprete, estaremos en condiciones de comenzar el estudio de las instrucciones de AutoLISP.

(DEFUN <simb> <lista argum> <expr>....)DEFINIR FUNCION.- En AutoLISP una función es directamente un programa, pues su evaluación ofrece un resultado una vez cargado el archivo que contiene su definición. Así, un archivo .LSP puede contener muchos programas, según el número de función definidas para él. Para evaluarlas no

es necesario volver al archivo; basta con cargarlo una sola vez.

El comando "**DEFUN**" se utiliza precisamente para definir funciones o programas de AutoLISP. Va seguido de <**simb**>, que es el símbolo o nombre de la función a definir. Conviene que los nombres de símbolos contengan como máximo seis caracteres, por razones de espacio ocupado en la memoria.

A continuación se especifica la lista de argumento (al ser una lista debe ir entre paréntesis), que puede estar vacía o contener varios argumentos y, opcionalmente, una barra inclinada y los nombres de uno o más símbolos locales de la función.

Los argumentos o símbolos globales son variables que se almacenan en la memoria y permanecen en ella lo mismo que los nombres de la función definidas, de forma que pueden ser utilizados por otros programas o funciones de AutoLISP.

Los símbolos locales son opcionales, estos se almacenan en la memoria solo temporalmente, mientras la función definida está en curso. En cuanto termina desaparece de la memoria, por lo que no pueden ser usados por otros programas o funciones.

El último elemento de la función definidora **DEFUN** es la expresión en AutoLISP que va a servir de definición **<simb>**. Esta expresión puede ser tan compleja como se quiera, ejecutando otras funciones definidas, cargando archivos .LSP, etc.

Por ejemplo, se define una función llamada "seno", cuya utilidad es calcular el seno de un ángulo. Su definición es la siguiente:

Orden:(DEFUN seno (x) (SETQ xr (* PI (/ x 180))) (SETQ s (SIN xr)))

(SETQ <simb> <expr>)ATRIBUIR VALORES.- Este comando atribuye el valor de la expresión **<expr>** a la variable cuyo nombre es **<simb>**. Si las

variables no están previamente definidas, las crea. La función devuelve el valor atribuido a la última variable. **SETQ** es la función básica de atribución de valores en AutoLISP. Por ejemplo:

(SETQ a 5) (SETQ a b) (SETQ a "Buenos días")

1.5.4 EDICION DE UN PROGRAMA

Los programas son creados en un simple archivo de texto y escritos con cualquier editor de los existentes en el mercado, como ya se dijo anteriormente.

Si optásemos por usar **EDLIN** trabajaríamos de la siguiente forma:

Una vez situados en el editor de dibujo de AutoCAD, entraremos en la línea de comandos:

Command:EDIT

Pasando a pantalla de texto AutoCAD nos pedirá:

File to edit: **Archivo a Editar:**

a lo que contestamos con el nombre que deseemos dar al archivo que vamos a editar. Por ejemplo:

EJEMPLO.LSP

1.5.5 CARGADA DE UN PROGRAMA

Los programas en AutoLISP son generalmente archivos ASCII de texto con extensión .LSP y para poder usarlos, deben ser cargados mientras usted dibuja en el editor de dibujos. Los archivos que contienen estos programas se deben encontrar en el directorio actual de trabajo o, en caso contrario, se debe especificar la vía de acceso -[path[4]]- de la siguiente manera:

```
Command: (LOAD "U.de disco:\\directorio\\archivo")
```

Si los archivos se encuentran en el directorio actual de trabajo se teclea:

```
Command: (LOAD "ejemplo.lsp")
```

No es necesario incluir la extensión .LSP junto con el nombre del archivo, pero sí toda la expresión de AutoLISP entre paréntesis.

Una vez cargado el programa en el área de apoyo del comando aparecerá el nombre de ese programa o el de la función dentro de ese programa.

Algunas funciones deben ser solicitadas entre paréntesis, después de cargadas.

Se puede crear un archivo llamado **ACAD.LSP** con una serie de programas incluidos en él. El contenido de este archivo se carga automáticamente en la memoria cada vez que se entra en el editor de dibujo. Para todos los programas contenidos en **ACAD.LSP** no es necesario utilizar el comando **LOAD**. Este archivo es muy útil cuando interesa disponer de determinadas rutinas cada vez que se entra en el dibujo.

CAPITULO II

FUNCIONES DE AutoLISP

2.1 FUNCIONES DE PANTALLAS.

AutoLISP dispone de cuatro instrucciones para situar datos en pantalla, aunque sólo utiliza dos. Las cuatro instrucciones son **"PRINC","PRIN1","PRINT","PROMPT"**.

(PROMPT <mensj>) ESCRIBE MENSAJE.- Este comando visualiza la cadena de texto <mensj> en la línea de ordenes de AutoCAD y devuelve **"nil"**. En configuraciones de dos pantallas visualiza el mensaje en ambas. Por ésta razón es preferible a otras funciones de escrituras como **"PRINC"**. Ejemplo:

```
Command:(PROMPT "Hola")
Holanil
```

Se observa que el mensaje se visualiza sin comillas. La evaluación de **"PROMPT"** devuelve **"nil"** a continuación.

(PRIN1 <expr>) ESCRIBIR EXPRESION.- Este comando escribe la expresión indicada <expr> y devuelve la propia expresión. La expresión no tiene por qué ser una cadena de texto; puede ser cualquiera. Se escribe sin añadirse sin añadirse ningún espacio en blanco o interlínea. Ejemplo:

```
(SETQ x 25.0 a "Hola")
(PRIN1 x)  visualiza y devuelve 25.0
(PRIN1 a)  visualiza y devuelve "Hola"
```

(PRINT <expr>) ESCRIBIR CON INTERLINEA.- Totalmente idéntica a "PRIN1", salvo que salta a una nueva línea antes de visualizar y añade un espacio en blanco después. Ejemplo:

```
Command:(PRIN1 "Hola")
"Hola""Hola"
Command:(PRINT "Hola")
"Hola" "Hola"
```

(PRINC <expr>)ESCRIBIR EXPRESION ASCII.- Este comando funciona de manera análoga al de "PRIN1", escribiendo la expresión en pantalla. Pero la diferencia es que escribe todo el juego de caracteres ASCII[4], incluidos los dos números superiores a 126. Ejemplo:

```
Command: (PRINC "manana")
escribe "manana"  devuelve "ma\244ana"
```

(TERPRI) TERMINAR ESCRITURA.- Este comando mueve el cursor al comienzo de una nueva línea. Se utiliza para saltar de línea cada vez que se escribe mensajes en el área de órdenes de la pantalla gráfica. Existen otros procedimientos como el carácter de control **"\n"** incluido dentro del texto de mensaje, pero a menudo interesa **"TERPRI"** por razones de claridad. Esta función no se utiliza para entradas/salidas de archivos. Para ello se emplea **PRINT** o **PRINC**. Ejemplo:

```
Command: (PROMPT "Hola") (TERPRI)
Hola
nil
```

(GRAPHSCR) PANTALLA GRAFICA.- Esta función o comando de AutoLISP conmuta de pantalla de texto a pantalla gráfica. Su efecto es el mismo que pulsar la tecla de función **"F1"** desde AutoCAD. Su evaluación devuelve siempre **"nil"**. Se utiliza para asegurar que el Editor de Dibujo se encuentre en pantalla gráfica, cuando es preciso procesar puntos, entidades y demás elementos gráficos.

(COMMAND <argumento>)ACCESO A ORDENES DE AutoCAD.- Este comando sirve para llamar a las órdenes de AutoCAD desde AutoLISP y siempre devuelve **"nil"**. Sus argumentos son nombres de órdenes y opciones de AutoCAD. Cada argumento es evaluado y enviado a AutoCAD como respuesta a sus mensajes o preguntas.

Los nombres de órdenes y de opciones se envían como cadenas de texto y, por lo tanto, debe ir entre comillas. Los puntos de 2D o 3D se envían como listas de dos o tres números reales. Ejemplo:

```
Command: (COMMAND "line" p1 p2 "")
```

En el ejemplo se dibuja una línea desde el punto p1 al punto p2, éstas variables deberían haber estado previamente definidos por el comando **SETQ**.

2.2 FUNCIONES DE ENTRADAS DE DATOS.

(GETPOINT [<pto>][<mensaje>])INTRODUCIR UN PUNTO.- Este comando aguarda a que el usuario introduzca un punto, bien directamente por teclado o señalado en pantalla, y devuelve las coordenadas de ese punto. Se puede indicar opcionalmente un mensaje para que se

(GETSTRING [<cr>] [<mensaje>] INTRODUCIR TEXTO.- Con ésta función se carga una variable con una cadena de texto introducida desde el teclado, debiendo situar una bandera (a excepción de nil), si se van a utilizar espacios en blanco. Esta bandera debe ir colocada detrás de la instrucción. Ejemplo:

```
Command: (GETSTRING T "Entre el texto")
```

(GETINT [<mensaje>] INTRODUCIR NUMERO ENTERO.- Este comando o función de AutoLISP acepta un número entero introducido por el usuario. El valor debe encontrarse entre -32768 y +32767. Ejemplo:

```
Command: (GETINT "Introducir número")
```

(QUOTE <expr>) LITERAL DE EXPRESION.- Este comando evita que se evalúen los símbolos. Se puede emplear con cualquier expresión de AutoLISP y siempre devuelve el literal de esa expresión, es decir sin evaluarlo. Ejemplo:

```
Command: (QUOTE (SETQ 25))
(SETQ 25)
```

visualice en la línea de órdenes y también un punto de base. En éste caso AutoCAD visualizará una línea elástica entre el punto de base y la posición del cursor de forma dinámica. Ejemplo:

```
Command: (GETPOINT `(5 2) "Segundo punto")
```

En el ejemplo se ha especificado un punto base de coordenadas (5,2). Aparece una línea elástica entre ese punto de base y la posición del cursor.

(GETREAL [<mensaje>]) INTRODUCIR NUMERO REAL.-

Esta instrucción asegura que la entrada se haga con un solo número real, no admitiendo ángulos ni caracteres. Ejemplo:

```
Command: (GETREAL "Numero real")
```

(GETDIST [<pto>] [<mensaje>]) INTRODUCIR DISTANCIA.-

Esta instrucción, como la anterior, asegura una entrada en números reales, diferenciándose en que permite introducir por pantalla uno o dos puntos. Si se especifica un punto base, AutoLISP visualiza una línea elástica entre ese punto y la posición del cursor. Ejemplo:

```
Command: (GETDIST p1 "Primer punto")
```

2.3 OPERADORES ARITMETICOS

Los cuatro comando de AutoLISP que efectúan las cuatro operaciones aritméticas básicas son Sumar, Restar, Multiplicar y Dividir.

(+ <num> <num>...)SUMAR.- Devuelve la suma de todos los números especificados a continuación del comando. Ejemplo:

```
Command:(+ 3 5 7)
15
```

(- <num> <num>...)RESTAR.- Devuelve el resultado de restar el primer número de todos los demás. Es decir, se resta al primero la suma de todos los demás. Ejemplo:

```
Command:(- 12 5 3)
4
```

(* <num> <num>...)MULTIPLICAR.- Evalúa el producto de todos los números indicados. Ejemplo:

```
Command:(* 5 6 3)
90
```

(/ <num> <num>...)DIVIDIR.- Divide el primer número por todos los demás. Es decir, divide el primer número por el producto de los demás. Ejemplo:

```
Command: (/ 80 5 3)
5
```

2.4 FUNCIONES MATEMATICAS

Aparte de las cuatro funciones ya vistas, AutoLISP contiene otras funciones matemáticas que servirán para desarrollar los cálculos que fuesen necesarios.

(ABS <num>)VALOR ABSQLUTO.- Este comando de AutoLISP devuelve el valor absoluto del número que se indique. Ejemplo:

```
Command: (ABS -25.78)
25.78
```

(GCD <num1> <num2>)MAXIMO COMUN DENOMINADOR.- Este comando devuelve el máximo común denominador de los números que se indiquen. Estos tienen que ser enteros. Ejemplo:

```
Command: (GCD 45 80)
5
```

(COS <ang>)COSENO.- Este comando devuelve el coseno del ángulo expresado en radianes. Ejemplo:

```
Command:(COS PI)
-1.0
```

(SIN <ang>)SENO.- Este comando devuelve el seno del ángulo expresado en radianes. Ejemplo:

```
Command:(SIN PI)
0.0
```

(EXPT <base> <pot>)POTENCIA.- Este comando devuelve el número base elevado a la potencia indicada. Ejemplo:

```
Command:(EXPT 5.0 3)
125.0
```

(SQRT <num>)RAIZ CUADRADA.- Este comando devuelve la raíz cuadrada del numero indicado. Ejemplo:

```
Command:(SQRT 25.0)
5.0
```

2.5 OPERADORES BOOLEANOS

Cuando se comparan unas acciones mediante unas pruebas condicionales, el resultado tiene un valor booleano de VERDADERO o FALSO, cumpliéndose la acción si ésta fuera verdadera.

En AutoLISP el valor de una expresión es normalmente **T** o **NIL** que son las constantes que equivalen a VERDADERO o FALSO respectivamente.

Estos valores constantes, al igual que los números, pueden ser asignados a una variable utilizando la función **"SETQ"**. Cuando una variable está vacía (no contiene ningún dato) se dice que es **NIL**. En otros lenguajes decimos que la variable está vacía cuando su valor es 0. Esto no sucede en AutoLISP, ya que el cero se considera un valor y por lo tanto, esa variable no sería igual a **NIL**.

2.6 OPERADORES LOGICOS

Los operadores lógicos en AutoLISP trabajan igual que en cualquier otro lenguaje, comparando dos o más

números con arreglos a la información pedida, basándose en el resultado de CIERTO o FALSO.

(= <átomo><átomo>...)IGUAL.- Compara todos los valores de los átomos especificados. Si son todos iguales, el resultado de la evaluación es **T**. En caso contrario devuelve **NIL**. Ejemplo:

```
Command:(= "Hola" "Hola")
T
```

(/= <átomo><átomo>)NO IGUAL.- Compara los valores y devuelve **T** si efectivamente son distintos. La función sólo está definida para dos argumentos. Ejemplo:

```
Command:(/= "Hola" "Días")
T
```

(< <átomo><átomo>...)MENOR.- Devuelve **T** si el valor del primer átomo es menor que el segundo. Si se indica más de dos átomos, el resultado seria **T** si cada uno de ellos es menor que el siguiente. Ejemplo:

```
Command:(< 15 5 2)
nil
```

(<= <átomo><átomo>...)MENOR o IGUAL.- Similar al comando anterior, funciona de la misma manera pero comparando la condición menor o igual. Ejemplo:

```
Command: (<= 15 15 5)
nil
```

(> <átomo><átomo>...)MAYOR.- Compara todos los átomos especificados y devuelve **T** sólo si cada valor es mayor que el siguiente; es decir, se encuentra ordenado de mayor a menor. Ejemplo:

```
Command: (> 30 15 5)
T
```

(>= <átomo><átomo>...)MAYOR o IGUAL.- Similar al anterior, establece la comparación Mayor o Igual. Devuelve T sólo si cada átomo es mayor o igual que el siguiente. Ejemplo:

```
Command: (>= 30 30 15)
T
```

(AND <expr>...)"Y" LOGICO.- Este comando realiza **"Y"** lógico de una serie de expresiones que son las indicadas. Evalúa todas las expresiones y devuelve **T** si todas son VERDADERAS. Ejemplo:

```
Command: (AND (= "Hola" "Hola") (/= 5 3))
T
```

(OR <expr>...)"O" LOGICO.- Realiza el "O" lógico de la serie de expresiones indicadas. Evalúa todas las expresiones y devuelve **NIL** si todas son FALSAS. Ejemplo:

```
Command: (OR (= "Hola" "Hola") (/= 5 5))
T
```

(NOT <expr>)NO LOGICO.- Devuelve **T** si las funciones analizadas son **NIL**, puesto que **NOT** niega el argumento. Ejemplo:

```
Command: (NOT (= "Hola" "Días"))
T
```

(EQUAL <expr><expr>)IDENTIDAD LOGICA.- Es muy parecida a "=" usándose para comparar acciones más complejas, como pueden ser las listas. Ejemplo:

```
Command: (EQUAL `(3 5) `(7 8))
nil
```

3.7 FUNCIONES DE ESTRUCTURAS

En AutoLISP, como en cualquier otro lenguaje de programación, existen estructuras de control que

tienen como misión repetir una serie de instrucciones. Estas estructuras son conocidas corrientemente por el nombre de **"BUCLES"**.

(IF<cond><acción-cierto> [<acción-falsa>])CONDICION.-
Este comando es una de las estructuras básicas de programación. La función **IF** evalúa la expresión indicada como condición. Si el resultado es **T**, entonces pasa a evaluar la condición de VERDADERO. Si es **NIL** evalúa la condición de FALSO. Ejemplo:

```
Command:(IF (= a b) (PROMPT "Son iguales")
(PROMPT "Son diferentes"))
```

(COND(<cond1><res1>)(<cond2> <res2>)...)CONDICIONAL.-
Este comando permite establecer una serie de condiciones especificando diferentes actuaciones según cuál sea la primera condición en cumplirse. Los argumentos de **COND** son un número cualquiera de listas, tantas como condiciones se pretenda establecer. Ejemplo:

```
Command:(COND (= c 1) (SETQ n T)
              (PROMPT "Correcto") (TERPRI))
              (T (PROMPT "Incorrecto")))
```

(PROGN <expr>...)SECUENCIA CONSECUTIVA.- Este comando admite como argumentos todas las expresiones que se indiquen y las evalúa secuencialmente. Devolviendo el valor de la última expresión evaluada.

En la instrucción **IF** se vio que sólo ejecuta una función y, como excepción, otra más. Con ésta función se podrá situar un grupo de instrucciones dentro del mismo **IF**. Ejemplo:

```
Command:(IF (= a b)
            (PROGN
             (.........)
             (.........)
             )
        )
```

(REPEAT <num> <expresión>...)REPETIR N VECES.- Este comando es la estructura básica de la secuencia repetitiva, cuando se requiere repetir una expresión un determinado número de veces. Ejemplo:

```
Command:(REPEAT n
               (.........)
               (.........)
```

```
)
```

(WHILE <cond> <acción>...)REPETIR SEGUN CONDICION.-

Es otro comando para establecer secuencias repetitivas en el programa. En éste caso el ciclo no se repite un determinado número de veces, sino mientras se cumpla la condición especifica. Esta condición sirve, pues, como elemento de control de la repetitiva.

Mientras el resultado de evaluar la condición sea diferente de **NIL**, el comando **WHILE** evalúa la expresión indicada que puede ser tan compleja como se desee. Ejemplo:

```
Command: (WHILE a
              (.........)
              (.........)
              (IF (= a b) (SETQ a nil))
         )
```

2.8 FUNCIONES DE ANGULOS

Debido a que todas las entidades no se dibujan a 90 grados, es muy importante conocer los procedimientos que tiene AutoLISP para trabajar con ángulos.

Los ángulos en AutoLISP son medidos en radianes, mientras que en AutoCAD, generalmente, se trabaja en grados decimales. Es muy importante que esté muy clara la diferencia que existe en éste apartado entre AutoCAD y AutoLISP, porque esto deriva en una Forma de Trabajo muy distinta.

(ANGLE <pt1> <pt2>)ANGULO DEFINIDO POR DOS PUNTOS.- Este comando devuelve el ángulo determinado por la línea entre dos puntos especificados. El valor del ángulo se mide respecto al eje X actual del dibujo. El ángulo se mide en radianes y su sentido positivo es el antihorario. Ejemplo:

```
Command: (ANGLE '(5 2) p1)
```

(DISTANCE <pt1> <pt2>)DISTANCIA ENTRE DOS PUNTOS.- Este comando devuelve la distancia entre los dos puntos especificados. Ejemplo:

```
Command: (DISTANCE '(5 2) p1)
```

(POLAR <pt><ang> <dist>)PUNTO EN COORDENADAS POLARES.- Este comando devuelve un punto obtenido en coordenadas relativas polares a partir del punto

especificado. Es decir, desde el punto se lleva una distancia en un ángulo y se obtiene el nuevo punto. Ejemplo:

```
Command: (POLAR '(5 4) (/ PI 2) dis)
```

(GETANGLE [<pt>][<mensaje>]) INTRODUCIR ANGULO.- Este comando espera a que el usuario introduzca un ángulo y devuelve el valor de ese ángulo. El ángulo se puede introducir tecleando directamente en el formato actual de unidades (podría ser, por ejemplo, grados/minutos/segundos), pero el valor devuelto por **"GETANGLE"** será siempre un número real en radianes. Ejemplo:

```
Command: (GETANGLE p1 "Segundo punto")
```

(GETORIENT [<pt>][<mensaje>]) ANGULO SEGUN ORIGEN.- A diferencia de **"GETANGLE"**, que mide siempre los ángulos desde la posición habitual de 0 grados y positivo en el sentido antihorario, el comando **"GETORIENT"** considera los ángulos con el origen 0 grados y el sentido y el sentido actuales definidos por la orden **UNITS**[2]. Ejemplo:

```
Command: (GETORIENT p1 "Segundo punto")
```

2.9 FUNCIONES DE CADENAS

(STRCASE<cad>[<expr>])CAMBIO A MAYUSCULA O MINUSCULA.- Este comando toma la cadena de caracteres que se indique y la convierte en mayúscula o minúscula dependiendo del valor que se indique. Ejemplo:

```
Command: (STRCASE "hola")
HOLA

Command: (STRCASE "HOLA" T)
hola
```

(STRCAT <cad1> <cad2>...)CONCATENAR CADENA DE TEXTO.- Este comando devuelve una cadena que es la concatenación de todas las cadenas indicadas. Por ejemplo si en la variable cad1 se almacena "Ejemplo de" y en la variable cad2 se almacena "concatenación" se tendría:

```
Command: (STRCAT cad1 cad2)
"Ejemplo de concatenación"
```

(SUBSTR<cad><prin> [<long>])SUBCADENA DE UNA CADENA.- Ente comando toma una parte determinada de una cadena

de texto de una posición indicada y con un número de caracteres dados. Ejemplo:

```
Command: (SUBSTR "Buenos días" 2 4)
"ueno"
```

(STRLEN<cad>) LONGITUD DE CADENA.- Este comando devuelve la longitud de la cadena que se indique. El valor de la longitud es el número de caracteres totales de la cadena de texto. Ejemplo:

```
Command: (STRLEN "Buenos días")
11
```

(ANGTOS <ang>[<modo> [<prec>]]) CONVERSIÓN DE ANGULOS.- Este comando toma el ángulo indicado, que debe ser un número real en radianes, y lo devuelve editado como cadena de texto según el formato especificado y la precisión que se indique. El argumento de <modo> es un número entero entre 0 y 4, cuyos formatos son los siguientes:

Modo 0	Grados
Modo 1	Grados/minutos/segundos
Modo 2	Grados centesimales g.
Modo 3	Radianes
Modo 4	Geodesia

TABLA 2.1 Modos de la función ANGTOS

Ejemplo:

```
Command: (ANGTOS PI 3 4)

"3.1416r"
```

(RTOS <num>[<modo> [<prec>]])CONVERSION DE NUMEROS.-

Este comando toma el número real especificado y devuelve una cadena según el formato especificado por el modo y la precisión. El argumento <modo> es un número entero entre 1 y 5, cuyos formatos son los siguientes:

Modo 1	Científico
Modo 2	Decimal
Modo 3	Pies-y-pulgadas I(fracción decimal)
Modo 4	Pies-y-pulgadas II(fracción propia)
Modo 5	Fracción en cualquier unidad

TABLA 2.2 Modos de la función RTOS

Ejemplo:

```
Command: (RTOS 24.75 2 1)

"24.7"
```

(ATOI <cad>)CONVERSION DE CADENA EN ENTERO.-

Este comando convierte la cadena indicada en un número entero. Si <cad> tiene decimales los trunca.

Ejemplo:

```
Command:(ATOI "38.7")
38
```

(ATOF <cad>)CONVERSION DE CADENA UN NUMERO REAL.- Este comando convierte la cadena indicada en un número real. Ejemplo:

```
Command:(ATOF "35.75")
```

(ITOA <entero>)CONVERSION DE ENTERO EN CADENA.- Convierte el valor entero indicado en una cadena de texto. Ejemplo:

```
Command:(ITOA -35)
"-35"
```

(FIX <num>)CONVERSION DE REAL EN ENTERO.- Este comando convierte el número real indicado en entero. Es decir, si es un número real con decimales, devuelve solo la parte entera. Ejemplo:

```
Command:(FIX 32.78)
32
```

(FLOAT <num>)CONVERSION EN REAL.- Este comando convierte el número que se indique en un número real. Ejemplo:

```
Command: (FLOAT 35)
```

.10 INSTRUCCIONES DE CAMBIO DE SISTEMA

AutoCAD dispone de una serie de variables denominadas Variables del Sistema que permiten efectuar variaciones en su forma de trabajo.

Todas las variables de Sistemas de AutoCAD pueden ser gestionadas desde los programas de AutoLISP, bien para extraer los valores actuales contenidos en ellas, bien para introducir nuevos valores (salvo las que sean de sólo lectura). La Tabla 2.3 presenta las variables más empleadas por AutoLISP.

El significado y función de cada una de las variable de Sistema viene determinado por el propio AutoCAD. No obstante, hay una variable que conviene reseñar en éste capítulo por su utilidad por los programas en AutoLISP que requieren designar puntos pertenecientes a entidades de dibujos. Esta Variable de Sistema es **OSMODE**[2] y su contenido es el de del modo de referencia a entidades, vigente actualmente en el dibujo. Cada vez que haya que introducir un punto en el programa de AutoLISP designado en

VARIABLE	DESCRIPCION
Aunits	Controla las unidades angulares: 0=grados decimal; 1=grados/minutos/segundos; 2=grados; 3=radianes
Auprec	Controla la precisión de las unidades angulares, determinadas por el numero de decimales.
Axismode	Controla la visualización de los ejes: 0=OFF; 1=ON
Blipmode	Controla la apariencia de las marcas auxiliares. 0=OFF; 1=ON
Cmdecho	Variable usada en AutoLISP; controla la visualización de mensajes. 0=OFF; 1=ON
Dragmode	Controla el arrastre de las imágenes transitorias cuando se desplaza el cursor. 0=OFF; 1=ON; 2=AUTO
Menuecho	Controla la visualización en la pantalla de los comandos y preguntas (indicadores) derivados de los menú. 1=Suprime la visualización de los comandos seleccionados desde el menú; 2=Suprime la visualización de los comandos y sus preguntas auxiliares cuando los comandos se invocan con macro en AutoLISP; 3= Es una combinación de las dos anteriores; 4=Desactiva el comando [Ctrl]+[P]
Osmode	Mantiene el valor para los modos OSNAP en uso.
Texteval	Controla si las preguntas o indicadores para textos y los atributos deben ser interpretados literalmente o como expresiones de AutoLISP: 0=Interpretación literal; 1=El texto introducido entre paréntesis o exclamaciones será interpretado como expresión de AutoLISP.

TABLA 2.3 Variables de AutoCAD mas usadas por AutoLISP[2]

pantalla, conviene tener muy en cuenta cuál es el valor actual de **OSMODE** y cambiarlo si fuera preciso. Los valores posibles de **OSMODE** y su significado son los siguientes:

VALORES	SIGNIFICADOS
0	NONE(ninguno)
1	END(Punto final)
2	MID(Punto medio)
4	CEN(Centro)
16	QUA(Cuadrante)
32	INT(Intersección)
64	INS(Punto de inserción)
128	PER(Perpendicular)
156	T(Tangente)
512	NEA(Cerca)
1024	QO(Rápido)

TABLA 2.4 Valores de OSMODE

(SETVAR <var> <valor>) INTRODUCIR VALOR EN VARIABLE.- Este comando asigna el valor especificado a la variable que se indique. El nombre de la variable debe ir entre comillas. Si la variable es de solo lectura, no se puede introducir ningún valor. Ejemplo:

```
Command: (SETVAR "filletrad" 2)
```

(GETVAR <var>)EXTRAER VALOR DE UNA VARIABLE.- Este comando extrae el valor actual de la variable indicada y devuelve ese valor. Ejemplo:

```
Command: (GETVAR "aperture")
```

(OSNAP <pt> <modos>)APLICAR MODOS DE REFERENCIA.- Este comando aplica el modo de referencia indicados en el punto que se indique, y devuelve un punto como resultado de esa aplicación. El modo o modos debe ser una cadena de texto y por lo tanto debe ir entre comillas. Si los modos indicados son varios, irán separados por comas. Ejemplo:

```
Command: (OSNAP p1 "cen,mid,end")
```

.11 ENTIDADES

Las Entidades son los elementos de que disponemos para crear los dibujos en pantalla tales como: LINEAS, CIRCULOS, ARCOS, TEXTO, etc.

Las polilineas y bloques también son tratados como Entidades pero es necesario EXPLOTARLAS -[**Explode**[2]]- previamente para que puedan ser tratadas como Entidades simples. De cada Entidad

AutoCAD guarda una gran cantidad de información en la Base de Datos.

Por ejemplo: de la Entidad línea son guardados los parámetros siguientes:

Nombre de la Entidad
Tipo de Entidad
Nombre de la capa
Coordenadas del punto inicial
Coordenadas del punto final
Nombre del tipo de línea
Número de color

TABLA 2.5 Base de datos de una linea

2.12 LAS ENTIDADES DE AutoCAD

A partir de la versión 2.6 AutoCAD incluye unas funciones especiales de AutoLISP con las que se pueden acceder y modificar las Entidades contenidas en la Base de Datos. Estas pueden facilitar el trabajo ya que reducen el tiempo de selección.

A cada entidad, en la Base de Datos le es automáticamente asignado un único nombre en el momento en el momento que es creada. El nombre de la Entidad es dado con ocho dígitos siendo éste un número en notación **HEXADECIMAL**[4]. La estructura interna de los nombres de las Entidades y sus asignaciones son de fácil manejo.

2.13 LISTAS ASOCIADAS

Cada Entidad lleva implícita una lista asociada, compuesta de los distintos datos que son necesarios para que esa Entidad pueda realizarse.

Dentro de éstas listas asociadas existen otras denominaciones denominadas **SUBLISTAS**, que contienen a su vez, los parámetros necesarios para completar la Entidad.

El primero de estos parámetros en una **SUBLISTA** es un número llamado **CODIGO DE GRUPO**, que identifica una propiedad particular en la Entidad. Por ejemplo, la primera sublista asociada a la lista de una Entidad es su propio nombre. El nombre de la entidad tiene como **CODIGO DE GRUPO -1**, seguido por el nombre de la **Entidad 600000014** (en nuestro caso), quedando la sublista del nombre de Entidad de la siguiente forma:

(-1.<nombre de entidad:600000014>)

El código de grupo sirve para identificar una Entidad en particular. Algunos Grupos de Códigos son comunes a todas las entidades del dibujo en una lista asociada y son los siguientes:

CODIGO	SIGNIFICADO
0	Tipo de entidad (Line,Circle,Point,etc)
2	Nombre del bloque (obtenida con INSERT)
6	Nombre del tipo de línea
7	Nombre del estilo de texto
8	Nombre de la capa
38	Valor de elevación
39	Valor de altura de objeto
62	Numero de color (0=porbloque, 256=porcapa)
66	Señal de que el bloque contiene atributos

TABLA 2.6 Código de las entidades principales de dibujo

La mayoría de las sublistas contienen sólo dos bloques de información: el Código de Grupo y la Especificación de la propiedad de información para el dibujo. Por ejemplo (0."CIRCLE"). Cuando una sublista contiene sólo dos bloques, estos van separados por un punto recibiendo el nombre de **PAR_PUNTEADO**, ocupando menos memoria que las listas comunes.

Algunas sublistas contienen información de las coordenadas de situación y pueden incluir tres o cuatro bloques de información: el Código de Grupo, la coordenada X, la coordenada Y, y en ocasionalmente la coordenada Z. Un ejemplo completo de una asignación sería:

```
( (-1.<Entity name: 60000108>)
  (0 . "INSERT")            Tipo de entidad "INSERT"
  (8 . "ELEMENT")           La capa se llama "ELEMENT"
  (66 . 1)                  El bloque contiene atributo
  (2 . "RES")               Nombre del bloque "RES"
  (10 80.0 100.0 0.0)       Coordenadas del punto de
                            inserción.
  (41 . 1.0)                Escala X del bloque
  (42 . 1.0)                Escala Y del bloque
  (43 . 1.0)                Escala Z del bloque
  (50 . 0.0)                Angulo de rotación
  (70 . 0)                  Número de columnas MINSERT
  (71 . 0)                  Número de filas MINSERT
  (44 . 0.0)                Espacio entre columnas
  (45 . 0.0)                Espacio entre filas
  (210 0.0 0.0 1.0)         Orienta la altura del objeto
)
```

TABLA 2.7 Base de Dato de un Bloque

Como se puede comprobar, cada apartado contiene dos partes. La primera es el **NUMERO DE CODIGO** (correspondiente a la clase de parámetro) y la segunda tiene dos funciones: una es determinar el **PARAMETRO** y la otra es un **NUMERO REAL.**

2.14 FUNCIONES DE ACCESO A ENTIDADES

Desde que asociación de listas es estructuradas como una lista de datos encerrada entre paréntesis, ésta puede ser usada y modificada por las funciones de AutoLISP. Algunas de éstas han sido predefinidas para actuar exclusivamente sobre asociaciones de listas.

La recuperación y actuación sobre asociaciones de entidades de listas se realiza en un proceso que conlleva en tres pasos:

1.) El nombre de la entidad es recuperado desde la Base de Datos del Dibujo.

2.) Las sublistas asociadas al nombre de la entidad son recuperadas y modificadas para referenciar al Código de Grupo apropiado.

3.) Las listas son actualizadas en el Dibujo principal incluyendo los cambios de entidades.

2.15 FUNCIONES DE TRABAJO CON ENTIDADES DE DIBUJO

Estas funciones proporcionan una gran variedad de usos para la recuperación de datos de una entidad y para el análisis del dibujo actual, permitiendo seleccionar una entidad desde la pantalla mientras se esté en el Editor de Dibujo.

(ENTNEXT [<nom-ent>])NOMBRE DE ENTIDAD SIGUIENTE.- Este comando devuelve el nombre de la primera entidad de la Base de Datos que sigue a la entidad cuyo nombre se indica. Si no se especifica ningún nombre, devuelve el de la primera entidad no eliminada de la Base de Datos. Ejemplo:

```
Command: (ENTNEXT ent1)
```

(ENTLAST)NOMBRE DE LA ULTIMA ENTIDAD PRINCIPAL.- Este comando devuelve el nombre de la última entidad principal no eliminada de la Base de Datos. Se utiliza fundamentalmente para obtener el nombre de las entidades recién añadidas a la Base de Datos con "COMMAND". Presenta la gran ventaja de que obtiene el nombre de la entidad aunque no sea visible en

pantalla o se encuentra en pantalla o se encuentre en una capa inutilizada. Ejemplo:

```
Command: (ENTLAST)
```

(ENTSEL<mens>) NOMBRE DE ENTIDAD DESIG. CON PUNTO.- Existen determinadas órdenes de AutoCAD que procesan entidades teniendo en cuenta el punto de designación de las mismas. El comando **"ENTSEL"** espera que el usuario designe una única entidad mediante un punto y devuelve una lista cuyo primer elemento es el nombre de la entidad, y el segundo elemento el punto de designación. Ejemplo:

```
Command: (ENTSEL "Designe una entidad")
(<Entity name: 6000032A> (10.0 20.0 0.0))
```

2.16 FUNCIONES PARA TRABAJAR CON LISTAS ASOCIADAS

Después de que los nombres han sido devueltos, las funciones que continuación presentaremos, actúan sobre ellas para extraer, borrar o hacer cambios en las listas asociadas.

(ENTGET <nombre-ent>) INTRODUCIR LISTA DE ENTIDADES.- Este comando busca en la Base de Datos el nombre de

la entidad indicada y devuelve la lista completa correspondiente a esa entidad. Ejemplo:

```
Command: (ENTGET nom)
```

(ENTDEL <nombre-ent>)BORRA O RECUPERA ENTIDAD.- La entidad cuyo nombre se indica es borrada si existe en la Base de Datos del dibujo actual, o recuperada si había sido previamente borrada en la actual sesión del dibujo. Ejemplo:

```
Command: (ENTDEL nom)
```

(ENTMOD <lista-ent>)ACTUALIZAR LISTA DE ENTIDAD.- Este comando actualiza la lista de una entidad en la Base de Datos. Recibe la lista de la entidad y la inserta en la Base de Datos sustituyendo a la que existía. Ejemplo:

```
Command: (ENTMOD entac)
```

2.17 FUNCIONES PARA HACER CAMBIOS EN LAS SUBLISTAS

Las siguientes funciones buscan el parámetro especificado para recobrar y poder modificar la información de las sublistas.

(ASSOC <elem> <list>)LISTA ASOCIADA.- Este comando busca el elemento especificado en la lista y devuelve la lista asociada cuyo primer elemento es <elem>. Por eso la lista debe ser una lista de asociaciones. Por ejemplo si se tiene una lista que contenga tres lista de asociaciones y además almacenada en una variable lis podrida ser:

```
( (cuerpo poliedro) (largo 25) (alto 12) )
Command: (ASSOC 'cuerpo lis)
(cuerpo poliedro)
```

(CONS <pr-elem> <list>)AÑADIR PRIMER ELEMENTO A LISTA.- Este comando toma la lista indicada y le añade un nuevo primer elemento, devolviendo la nueva lista resultante. Si como segundo elemento se especifica no una lista si no un valor concreto o de una variable, entonces **CONS** construye un tipo de lista de dos elementos llamado **PAR PUNTEADO** puesto que tiene un punto como separación entre ambos elementos. Ejemplo:

```
Command: (CONS 'tipo 32)
(TIPO . 32)
```

(SUBST<nue-elem><ant-elem><lis>) SUSTITUIR ELEMENTO.- Este comando busca en la lista indicada el elemento actual y lo sustituye por el nuevo elemento, devolviendo la nueva lista resultante. Si el elemento actual no se encuentra, devuelve la lista tal como está, sin cambios. Ejemplo:

```
Command: (SUBST '(8 . "Pieza") '(8 . 0) lis)
```

2.18 FUNCIONES DE SELECCION DE ENTIDADES

Estas funciones seleccionan un grupo de nombres de entidades. Una vez dado el nombre del grupo las entidades pueden actuar como un conjunto único. Este procedimiento es similar al mecanismo de ventana para Selección de Objetos de AutoCAD. Las siguientes funciones trabajan específicamente con éstas selecciones.

(SSGET [<modo>][<pt1>][<pt2>]) INTRODUCIR CONJUNTO.- Este comando acepta un conjunto designado de entidades. Las posibilidades de utilización son las siguientes:

1.) SSGET.- Solicita del usuario una designación completa. Se puede emplear todos los modos de

designación: Windows, Crossing, Previus, etc., y en el momento en que se pulse RETURN, el conjunto designado es aceptado por SSGET. Ejemplo:

```
Command: (SSGET)
Select objects:
```

2.) **SSGET "P".-** Designa el último conjunto previamente designado. Ejemplo:

```
Command: (SSGET "P")
Select objects
```

3.) **SSGET p1.-** Designa la entidad que pasa por el punto "p1". Es equivalente a señalar ese punto en pantalla. El resultado dependerá del modo de referencia actual, es decir, el valor de la variable **OSMODE**. Ejemplo:

```
Command: (SSGET '(3 4))
```

4.) **SSGET "C" p1 p2.-** Designa el conjunto resultante de la captura cuyos vértices son los puntos p1 y p2. Ambos puntos hay que indicarlos. Ejemplo:

```
Command: (SSGET "C" '(1 3) '(5 8))
```

5.) SSGET "W" p1 p2.- Designa el conjunto resultante de la captura cuyos vértices son los puntos p1 y p2. Esos puntos hay que indicarlos. Ejemplo:

```
Command:(SSGET "W" '(1 3) '(5 8))
```

(SSNAME <conj> <ind>)NOMBRE DE ENTIDAD EN CONJUNTO.- El comando "SSNAME" devuelve el nombre el nombre de entidad del conjunto designado, situada en el lugar indicado por el índice. Las entidades del conjunto designado se empiezan a enumerar desde 0. Así un conjunto de cuatro entidades estarían numeradas de 0 a 3. Ejemplo:

```
Command:(SSNAME conj 1)
<Entity name: 60000A3C>
```

(SSLENGTH<conj>)NUMERO DE ENTIDADES EN UN CONJUNTO.- Este comando determina el número de entidades contenidas en el conjunto designado. El número de entidades es siempre un entero positivo salvo si es mayor que 32767, en cuyo caso es un número real. Ejemplo

```
Command:(SSLENGTH conj)
```

2.19 FUNCIONES ESPECIALES

Durante toda la tesis, se han explicado instrucciones o funciones que trabajan de una forma directa, en los programas, pero AutoLISP dispone de un grupo que hace que dicho programas no presenten problemas en su momento de ejecución. Estas instrucciones se explican a continuación.

(VMON) PAGINACION VIRTUAL DE FUNCIONES.- Desde la versión 2.5 AutoCAD incorpora ésta instrucción que tiene por objeto, activar una página en memoria virtual, con el fin de poder almacenar en ella funciones especificas de AutoLISP.
Cuando "VMON" es llamada, todas las funciones que se encuentran a continuación serán introducidas en la citada página.

(CADDR <lista>) TERCER ELEMENTO DE LISTA.- En capítulos anteriores hemos visto como "CAR" toma la coordenada X de un punto, y "CADR" toma la Y, el comando "CADDR" introduce la coordenada Z en los dibujos 3D. Ejemplo:

```
Command: (CADDR '(4 5 7))
7
```

(INITGET [<bits>][<cad>])MODOS DE LA FUNCION GET.- Este comando especifica el modo en que va a operar el siguiente comando de tipo "**GET...**" El comando devuelve siempre **nil**. El valor <bits> es un numero entero, cuyos valores posibles son los siguiente:

BITS	MODO
1	No admite valores nulos
2	No admite valores cero
4	No admite valores negativos
8	No verifica limites
16	Devuelve punto 3D en vez de 2D

TABLA 2.8 Modos de la función INITGET

(GETKWORD [<mens>]) PERMITIR OPCIONES O PALABRAS CLAVE.- Este comando permite solicitar del usuario que introduzca una serie de opciones o palabras clave. Ejemplo:

```
Command: (GETKWORD "Continuar Si/No")
```

(PAUSE) PAUSA.- Si se llama a una orden de AutoCAD y se especifica como argumento de **COMMAND** el símbolo predefinido **PAUSE**, la orden se interrumpe hasta que

el usuario introduzca los datos que en ese momento se requieran. Ejemplo:.

```
Command: (COMMAND "line" p1 pause)
```

(VER)VERSION DE AutoLISP.- Este es un comando sin argumento, simplemente informativo. Devuelve una cadena de texto que contiene el número de versión de AutoLISP cargado.

CAPITULO III

DISEÑO GRAFICO DE RESORTES HELICOIDALES

3.1 INTRODUCCIÓN

El programa desarrollado en éste capítulo crea una nueva orden en AutoCAD que permite el gráfico de resortes de tensión y compresión, basándose en normas y estándares definidos.

Una de las características del programa es la interface que se logra con el sofware RESORTES, el cual es un programa de diseño de resortes helicoidales. El programa "RESORTES" fue desarrollado en la FIM-ESPOL[6], éste permite realizar un análisis de esfuerzos tanto estático como dinámico de resortes helicoidales teniendo como característica guardar los resultados en un archivo de texto.

Este archivo de texto es abierto en forma de lectura, por el programa **MUELLE.LSP.** (Desarrollado en esta tesis), tomando todos los datos necesarios para la graficación del resorte.

Además, el programa brinda la opción de introducir estos parámetros por medio de una caja de diálogo o si el usuario lo prefiere desde la línea de orden de AutoCAD.

El programa presenta la opción de poder imprimir una tabla, la cual contiene los parámetros del resorte. Esta tabla de datos es graficada en el editor de dibujo como una entidad más de AutoCAD.

Finalmente el programa tiene la característica para poder "recordar" las respuestas anteriores a las peticiones del usuario y ofrecerlas como opciones para el próximo cálculo. Esto es de gran ayuda cuando el usuario está dibujando más de un objeto y alguno de los parámetros permanecen sin cambio.

2 CONSIDERACIONES DE DISEÑO.

Para el diseño gráfico del resorte es necesario desarrollar un archivo, el cual contiene los parámetros de diseño. Estos parámetros son resultados de un análisis completo de diseño del programa **RESORTES**.

El programa RESORTE se lo realizó en TURBO C++[7], el cual es un lenguaje de nivel medio y estructurado. El programa RESORTE permite el diseño de resortes helicoidales de tensión, compresión y torsión de alambre de sección circular. Además en su análisis considera la forma de los extremos de los resortes, así como los parámetros físicos que deben considerarse como por ejemplo el número de espiras activas, las condiciones de carga dentro de las que el resorte trabajará, tales como el espacio físico, diámetro permitido y paso del resorte, si es que se va a trabajar en situaciones de cambio frecuentes de las fuerzas actuantes, así como el material que se puede utilizar. Si el resorte va a trabajar en situaciones de fatiga, el programa considerara la frecuencia natural para que el resorte no entre en resonancia, el acabado superficial del alambre. Adicionalmente a estos parámetros, para resortes helicoidales de compresión, el programa considera la estabilidad en base de sus dimensiones y de la forma en la que deberán ser sujetos los extremos.

Una vez que el programa RESORTE ha terminado de hacer todos los cálculos, los resultados pueden ser presentados en pantalla o si es que el resorte diseñado será graficado, dichos resultados son

almacenados en un archivo de texto, con extensión .RES (TABLA 3.1).

TIPO	5
Di	1.50
Do	12.00
P	1.5
Na	10.00
Ra	10
MATERIAL	ASTM A228-51

TABLA 3.1 Archivo generado por RESORTES

La Tabla 3.1 muestra el archivo de resultados generado por el programa RESORTES, a continuación se explica su contenido:

- TIPO .- Nos indica el tipo de resorte que fue seleccionado por el usuario. En la Tabla 3.1 se diseñó un resorte TIPO 5, lo cual nos indica que se trata de un resorte de tensión.

TIPO	EXTREMOS
1	Simples
2	Simple rebajado
3	A escuadras
4	Rebajados a escuadras
5	Gancho

TABLA 3..2 Extremos de los RESORTES

- Di.- Indica el valor diámetro del alambre.
- Do.- Indica el valor diámetro del resorte.
- P.- Indica el valor del paso del resorte.
- Na.- Indica el valor del número de espiras activa.
- Ra.- Indica el valor del radio del gancho.
- MATERIAL.- Indica el material utilizado en el diseño del resorte. La Tabla 3.3 contiene los principales materiales para la fabricación de resortes.

MATERIAL	ESPECIFICACIONES
Alambre Musical	ASTM A228-51
Alambre Revenido en Aceite	ASTM 229-41
Alambre Estirado Duro	ASTM A227-47
Alambre Cromo-Vanadio	ASTM 231-41
Alambre Acero Inoxidable	

TABLA 3..3.Materiales para Resortes[1]

3.3 EXPLICACION DEL PROGRAMA.

3.3.1 FUNCIONES PRINCIPALES

Ocho son las funciones que hacen posible la graficación de los Resortes en la rutina

MUELLE.LSP, el programa MUELLE.LSP en su totalidad se muestra en el Listado 3.1 del Apéndice A. Las funciones de la rutina se las detallan a continuación:

FUNCION INIC_MUELLE.- Esta función es la que permite la interacción entre el usuario y el interprete AutoLISP, permitiendo que el usuario introduzca los valores geométricos para la graficación del resorte, al mismo tiempo que va chequeando el ingreso de los datos, si encuentra que el usuario ha introducido algún dato que no cumple con las condiciones establecidas en el programa, entonces AutoCAD presentara en el prompt un mensaje de aviso que alguna condición no se cumple o que se ha introducido un dato erróneo, y volverá a pedir que ingrese el dato correctamente.

FUNCION PARA_FILE.- Esta función es llamada en el momento en que el usuario seleccione en la caja de diálogo la opción de ingresar datos por archivo, en ese instante se tendrá que ingresar el nombre del archivo que contenga los datos geométricos del resorte que previamente fue

diseñado por un programa **RESORTES**, desarrollado en la **FIM-ESPOL.**

A continuación el interprete AutoLISP pedirá que seleccione un punto, éste punto puede ser ingresado por medio del **RATON** o por medio del **TECLADO**, almacenándose su valor en la variable **pin**, a continuación se abrirá el archivo seleccionado en el modo de solo lectura y dicho descriptor se almacena en la variable **fich**, seguido se iniciara un lazo en el cual se extrae la información almacenada en el archivo, y cuyos valores serán almacenados en la variables **p1 p2....p6**, una vez obtenida toda la información se procede a cerrar el descriptor de archivo.

FUNCION INT.- Esta función desempeña su papel dentro de la función "**PROG**", lo que hace es calcular los puntos **p5, p6,...,p8,i1,i2,** tal como se muestra en la figura 3.1.

FUNCION ACT.- Esta función tiene por objeto actualizar las variables para poder dibujar el nuevo tramo del resorte. Es llamada de la

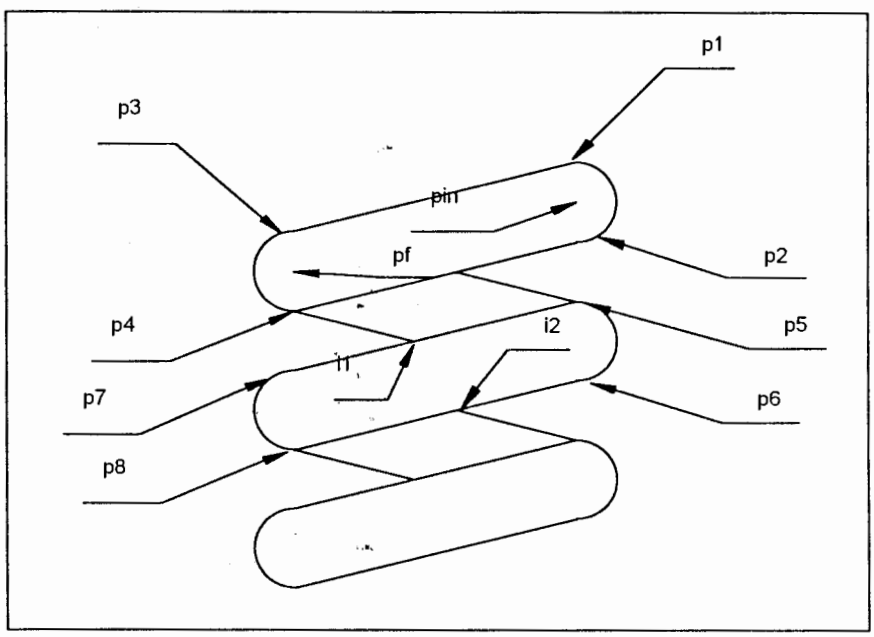


FIGURA 3. 1 Puntos dibujados por (INT)

diseñado por un programa **RESORTES**, desarrollado en la **FIM-ESPOL**.

A continuación el interprete AutoLISP pedirá que seleccione un punto, éste punto puede ser ingresado por medio del **RATON** o por medio del **TECLADO**, almacenándose su valor en la variable **pin**, a continuación se abrirá el archivo seleccionado en el modo de solo lectura y dicho descriptor se almacena en la variable **fich**, seguido se iniciara un lazo en el cual se extrae la información almacenada en el archivo, y cuyos valores serán almacenados en la variables **p1 p2....p6**, una vez obtenida toda la información se procede a cerrar el descriptor de archivo.

FUNCION INT.- Esta función desempeña su papel dentro de la función **"PROG"**, lo que hace es calcular los puntos **p5, p6,...,p8,i1,i2**, tal como se muestra en la figura 3.1.

FUNCION ACT.- Esta función tiene por objeto actualizar las variables para poder dibujar el nuevo tramo del resorte. Es llamada de la

función **"PROG"**. Su contenido es actualizar los puntos **pin pf P1...p4** para que sean tomados por los puntos anteriores al empezar a calcular el nuevo tramo.

Por último la función incrementa en 1 el valor de **n**, que es contador del número de espiras del resorte.

FUNCION PROG.- Su estructura básica es un ciclo repetitivo establecido mediante **"REPEAT[3]"**, con la variable **ns** como control. Esta función es llamada desde la función **"SIMP"**

Primeramente dibuja la mitad de la primera circunferencia teniendo como centro el punto pin y los puntos **p1 p2**, luego la función **"INT"** se encarga de calcular los puntos **p5, p6,...,p8, i1,i2**, a continuación se dibuja las rectas que unen los puntos **p1 p3, p2 p4**, y además se dibuja la mitad de la segunda circunferencia, (FIG. 3.1) seguido se puso un condicional "IF" el mismo que ejecuta la condición de verdadero hasta que la variable "n" iguale a la variable

ns, esto es para poder dibujar las líneas entre los puntos i1 p5, p4 i2.

FUNCION SIMP.- Esta función es la encargada de unir los diferentes extremos del resorte con el cuerpo del mismo, que es generado con la función **"PROG"**. El extremo del resorte a graficar depende de que tipo de muelle el usuario seleccione, ya que es diferente para cada caso, pero el cuerpo es el mismo. La selección del usuario de los diferentes tipos de resorte se guarda en una variable **op**, la misma que se compara con el condicional **IF**, en el momento en que la variable **op** encuentra igual a la condición de **IF**[3] se ejecutara la opción de verdadero, por lo cual el muelle será graficado. (fig. 3.2).

FUNCION CUADRO.- Esta función es la encargada de graficar en el editor de dibujo de AutoCAD el cuadro de datos el mismo que contiene los principales parámetros geométricos para su fabricación.

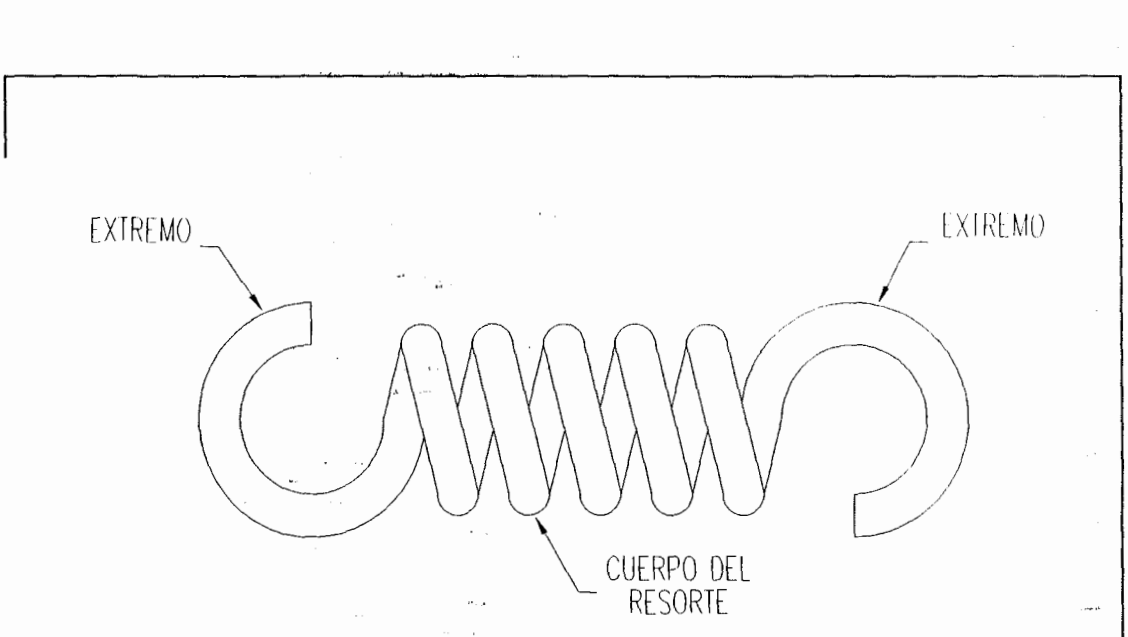


FIGURA 3. 2 Resorte de Tensión

La función pide al usuario que seleccione el punto de inserción del cuadro, luego calcula la escala que no es otra cosa que cuatro veces el diámetro del resorte, y presenta éste valor en pantalla. A continuación se inserta el archivo **TESIS.DWG** que contiene una serie de bloques predefinido, entre los cuales se encuentra el bloque **CUADRO2**, el mismo que es insertado en el editor de dibujo de AutoCAD. (FIG. 3.3)

El cuadro de datos se ubicara en la capa **FORMATO**, lo que le permite al usuario usar todas las características de la orden "**LAYER**[3]".

Los colores que fueron seleccionados son tales que no permite una confusión al ser impresos mediante un ploter.

3.3.2 FUNCIONES DEL CUADRO DE DIALOGO.

El Listado 3.2 del Apéndice A, es un archivo DCL que contiene las especificaciones para el cuadro de diálogo que se ve en la FIGURA 3.6.

DATOS DEL RESORTE	
MATERIAL	ASTM A220-51
DIAMETRO INTERIOR	0.20 mm
DIAMETRO EXTERIOR	0.70 mm
PASO	0.35 mm
NUMERO DE ESPIRAS	5.00
RADIO DEL GANCHO	0.70

FIGURA 3.3 Cuadro de Datos

La rutina de AutoLISP **DMUELLE.LSP** usa las funciones encontradas en **MUELLE.LSP** para dibujar los diferentes tipos de resorte, más funciones adicionales para manejar el cuadro de diálogo de entrada de datos, para convertir los datos según sea necesario, y verificar que se han introducido los datos correctos. El Listado 3.3 del Apéndice A, contiene la rutina completa.

Observe que, comparte funciones con **MUELLE.LSP**, necesita también que **MUELLE.LSP** se cargue antes de ejecutar **DMUELLE.LSP**. Varias funciones en **DMUELLE.LSP** merecen ser examinadas con más detalle.

FUNCION SETLOC_MUELLE.- Esta función está diseñada para inicializar las variables locales que contienen, o bien los valores por defecto de los parámetros de los resortes la primera vez que se ejecuta la rutina, o bien los valores globales en vigor del anterior proceso.

Observe que una serie de funciones **IF** comprueban la existencia de los valores globales, e

inicializan las variables locales de acuerdo con esto. Además, observe como los datos numéricos se convierten en datos de cadena, usando las funciones **RTOS**[3] e **ITOA**[3] antes de asignar los datos a las variables locales.

FUNCION OK_REAL.- La función "**OK_REAL**" es una función general de una gran ayuda para comprobar los datos de cadena antes de convertirlos en datos numéricos. Si la cadena no puede convertirse, se presenta un mensaje de error.

FUNCION VERIFY_MUELLE.- Esta función está diseñada para examinar los datos que se han introducido hasta que encuentra alguna forma de dato no válidos. Presenta un mensaje de error si encuentra datos no validos.

Una variable local "**OK_PARA**", se pone en nulo cuando se encuentra un error, no permitiendo de ésta manera que se salga del letrero de diálogo hasta que dicho error sea corregido. En el momento que la variable "**OK_PARA**" tome un valor

diferente de nulo, se sale del cuadro de diálogo y se ejecuta el programa graficando el resorte.

FUNCION GET_<NOMBRE>.- Todas las funciones que comienzan con **"(GET_"** están diseñadas para ser activadas cuando el usuario selecciona varias mosaicos de entrada de datos en el cuadro de diálogo, obteniendo el valor introducido por el usuario, y a la vez comprueba si dicho valor puede ser convertido por la función **"OK_REAL"**.

FUNCION C:RESORTES.- La función principal **"C:RESORTES"** define una nueva función de AutoCAD, que tras desactivar las marcas auxiliares y eco de ordenes activa el cuadro de diálogo y llama a las funciones anteriormente explicadas.

La figura 3.4 muestra el diagrama de flujo de las funciones del programa **DMUELLE.LSP**.

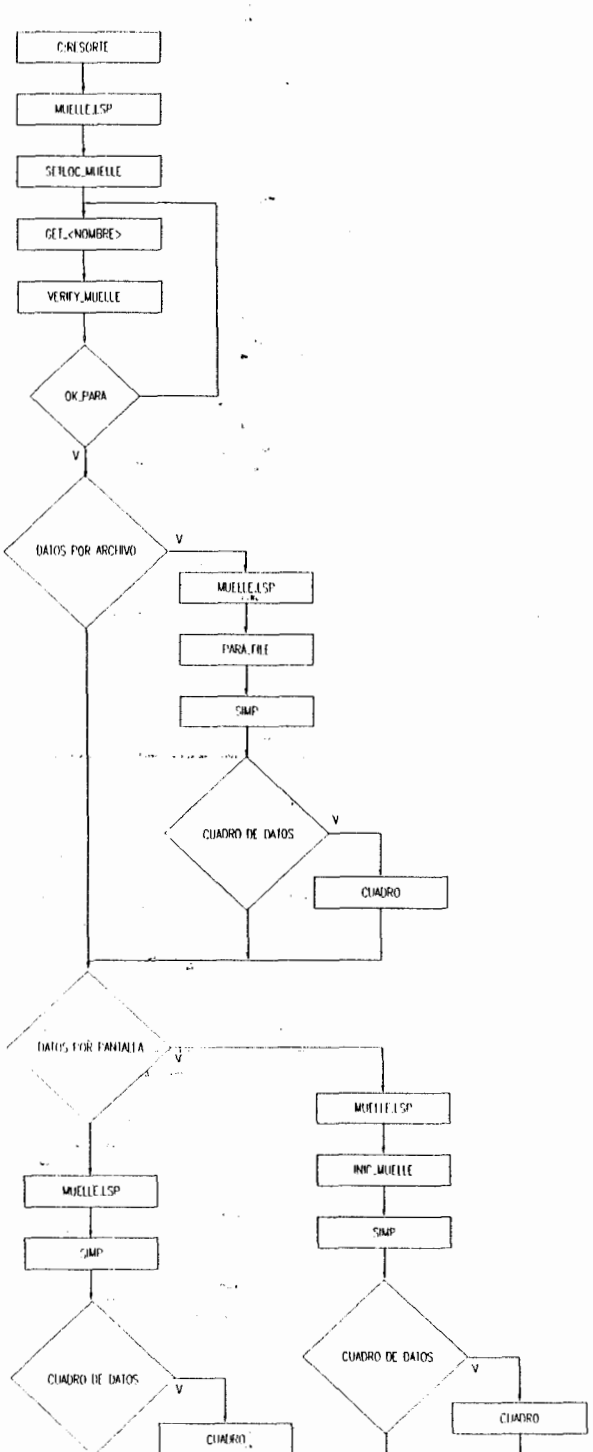


FIGURA 3.4 Diagrama de Flujo del Programa DMUELLE.LSP

EXPLICACION DEL USO DEL PROGRAMA

La secuencia de trabajo para ejecutar el graficador automático de rescrtes es:

Por el teclado, a partir del indicador [prompt] **Command, teclee: MENU,** Seleccione **TESIS.MNX,** y a continuación **OK.**

En el sistema de Menúes desplegables, seleccione:

DRAW/E-MAQUINA/RESORTES (FIG. 3.5).

La Orden **RESORTE** utiliza un letrero de diálogo para especificar todos los datos necesarios para poder graficar los diferentes tipos de resortes (FIG. 3.6).

atos por Pantalla:

1. Activada la casilla:

   El Punto de Inserción, los Tipos de Resortes, y los parámetros del resorte deberán ser indicados desde la línea de ordenes de AutoCAD.

2. Desactivada la casilla:

   2.1 Punto de Inserción:

   En ésta casilla de edición se introduce las coördenadas del punto de inicio del resorte, el cual es el extremo superior derecho.

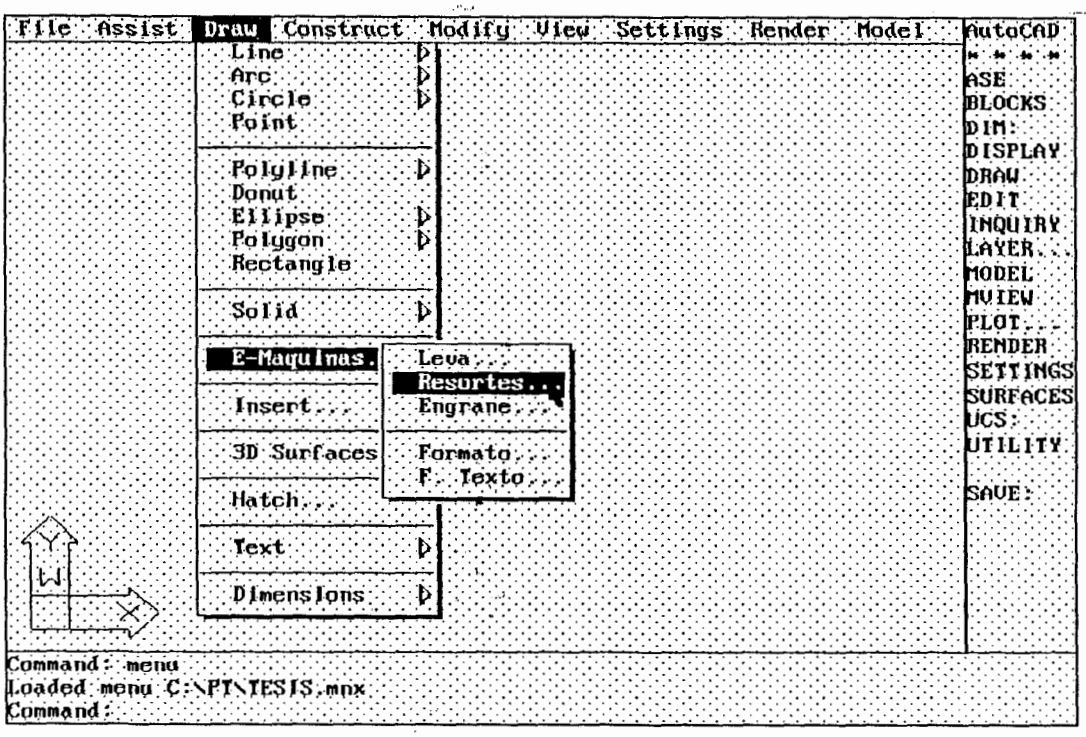


FIGURA 3.5 Menú desplegable de la Orden RESORTES

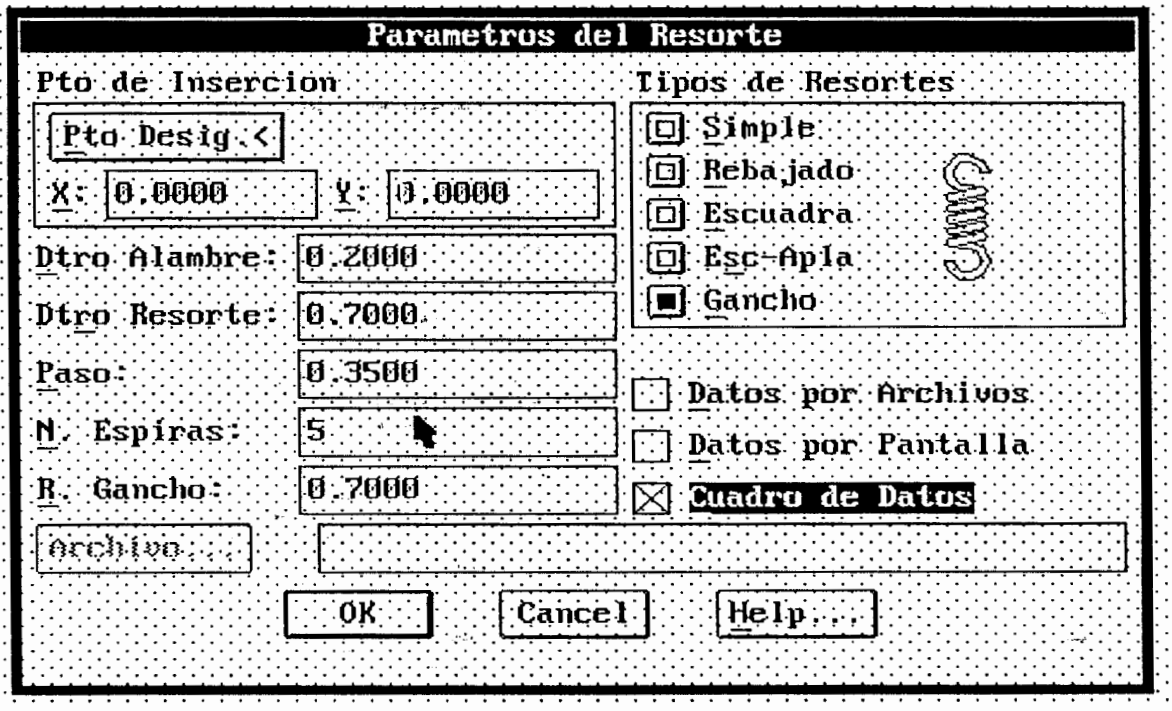


FIGURA 3.6 Orden RESORTES. Letrero de diálogo

2.2 Tipos de Resortes:

En éstas casillas de selección se elige el tipo de resorte a graficar, apareciendo la imagen del resorte seleccionado. Los tipos de resortes a seleccionar son:

- Resortes de compresión

  1. Resortes de extremos simples.
  2. Resortes de extremos rebajados simples.
  3. Resortes de extremos a escuadra
  4. Resortes de extremos rebajados a escuadra.

- RESOTE DE TENSION

  Resorte de extremo de gancho.

2.3 Casillas de edición:

Las casillas de edición correspondiente al Diámetro del Alambre, Diámetro del resorte, Paso, Número de Espiras, Radio

del Gancho se habilitan para su uso y modificación.

- Datos por Archivo.

Activada la casilla, Se puede acceder a la casilla "Archivo..." la cual accede al letrero de diálogo de gestión de archivo, para poder seleccionar los archivos que contienen los parámetros de los resortes emitidos por el Programa de Diseño **"RESORTES"**. El archivo a seleccionar debe poseer extensión **.RES** (FIG.3.7)

- Cuadro de Datos

Activa ésta casilla, permite la inserción del cuadro de corte con todos los datos del resorte, su Punto de inserción, y su Escala son indicados desde la línea de ordenes de AutoCAD.

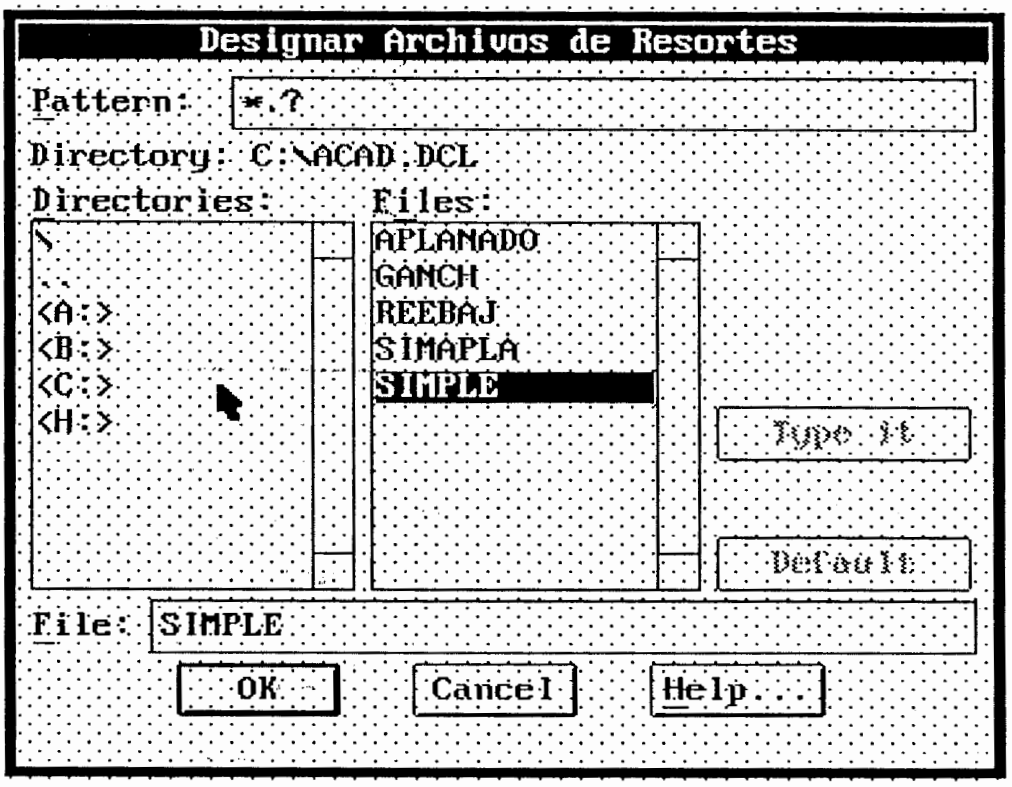


FIGURA 3.7 Submenú de la orden RESORTES

CAPITULO IV

DISEÑO GRAFICO DE LEVAS DE PLACA PLANA.

4.1 INTRODUCCION

El programa que se ha diseñado para éste capítulo crea una nueva orden en AutoCAD[2] que permite el gráfico de levas de placa plana.

Esta rutina está basada netamente en el desarrollo de un archivo de texto bajo el código **ASCII**[4]. Una de las características del programa es la interface lograda con el software LEVAS el cual es un programa de diseño de levas de placas planas realizado en **TURBO/C++**[7] y desarrollado en la FIM-ESPOL[8]. El programa LEVAS permite realizar un análisis cinemático, análisis dinámico, y análisis de la resistencia de los materiales del sistema; al final del cálculo el programa guarda los resultados en un archivo de texto con extensión **".LEV"**.

Este archivo de texto es abierto en forma de lectura, por el programa **LEVA.LSP**. El archivo contiene en su

interior una serie de puntos (máximo 360 puntos) que al unirlos dan el perfil de la leva.

Básicamente el programa consiste en buscar un archivo con extensión ".LEV" (por ejemplo LEVA.LEV), abrirlo y leer el contenido del mismo, para luego ir uniendo mediante líneas punto por punto hasta completar el perfil de la leva.

Además el programa brinda la opción de introducir parámetros por medio de una caja de diálogo o si el usuario lo prefiere desde la línea de orden de AutoCAD.

El programa también brinda la oportunidad de graficar el agujero para el eje con su respectiva muesca para la chaveta.

La rutina LEVA.LSP es el programa base para la graficación de la leva. El programa se muestra en su totalidad en el Listado 4.1 del Apéndice A. Esta rutina puede ser usada solo por la versión 12.

4.2 CONSIDERACIONES DE DISEÑO.

Para el gráfico de levas de placa plana es necesario desarrollar un archivo, el cual contiene los puntos necesarios para graficar el perfil de la leva. Estos puntos son resultados de un análisis completo de diseño del programa "LEVAS"

El programa "LEVAS" se lo realizó en el lenguaje **TURBO C++**, el cual es un lenguaje de nivel medio y estructurado. El programa **LEVAS** primeramente hace un análisis cinemático en el cual se determinan las curvas de desplazamiento, velocidad, aceleración y jalón. Esta última curva indica el golpeteo existente entre la leva y el seguidor.

Posteriormente, se realiza un análisis geométrico, para lo cual se deberá seleccionar un seguidor, el computador determina las coordenadas del perfil de la leva y si el usuario lo solicita, las coordenadas del eje de una fresa que se usa para maquinar el perfil de la leva. El programa en base del análisis cinemático, hace un análisis dinámico en el cual se

calcula las fuerzas normales entre las superficie de la leva y el seguidor, el torque que debe vencer la leva.

Por último el programa determina los esfuerzos máximos en la leva y en el seguidor, además comparando los valores de los esfuerzos máximos que pueden soportar el material con los esfuerzos máximos en la leva.

Una vez que el programa **LEVA** ha terminado de hacer todos los cálculos, las coordenadas del perfil de la leva pueden ser almacenados en un archivo de texto con el fin de cubrir la posibilidad de que puedan ser leídos por otros programas (TABLA 4.1).

x 122.996	y -0.953488
x 123.157	y 15.442
x 123.992	y 32.7851
x 125.747	y 51.686
x 127.681	y 72.7669
x 128.375	y 96.2771
x 126.201	y 121.773
x 119.754	y 148.127
x 108.175	y 173.786
x 91.2791	y 197.097
x 69.5099	y 216.611
x 43.7721	y 231.276

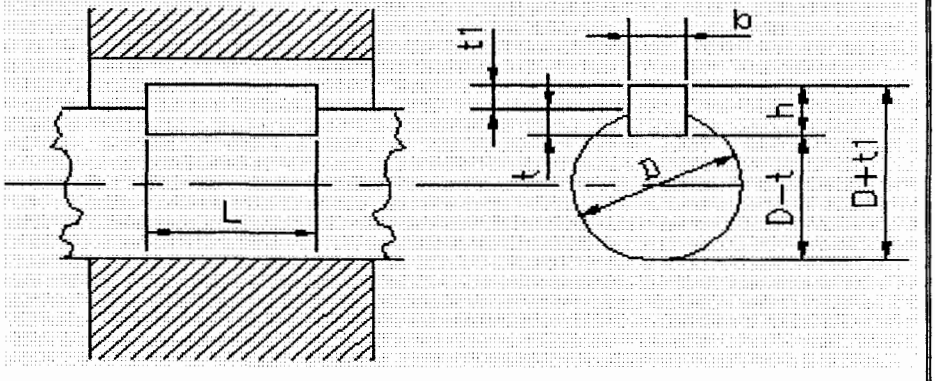
TABLA 4.1 Archivo generado por LEVAS

La Tabla 4.1 muestra el archivo de resultados del perfil de la leva (originalmente son 240 puntos) generado por el programa LEVA. El número de puntos que calcula el programa LEVA depende del número de puntos para los que se ha determinado el valor de las curvas características, con un máximo de 360 puntos.

El gráfico de la chaveta está basada en la selección de los tamaños estándares[9] de éstas, junto con el diámetro del eje. Las chavetas o cuñas se las usa para prevenir el movimiento relativo entre un eje y el elemento de contacto a través del cual se transmite un momento de torsión. Aún cuando los engranajes, las levas, etc. están montadas con un ajuste de interferencia, es aconsejable usar una cuña diseñada para transmitir el momento total. Existen normas ASME y ASA[12] para las dimensiones de la cuña y de las ranuras. En la Tabla 4.2. muestra las diferentes dimensiones de las chavetas estandarizadas, junto con el intervalo de diámetros de eje aplicables. La longitud de la chaveta se basa en la longitud del cubo y la carga torsional por trasmitir.

Chaveta Prismatica

Dimenciones de las secciones de las chavetas y chaveteros segun las Normas



Diámetro del árbol D	Dimensiones de la chaveta		Profundidad árbol	Profundidad manguito	longitud de la chaveta
	b	h	t	t1	
6 - 8	2	2	1.2	0.6	5 - 20
8 - 10	3	3	1.8	1.4	6 - 30
10 - 12	4	4	2.5	1.8	8 - 45
12 - 17	5	5	3	2.3	10 - 56
17 - 22	6	6	3.5	2.8	14 - 70
22 - 30	8	7	4	3.1	18 - 90
30 - 38	10	8	5	3.3	22 - 100
38 - 44	12	8	5	3.6	28 - 140
44 - 50	14	9	5.5	3.8	36 - 160
50 - 58	16	10	6	4.3	45 - 180
58 - 65	18	11	7	4.4	50 - 200
65 - 75	20	12	7.5	4.9	56 - 220
75 - 85	22	14	9	5.4	63 - 250

TABLA 4.2 Dimensiones en milímetros para chavetas prismáticas de tipo estándar[9]

4.3 EXPLICACION DEL PROGRAMA.

4.3.1 FUNCIONES PRINCIPALES.

El programa **LEVA.LSP** consta de tres funciones, las mismas que se detallan a continuación.

FUNCION INIC_LEVA.- Esta función solicita los datos necesarios que el usuario debe introducir para definir en éste caso la leva. En primer lugar solicita el punto de centro de la lleva, no permitiendo que el usuario introduzca un valor nulo "RETURN", dicho punto será almacenara en la variable **"pcen"**.

A continuación la función visualizara un mensaje para solicitar si se desea graficar el chavetero, si se respondiera afirmativamente el computador solicitara del usuario el diámetro del eje, el mismo que se almacena en la variable **"dia"**.

FUNCION LPTOS.- Esta función es la encargada de abrir el archivo con extensión **".LEV"** en modo de lectura para obtener los valores respectivos de

los diferentes puntos contenidos en él, estos puntos son almacenados en forma de cadena de texto en la variable **"cad"**.

El punto contenido en la variable **"cad"** es separado en sus componentes rectangulares x e y respectivamente para ser convertido en variables reales, para luego convertirlos en punto por medio del comando **"LIST[3]"**. Finalmente se llama a la orden de AutoCAD **"PLINE[3]"** para ir uniendo punto por punto hasta que la variable **"cad"** tome el valor de nulo.

FUNCION CHAVET.- Esta función gráfica exclusivamente el agujero del eje, y la muesca del chavetero, está basada en puro cálculo geométrico básico.

Básicamente la graficación del chavetero consiste en unir los puntos **pa, p1, p2, p3, p4,** mediante el comando de AutoCAD **"PLINE"**. Nótese que el quid está en encontrar el punto **"pa"** (FIGURA 4.1), ya que para ello es preciso

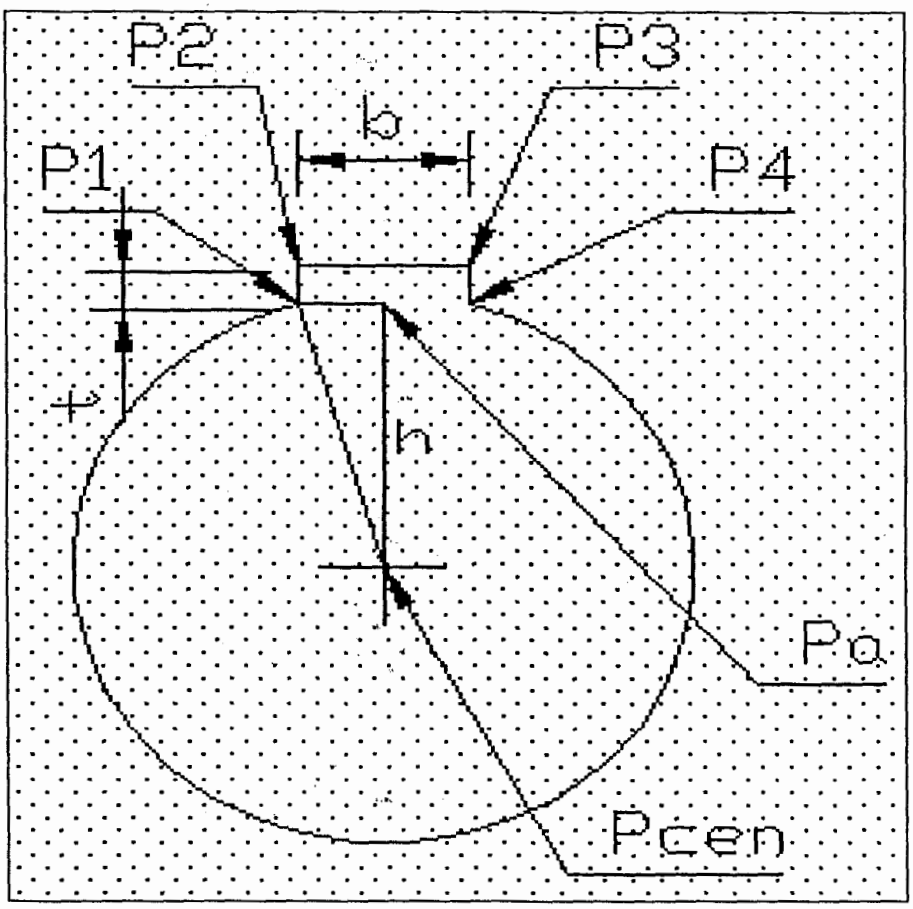


FIGURA 4.1 Agujero de eje

calcular la distancia entre **"pcen"** y **"pa"** mediante la formula:

$$h=\sqrt{(\frac{dia}{2})^2-(\frac{b}{2})^2}$$

3.3.2 FUNCIONES DEL CUADRO DE DIALOGO.

El Listado 4.2 del Apéndice A, es un archivo DCL que contiene las especificaciones para el cuadro de diálogo que se ve en la FIGURA 4.4. Este cuadro de diálogo es controlado por medio del programa **DLEVA.LSP**, además dicho programa llama a la rutina **LEVA.LSP**, para poder graficar la leva, el Listado 4.3 del Apéndice A, contiene la rutina completa. Las funciones contenidas en **DLEVA.LSP** se explican a continuación:

FUNCION SETLOC_LEVA.- Esta función, igual que en el capítulo anterior, está diseñada para inicializar las variables locales, o bien los valores por defecto de los parámetros de la leva.

FUNCION OK_REAL.- Esta función general se encuentra diseñada para comprobar los valores introducidos por el usuario, si uno de estos valores no se puede convertir en valores reales, la función devuelve un mensaje de error, advirtiendo al usuario de que algún valor se encuentra mal introducido.

FUNCION VERIFY_LEVA.- Esta función es la encargada de chequear las restricciones condicionales y los valores introducidos por el usuario, si encuentra un dato mal introducido, se presenta un mensaje de error, y a la vez pone a la variable local de prueba **"ok_para"** en su valor nulo, impidiendo de ésta manera que la caja de diálogo se desvanezca, al mismo tiempo el cursos se coloca en el mosaico del cual proviene el error. Una vez corregido el error la variable local **"ok_para"** toma el valor de verdad permitiendo de ésta manera que la caja de diálogo se desvanezca y se grafique la leva.

FUNCION GET_<NOMBRE>.- Todas las funciones que comienzan con **"(GET_"** están diseñadas para ser

activadas cuando el usuario selecciona varias mosaicos de entrada de datos en el cuadro de diálogo, obteniendo el valor introducido por el usuario, y a la vez comprueba si dicho valor puede ser convertido por la función **"OK_REAL"**.

FUNCION FILE_LEVA.- Esta función tiene por objeto de pedir por medio de un letrero de diálogo el nombre del archivo con su extensión **".LEV"**, almacenándolo en la variable **"fich"**. (FIGURA 4.5)

FUNCION C:DLEVA.- Esta función desactiva las marcas auxiliares y ecos de ordenes, y llama a todas las funciones anteriormente explicadas, las cuales hacen posible la graficación de la leva, además es aquí donde se activa el cuadro de diálogo que facilita el ingreso de los datos para el usuario.

La figura 4.2 muestra el diagrama de flujo de las funciones del programa **DLEVA.LSP.**

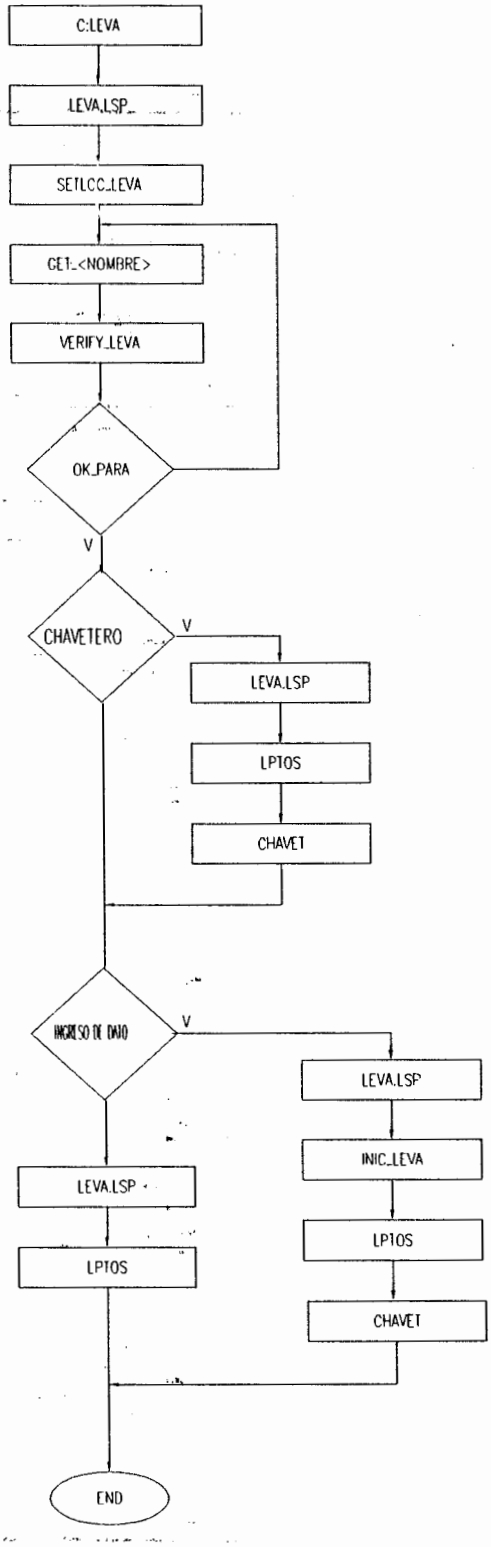


FIGURA 4.2 Diagrama de flujo de las funciones del programa DLEVA.

4.4 EXPLICACION DEL USO DEL PROGRAMA.

La secuencia de trabajo para ejecutar el graficador automático del programa resortes es:

Por el teclado, a partir del indicador [prompt] **Command, teclee: MENU,** Seleccione **TESIS.MNX,** y a continuación **OK.**

En el sistema de Menús desplegables, seleccione:

DRAW/E-MAQUINA/LEVA (FIGURA 4.3)

La Orden LEVA utiliza un letrero de diálogo para especificar todos los datos necesarios para poder graficar la leva de placa plana (FIGURA 4.4).

- **Seleccione el Nombre del Archivo.**

  Esta área se utiliza para especificar el nombre del archivo. La casilla **"Archivo..."** accede al letrero de diálogo de gestión de fichero para poder seleccionar cualquier archivo con extensión ".LEV" (FIGURA 4.5)

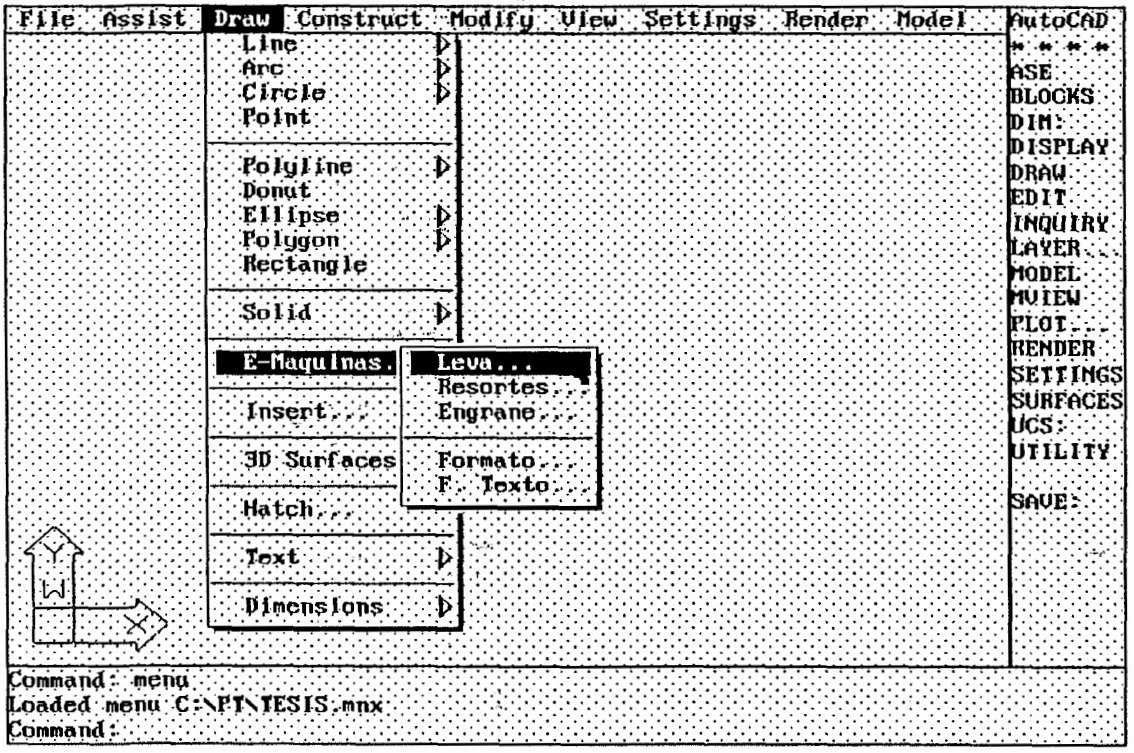


FIGURA 4.3 Menú desplegable de la Orden LEVA.

FIGURA 4. 4 Orden DLEVA. Letrero de diálogo

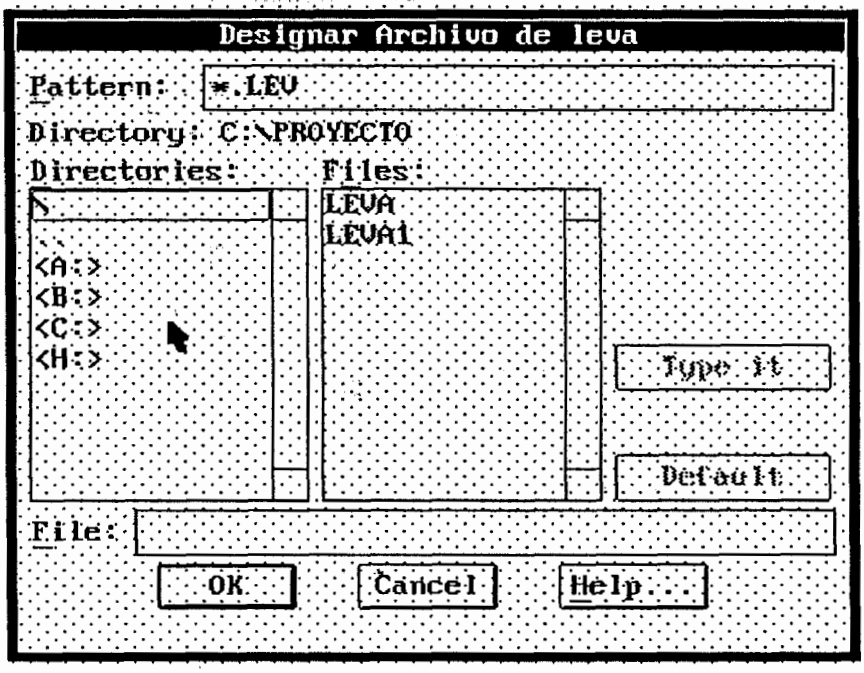


FIGURA 4. 5 Submenú de la orden DLEVA.

- **Opciones.**

Esta área sirve para aportar los datos necesarios para la graficación de la leva por dos maneras:

1. Ingreso de Datos:

   Activa ésta casilla, el Centro de Leva, y el Diámetro del Arbol deberán ser indicados desde la línea de ordenes de la misma forma que cualquier comando de AutoCAD.

2. Centro de Leva, Diámetro del Arbol:

   Desactiva la casilla anterior, las casillas de edición correspondientes al Centro de Leva, Diámetro del Arbol, se habilitan su uso y modificación.

CAPITULO V

DISEÑO PARAMETRICO DE ENGRANAJES DE DIENTES RECTOS

5.1 INTRODUCCIÓN

El programa desarrollado en éste capítulo crea una nueva orden en AutoCAD que permite el gráfico de engranajes de dientes rectos, basándose en el estudio de la geometría de la involuta, involumetría[10]. Todas las fórmulas y cálculos que se emplean en el programa están basados en las normas y estándares definidos.

Uno de los objetivos más importantes del programa es que fue desarrollado pensando en la inmediata maquinación del mismo, mediante los programas de control numérico, esto es posible gracias a un enlace que existe entre AutoCAD y la maquina de control numérico.

Este programa brinda la oportunidad de graficar el agujero para el eje con su respectiva muesca para la chaveta, además ofrece la opción de presentar

los resultados de los cálculos del diente del engranaje en pantalla o guardarlos en un archivo con extensión **".RPZ"** para su posterior revisión.

La rutina ENGRANE.LSP es el programa de soporte para la graficación del engranaje. El programa se muestra en su totalidad en el Listado 5.1 del Apéndice A. Esta rutina puede ser usada solo por la versión 12 de AutoCAD.

5.2 CONSIDERACIONES DE DISEÑO.

De las muchas curvas posibles que generan el perfil del dientes de los engranajes se ha escogido a la involuta, ya que éste presenta varias ventajas, entre las que se destacan su facilidad de generación del perfil y el hecho de que se está usando en todas las aplicaciones excepto en relojes de pulso y de pared en los que se emplean la cicloide.

Para el estudio del diseño gráfico del engranaje se ha considerado, que la envolvente comienza en la circunferencia base, ya que por definición de la involuta no se encuentra definida bajo ésta. Por consiguiente, al dibujar los dientes del engranaje se

acostumbra trazar como rectas radial el perfil situado abajo de la circunferencia base. Sin embargo la forma real dependerá de la clase de maquina herramienta empleada para fabricar los dientes.

Para el trazado de la envolvente se lo hará generando puntos para luego unirlos por líneas, ésta es la única manera de generar el perfil del diente del engranaje ya que AutoCAD no presenta ninguna curva capaz de graficar la envolvente. Esto es posible gracias al estudio de la geometría de la involuta (involumetría) la cual ha desarrollado la siguiente formula:

$$t = 2R[\frac{t_p}{2R_p} + inv\phi_p - inv\phi\ ] \quad (5.2.1)$$

Con la formula 5.2.1[10] es posible calcular el espesor del diente en cualquier punto de la involuta, a partir del espesor en cualquier otro punto. Se considerara como espesor conocido, al espesor del diente en el circulo de paso.

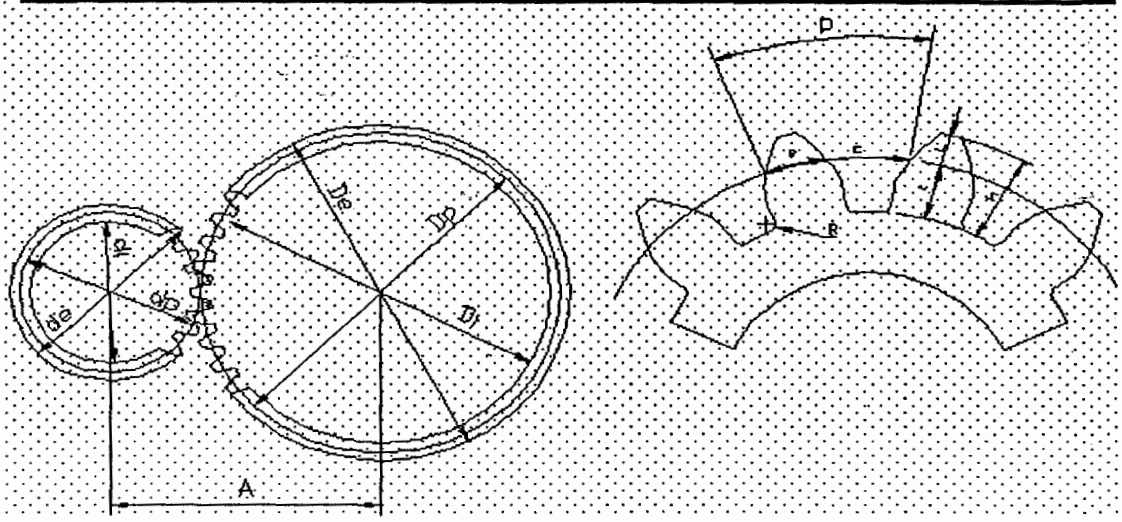
Para el gráfico de los engranajes rectos se usará el sistema normal del Módulo (M) el cual se define como la razón o relación del diámetro de paso al

número de dientes. La unidad de longitud que se utiliza habitualmente es el milímetro. El módulo es el índice de tamaño de los dientes en el Sistema Internacional (SI).

La ventaja de usar el sistema normal del módulo es que el Diámetro Primitivo y el Diámetro Exterior son siempre números enteros, a parte de que encontrar el módulo de un engranaje ya creado es simple, solo se mide el diámetro exterior y éste se lo divide para el número de dientes que tenga el engranaje, aumentando dos dientes. El modulo no solo sirve para hallar los diámetros del engranaje, sino que los dientes están relacionados con él, en otras palabras las diferentes partes del diente están en función del módulo.

FORMULAS GENERALES

para engranajes rectos segun el sistema normal de Modulo



DESIGNACION

P = Paso	h = Altura total del diente
M = Módulo	L = Adendo
De = Diámetro exterior	l = Dedendo
Di = Diámetro interior	R = Radio de entalle
c = espacio entre dientes	A = distancia entre ejes
e = Espesor del diente	N = número de dientes

$$M = \frac{P}{\pi} = \frac{D_p}{N} = \frac{D_e}{N+2}$$

$$P = M * \pi$$

$$D_p = M * N$$

$$D_e = M * (N+2)$$

$$D_i = D_p - (2M * 1{,}167)$$

$$c = \frac{P}{2} = M * 1{,}5708$$

$$e = \frac{P}{2} = M * 1{,}5708$$

$$h = M * 2{,}167$$

$$A = \frac{D_p + d_P}{2} = \frac{N+n}{2} * M$$

$$L = M$$

$$l = M * 1{,}167$$

$$R = 0{,}3 * M$$

TABLA 5.1 Fórmulas generales de los dientes de un engranaje[11]

5.3 EXPLICACION DEL PROGRAMA.

5.3.1 FUNCIONES PRINCIPALES

El programa **ENGRANE.LSP** está integrado por once funciones las mismas que hacen posible la graficación del engranaje, éstas funciones se detalla a continuación.

FUNCION RAD_DEG.- Es una función común que convierte los ángulos expresados en radianes a grados. Se la usa para presentarle al usuario la información del ángulo en grados, ya que AutoLISP trabaja solo en radianes.

FUNCION INIC_ENGRANE.- Igual que en los programas anteriores ésta función está diseñada para pedir al usuario los parámetros necesarios para poder graficar el engranaje, en éste caso pide que se ingrese el punto centro del engrane, el modulo, el número de dientes, el ángulo de presión, y por último se presenta la alternativa si desea graficar el chavetero, si la respuesta es afirmativa, el interprete AutoLISP pedirá que se ingrese el diámetro del eje, calculando

automáticamente el valor de la muesca de la chaveta.

FUNCION CAL.- Una vez ingresado los parámetros por el usuario, ésta función está preparada para calcular los valores necesarios para la graficación del engranaje, mediante una serie de fórmulas generales para engranajes rectos según el sistema normal de Módulo (Tabla 5.1).

FUNCION PUNTOS.- Esta función es la principal del programa **ENGRANE.LSP**, ya que es aquí donde se grafican los diferentes puntos que hacen posible el diseño gráfico del diente del engranaje. Para generar los puntos se utilizo la formula:

$$t = 2R[\frac{t_p}{2R_p} + inv\phi_p - inv\phi] \quad (5.3.1)$$

donde:

t = espesor del diente en cualquier punto.

t_p = espesor del diente en el circulo de paso.

R = radio a determinar el espesor del diente.

R_p = radio del circulo de paso.

φ = ángulo de presión en cualquier punto

φ_p= ángulo de presión en el radio de paso.

$inv\varphi = tan\varphi - \varphi$ función involuta.

El número de puntos por involuta generada es diez o sea un total de 20 puntos por diente dibujado. Cada punto se calcula de la siguiente manera:

De la ecuación 5.3.1 y remplazando

$inv\varphi = tan\varphi - \varphi$ se obtiene:

$$t = 2R[\frac{t_p}{2R_p} + (tan\phi_p - \varphi_p) - (tan\varphi - \varphi)] \quad (5.3.2)$$

Del gráfico 5.1, aplicando Pitagoras al triángulo BOG podemos calcular GB, por lo tanto:

$$GB = \sqrt{R^2 - R_b^2} \quad (5.3.3)$$

Pero GB también es igual a:

$$GB = R_b \, tan\varphi \quad (5.3.4)$$

Igualando (5.3.3) y (5.3.4) se tiene:

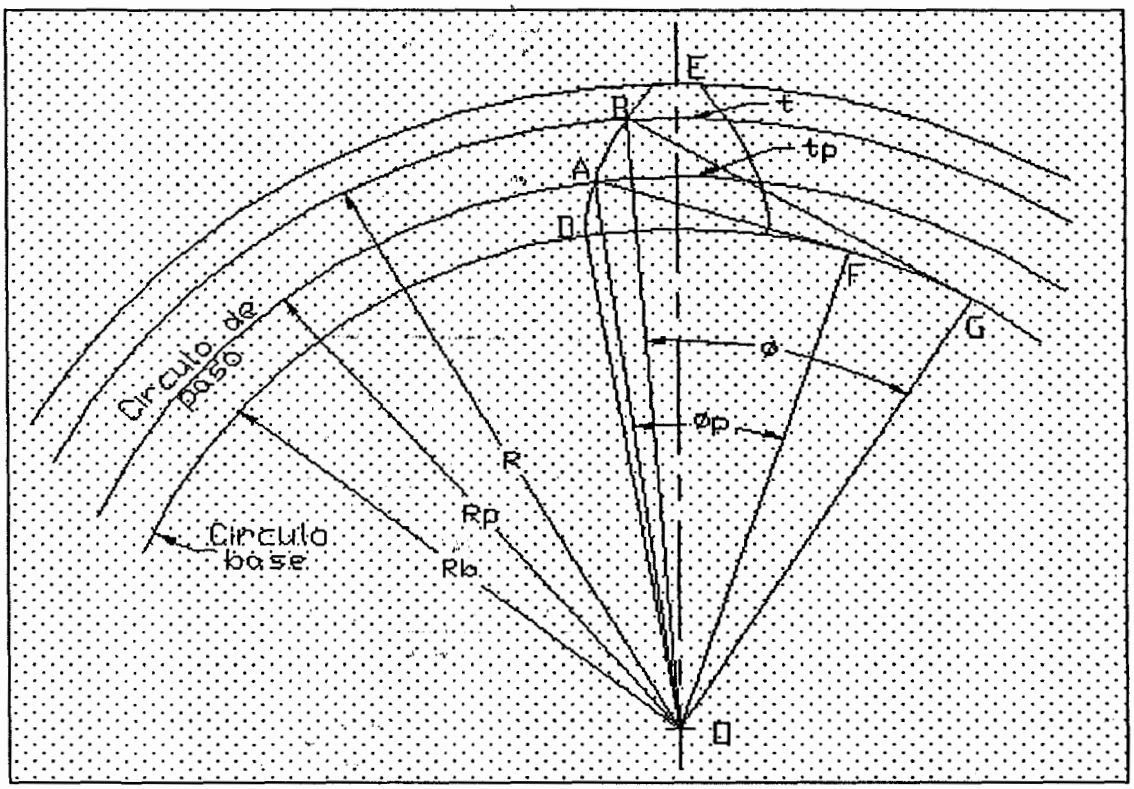


FIGURA 5.1 Parámetros principales del Engranaje

$$tan\varphi = \frac{\sqrt{R^2 - R_b^2}}{R_b} \quad , \quad tan\varphi_p = \frac{\sqrt{R_p^2 - R_b^2}}{R_b} \qquad \textbf{(5. 3.5)}$$

Remplazando 5.3.5 en 5.3.2 se obtiene:

$$t = 2R\left[\frac{t_p}{2R_p} + \left(\frac{\sqrt{R_p^2 - R_b}}{R_b} - tan^{-1}\left(\frac{\sqrt{R_p^2 - R_b}}{R_b}\right) - \frac{\sqrt{R^2 - R_b^2}}{R_b} + tan^{-1}\left(\frac{\sqrt{R^2 - R_b^2}}{R_b}\right)\right)\right.$$

(5.3.6)

Con la ecuación 5.3.6, podemos tener el espesor en cualquier punto del diente del engranaje, en función del espesor del diente en el circulo de paso, el diámetro de paso y el diámetro base, que son parámetros conocidos.

Para calcular los punto hacemos uso de la función **"POLAR[3]"**, para ello necesitamos ángulo comprendido entre la línea del centro del engranaje y la línea que se forma entre el centro y un punto cualquiera de la involuta. Este ángulo es igual a:

$$\frac{t}{2R} = \theta = \left[\frac{t_p}{2R_p} + \left(\frac{\sqrt{R_p^2 - R_b}}{R_b} - tan^{-1}\left(\frac{\sqrt{R_p^2 - R_b}}{R_b}\right) - \frac{\sqrt{R^2 - R_b^2}}{R_b} + tan^{-1}\left(\frac{\sqrt{R^2 - R_b^2}}{R_b}\right)\right)\right.$$

(5.3.7)

Además **"POLAR"** necesita la distancia entre el centro del engranaje y un punto cualquiera de la involuta. Para elle se divide al diente en diez partes iguales $(R_e - R_b]/10)$, sumando ésta fracción a la distancia inicial (radio base) se obtendrá el valor de la distancia para el nuevo punto.

La FIGURA 5.2 muestra todos los puntos calculados por la función **"PUNTOS"**.

FUNCION ENVOL.- Una vez creadas todas las variables de puntos con valores, hay que dibujar el engranaje. Para ello se llama con **"COMMAND[3]"** a la orden de AutoCAD y se suministran todos los puntos. Como éste número de variables, se establece en un ciclo repetitivo para formar la expresión de AutoLISP que contenga todos los puntos, al mismo tiempo se va almacenando todos los puntos como una cadena de caracteres en las variables **"expr1"** y **"expr2"**. Se inicializa la expresión **"expr1"** con la variable **"p2"** y la expresión **"expr2"** con el número de puntos generados menos dos, finalmente se forma una sola expresión **"exp"**

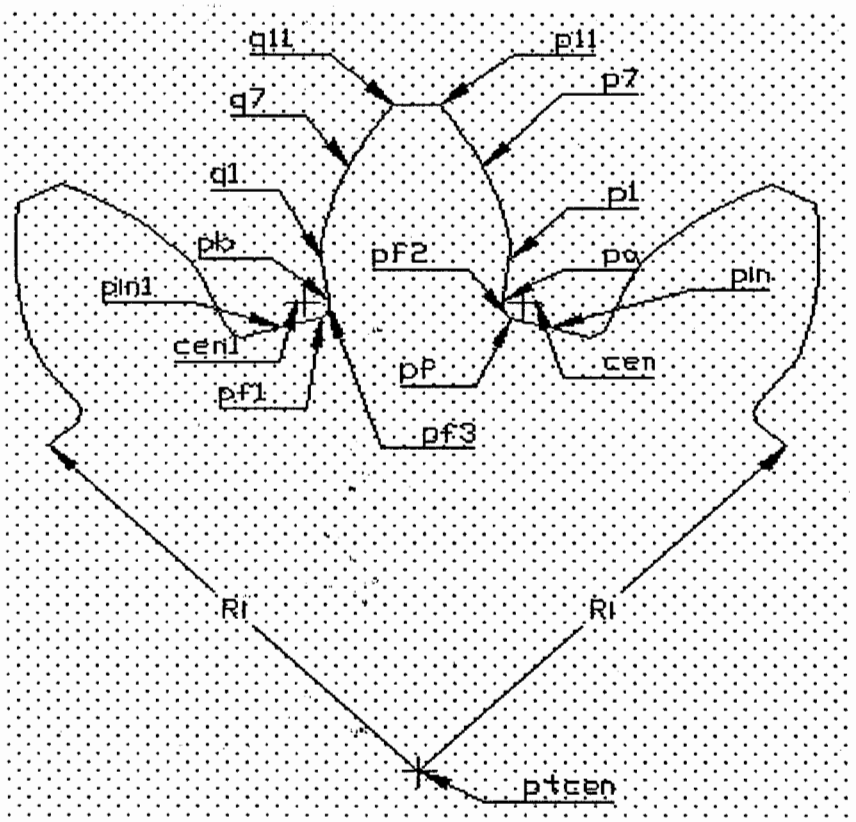


FIGURA 5. 2 Puntos generados por la función Puntos

que es una cadena de caracteres que contiene todos los puntos necesarios para la graficación de un diente del engranaje. Una vez formada la expresión solo falta evaluarla lo cual se hace en la función "**DIB**".

FUNCION DIB.- Consiste en un ciclo repetitivo que dibuja todos los dientes del engranaje. Previamente se inicializa a cero el ángulo de referencia "**angl**" para empezar a dibujar cada diente del engranaje.

El ciclo llama a las funciones "**CAL**", "**PUNTOS**", y "**ENVOL**" que fueron explicados anteriormente. Al final del ciclo se actualiza el ángulo de referencia "**angl**" sumándole cada ves el incremento del ángulo $\frac{2\pi}{nd}$.

FUNCION NULO.- El engranaje ya ha sido dibujado. Sin embargo, todas las variables creadas para cada diente permanecen cargadas en la memoria. Esta función tiene, pues, por objeto liberar todas esas variables de puntos poniendo su valor en nil.

FUNCION CHAVET.- Esta rutina está diseñada en calcular los puntos y graficar el agujero del eje y la muesca de la chaveta. Esta función es igual a la explicada en el CAPITULO IV.

FUNCION LIST_RESULTADO.- Esta función tiene por objeto de presentar en pantalla todos los resultados calculados por la función **"CAL"** o guardar dichos cálculos en un archivo con extensión **".RPZ"** para su posterior revisión.

La función comienza almacenando en la variable **"list1"** una lista con los diferentes nombres de los parámetros calculados.

En la variable **"list2"** almacena una lista con el signo "=", pero antes se emplea el carácter de control **"\t"** que es un tabulador, el cual permite un espacio entre el nombre del parámetro y el signo "=" (TABLA 5.2).

TABLA DE RESULTADOS DE LOS PARÁMETROS DEL ENGRANAJE

Modulo	=	12.00
Ángulo de Presión	=	25.00 grados
Número de Dientes	=	12.00 diente
Paso	=	37.70 mm
Diámetro Interior	=	115.99 mm
Diámetro Primitivo	=	144.00 mm
Diámetro Base	=	130.51 mm
Diámetro Exterior	=	168.00 mm
Espacio entre Diente	=	18.85 mm
Espesor del Diente	=	18.85 mm
Radio de Entalle	=	3.60 mm
Adendo	=	14.00 mm
Dedendo	=	12.00 mm
Altura del Diente	=	26.00 mm

TABLA 5.2 Tabla de resultados generado por le programa DENGRANE.LSP

En la variable "**list3**" se almacena una lista con los caracteres de control "\t", para poder dar un espacio de tabulación entre el signo de igual y los valores de los parámetros calculados.

En la variable "**list4**" se almacena una lista con los valores de los diferentes parámetros calculados con la función "**CAL**". Finalmente solo queda unir todas las listas, y esto se logra mediante la función de AutoLISP **MAPCAR**[3]". y presentarlo en pantalla de texto mediante la función de AutoLISP "**PRINC**[3]".

5.3.2 FUNCIONES DEL CUADRO DE DIALOGO

El Listado 5.2 del Apéndice A, es un archivo DCL que contiene las especificaciones para el cuadro de diálogo que se ve en la FIGURA 5.6.

La rutina AutoLISP **DENGRANE.LSP** usa las funciones encontradas en **ENGRANE.LSP** para graficar el engranaje, más funciones adicionales para manejar el cuadro de diálogo de entrada de datos, para convertir los datos según sea necesario, y verificar que se han introducido los datos correctos. El Listado 5.3 del Apéndice A, contiene la rutina completa.

Varias funciones en **DENGRANE.LSP** merecen ser examinadas con más detalle.

FUNCION SETLOC_ENGRANE.- Esta rutina está diseñada para inicializar las variables locales que contienen, o bien los valores por defecto de los parámetros del engranaje la primera vez que se ejecuta la rutina, o bien los valores globales en vigor del anterior proceso.

Obsérvese que una serie de funciones IF comprueban la existencia de los valores globales, e inicializan las variables locales de acuerdo con esto. Además, observe como los datos numéricos se convierten en datos de cadena, usando las funciones **RTOS** e **ITOA** antes de asignar los datos a las variables locales.

FUNCION OK_REAL.- La función **OK_REAL** es una función general de una gran ayuda para comprobar los datos de cadena antes de convertirlos en datos numéricos. Si la cadena no puede convertirse, se presenta un mensaje de error.

FUNCION VERIFY_ENGRANE.- Esta función examina los datos que se han introducido hasta que encuentra alguna forma de dato no válidos. Presenta un mensaje de error si encuentra datos no validos.

Una variable local **OK_PARA,** se pone en nulo cuando se encuentra un error, no permitiendo de ésta manera que se salga del letrero de diálogo hasta que dicho error sea corregido. En el

momento que la variable **OK_PARA** tome un valor diferente de nulo, se sale del cuadro de diálogo y se ejecuta el programa graficando el engranaje.

FUNCION GET_<NOMBRE>.- Todas las funciones que comienzan con **"(GET_"** están diseñadas para ser activadas cuando el usuario selecciona varias mosaicos de entrada de datos en el cuadro de diálogo, obteniendo el valor introducido por el usuario, y a la vez comprueba si dicho valor puede ser convertido por la función **"OK_REAL"**.

FUNCION GET_ALERTA.- Esta función está diseñada para presentar un cuadro de diálogo de advertencia en el momento en que se quiera guardar los parámetros del engranaje en un archivo y éste archivo es existente en el directorio actual de trabajo.

En el momento que el usuario introduzca el nombre del archivo a guardar los parámetros del engranaje, la función busca si se encuentra dicho nombre en el directorio actual de trabajo,

en caso de encontrarse presentara el cuadro de diálogo de la FIGURA 5.3 preguntando si desea remplazar el archivo existente. Si la respuesta es afirmativa, toda la información existente en el archivo será borrada y será remplazada por los nuevos valores, Si la respuesta es negativa, el programa volverá a pedir al usuario que introduzca un nuevo nombre para el archivo.

FUNCION C:DENGRANE.- La función principal **C:DENGRANE** define una nueva función de AutoCAD, que tras desactivar las marcas auxiliares y eco de ordenes activa el cuadro de diálogo **ENGRANE.DCL** y llama a las funciones anteriormente explicadas.

La figura 5.4 muestra el diagrama de flujo de las funciones del programa DENGRANE.LSP.

FIGURA 5. 3 Letrero de advertencia

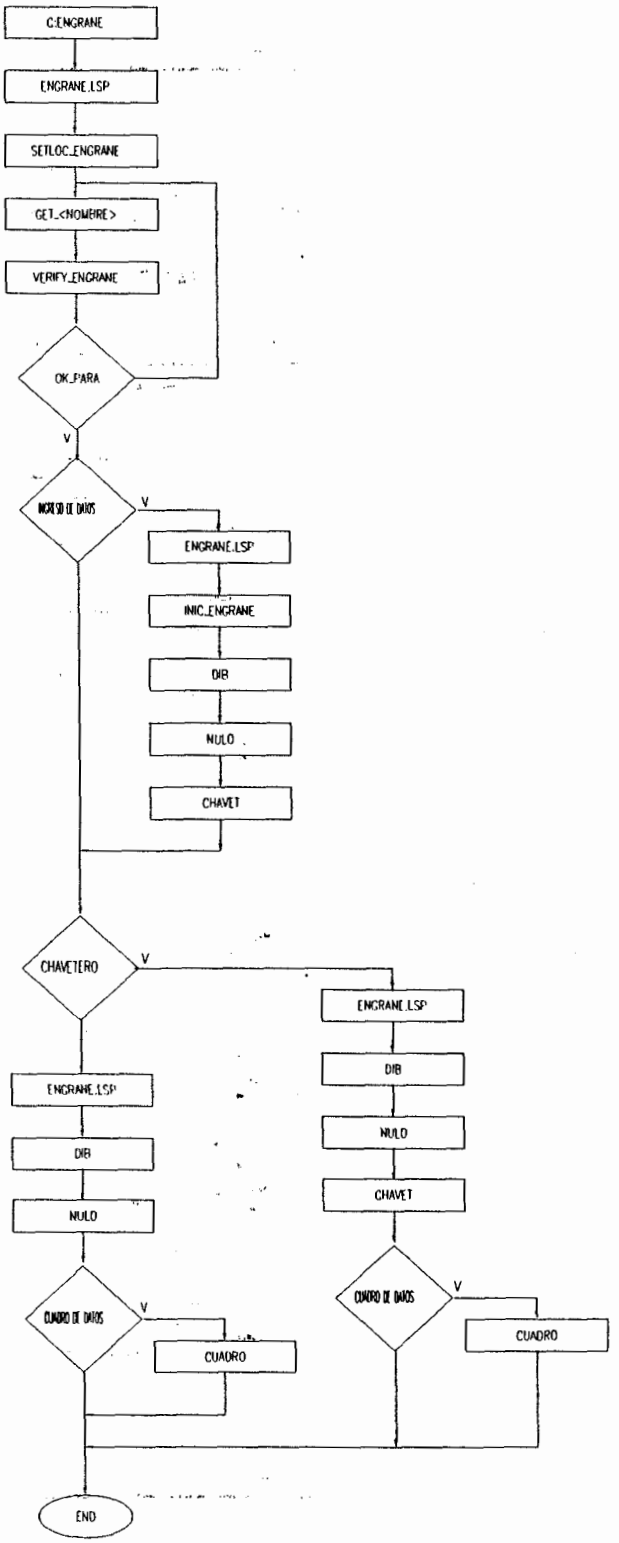


FIGURA 5.4 Diagrama de Flujo del Programa ENGRANE.LSP

5.4 EXPLICACION DEL USO DEL PROGRAMA

Para ejecutar el Programa sobre Diseño Paramétrico de Engranajes de Dientes Rectos, se procede como sigue:

Por el teclado, a partir del indicador [prompt] **Command, teclee: MENU,** Seleccione TESIS.MNX, y a continuacion **OK.**

En el sistema de Menúes desplegables, seleccione:

DRAW/E-MAQUINA/ENGRANE (FIG. 5.5)

La Orden **ENGRANE** utiliza un letrero de diálogo para especificar todos los datos necesarios para poder graficar los engranajes de dientes rectos (FIG. 5.6).

- Presentación de Resultados:

  Esta área sirve para insertar los datos necesarios para guardar o presentar los valores calculados para el engranaje.

  1. Resultados en Archivo:

     Activada ésta casilla, los parámetros del engranaje se guardaran en un archivo cuyo nombre se coloca la casilla "File:" con o sin extensión, ésta casilla solo acepta la extensión ".RPZ" y máximo ocho caracteres en el nombre del archivo.

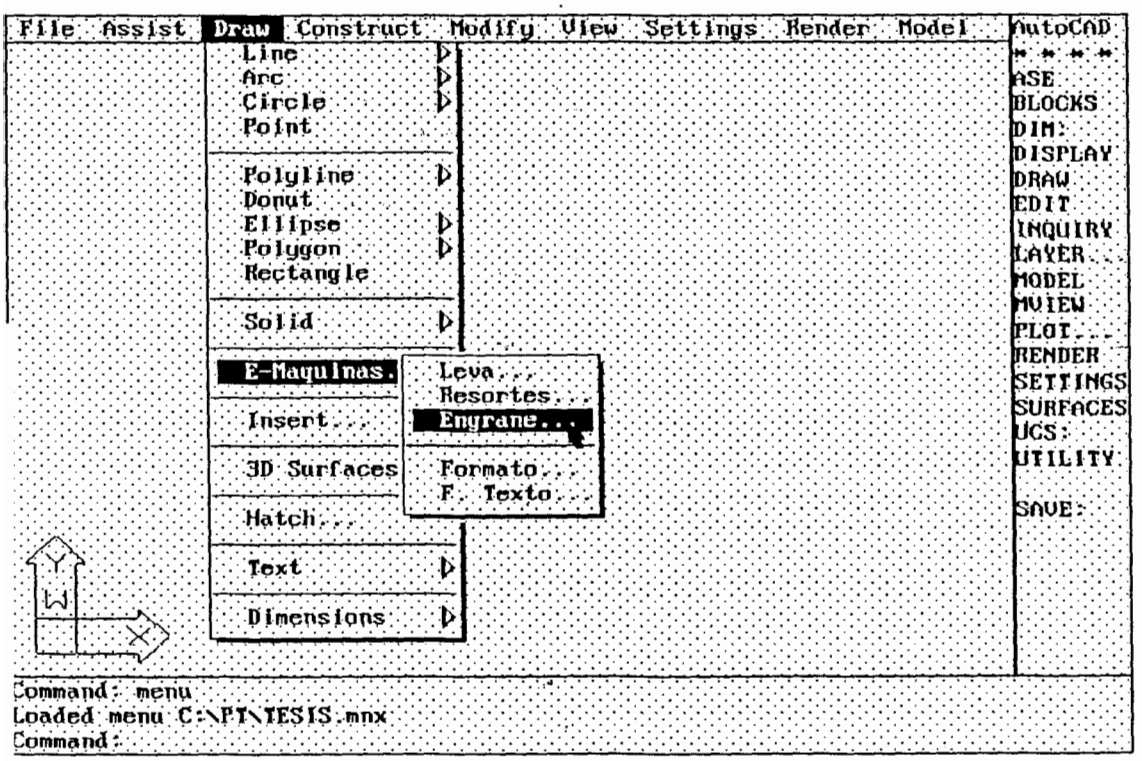


FIGURA 5. 5 Menú desplegable de la Orden ENGRANE

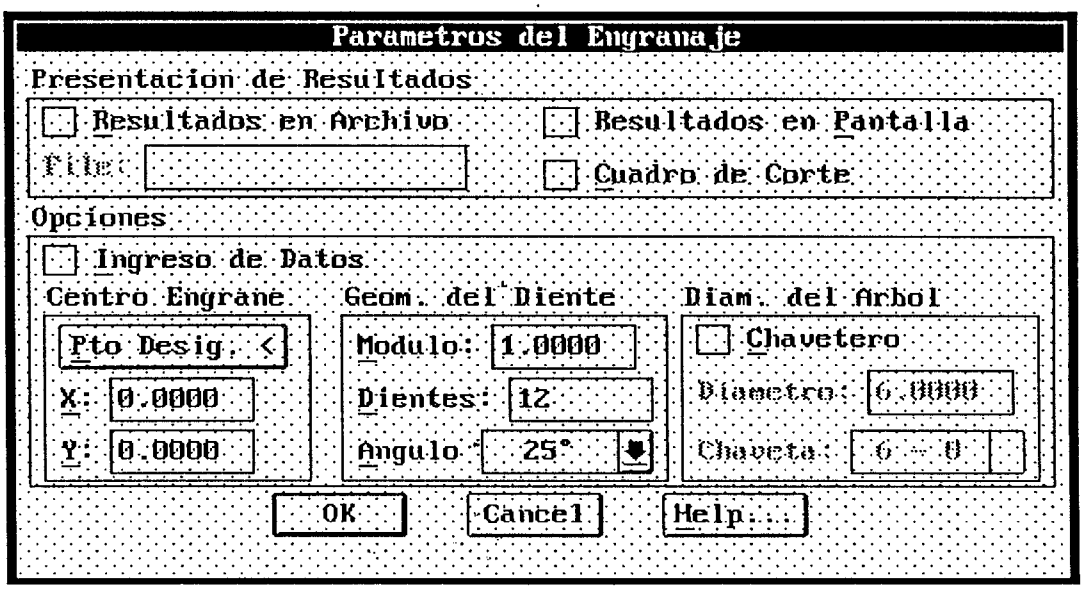


FIGURA 5. 6 Orden ENGRANE. Letrero de diálogo

2. Resultados en Pantalla:

Activada ésta casilla, los parámetros del engranaje serán presentados en una lista en la pantalla de texto de AutoCAD (TABLA 5.2).

3. Cuadro de Corte:

Activada ésta casilla, los parámetros más importantes para el diseño se imprimirán en un cuadro en la pantalla gráfica de AutoCAD como una parte del dibujo (FIGURA 5.7).

- Opciones:

Esta área sirve para aportar los datos necesarios para poder graficar el engranaje seleccionando dos procedimientos:

1. Ingreso de Datos:

Activa ésta casilla, el Punto centro del engranaje, el Modulo, el Número de Dientes, Ángulo de Presión, el Diámetro de la Chaveta del engranaje deberán ser

DATOS DE CORTE	
MODULO	1.00
NUMERO DE DIENTES	12.00
DIAMETRO PRIMITIVO	12.00mm
PASO CIRCUNFERENCIAL	3.14mm
ANGULO DE PRESION	25.00°
PROFUNDIDAD DEL DIENTE	2.17mm
ADENDO	1.17mm
RADIO DE ENTALLE	0.30mm

FIGURA 5.7 Cuadro de Datos del engranaje

indicados desde la línea de ordenes de AutoCAD.

2. Punto Centro, Modulo, N. de Dientes:

Desactivada la casilla anterior, la casilla de edición correspondiente al Punto Centro, Modulo, Número de Dientes y el Diámetro de la Chaveta del engranaje se habilitan para su uso y modificación.

CAPITULO VI

AUTOMATIZACION DE ROTULADO DE PLANOS

6.1 INTRODUCCIÓN

El programa desarrollado en este capitulo es de una gran utilidad ya que simplifica enormemente el tiempo de trabajo para los usuarios de AutoCAD que lo aplican en el diseño de la Ingeniería Mecánica.

La rutina consiste en la graficación del cuadro de rotulación normalizado que se emplea en Ingeniería Mecánica, además el programa brinda al usuario la oportunidad de crear un cuadro de especificaciones con solo ingresar el número de filas a editar, el cuadro de especificaciones se ubicara en la parte superior del cuadro de rotulación. Las divisiones de las filas y de las columnas son creadas de forma automática por el interprete AutoLISP. El cuadro de rotulación y el cuadro de especificaciones en el momento de insertarse lo hace en una LAYER[2] denominada "FORMATO". Esto fue hecho con el propósito de poder cambiar inmediatamente cualquier propiedad

que brinda el comando **LAYER** de AutoCAD, exceptuando el color de las líneas, ya que ésta propiedad no es por capa. La única forma de poder cambiar el color de las líneas una vez insertado el **FORMATO** es haciendo uso del comando **COLOR**[2] de AutoCAD.

Con la ayuda de un programa auxiliar y una vez insertado el **FORMATO** se procede a su llenado tan solo escribiendo por medio del teclado, además el usuario puede seleccionar el tipo de letra a usar, ya que el programa proporciona 26 tipos de letras a elegir, también ofrece al usuario la oportunidad de seleccionar el color del texto, y en la capa que se va a insertar el texto.

6.2 EXPLICACION DEL PROGRAMA.

El programa **FORMATO.LSP** es la rutina responsable que hace posible la inserción del formato de dibujo que se utiliza en Ingeniería Mecánica. Las funciones por el cual está constituido el programa **FORMATO.LSP** se detalla a continuación.

FUNCION OK_REAL.- Es una función que tiene por objeto de comprobar los datos de cadena antes de convertirlos en datos numéricos reales, si la cadena no puede convertirse, se presentara un mensaje de error en el cuadro de diálogo.

FUNCION INIC_FORMATO.- Esta función está diseñada para graficar el borde que encierra el área de dibujo, y preparar el contorno de trabajo.

La función **"INIC_FORMATO"** se inicia creando una nueva capa denominada **"FORMATO"** y además la activa con el propósito de insertar el cuadro de rotulación en dicha capa. A continuación el programa cambia los limites del dibujo según el formato de la lamina escogido por el usuario, y por último gráfica la línea de bordes del área de dibujo e inserta el cuadro de rotulación del archivo **FORMAT.DWG.** La forma de los bordes del área de dibujo depende de los limites seleccionados y de la selección del formato ya sea ésta horizontal o vertical (FIGURA 6.1 y 6.2).

FUNCION CAJSUP.- Esta rutina tiene por objetivo principal el dibujar el cuadro de especificaciones.

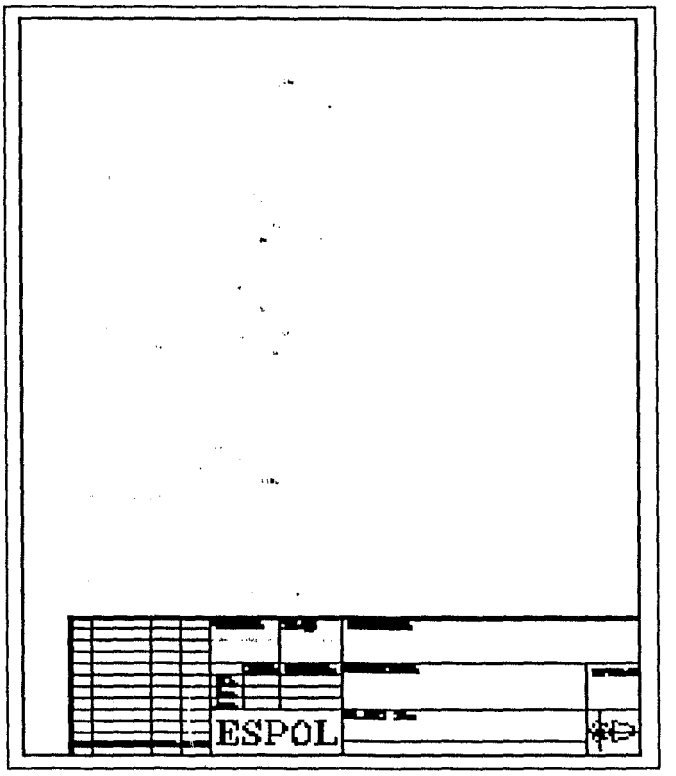


FIGURA 6. 1 Formato A-4 Tipo Vertical

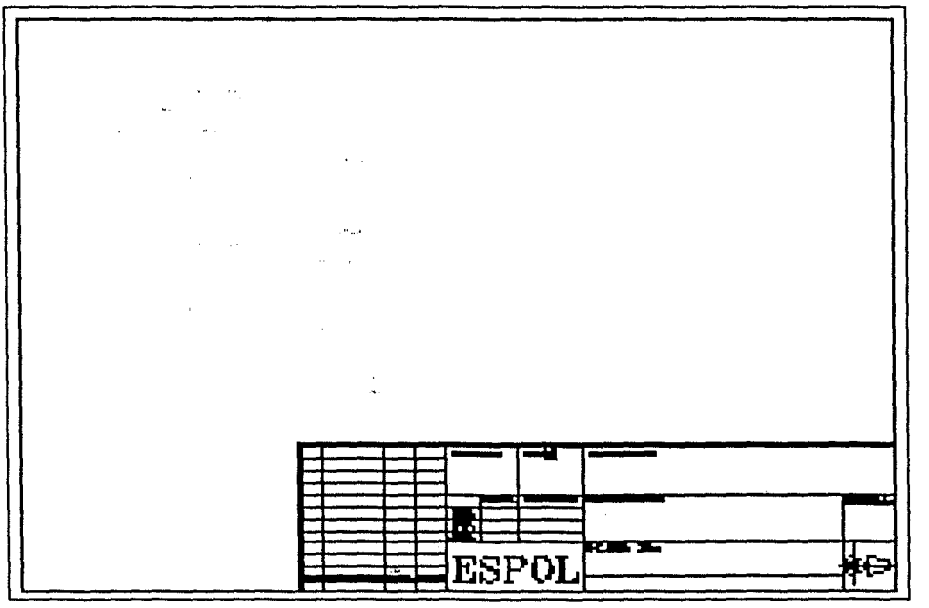


FIGURA 6. 2 Formato A-4 Tipo Horizontal

El programa se inicia con un condicional "IF[3]", si el condicional toma la rama del "No", entonces se visualiza en la línea de ordenes el mensaje "Formato insertado", claro está que anteriormente se inserto el cuadro de rotulación. Si el condicional se va por la rama del "Si", se inicia un lazo de repetición **"REPEAT[3]"** que es controlado por el número de filas a crear, dentro de este lazo de repetición se ejecuta la función **COMMAND[3]** para crear las líneas horizontales para que se vallan formando las filas, una vez creadas las filas, se emplea siete veces el comando **COMMAND** para crear las línea verticales y de ésta forma queda formadas la siete columnas de división del cuadro de especificaciones (FIG. 6.3).

FUNCION GET_<NOMBRE>.- Varias funciones que empiezan con los caracteres **<<GET_>>** están diseñadas para ser activadas cuando el usuario selecciona varios mosaicos de entrada de datos en el cuadro de diálogo.

Si, los mosaicos son de casillas de edición, la función controla la entrada de datos comprobando que la cadena se pueda convertir en un número real,

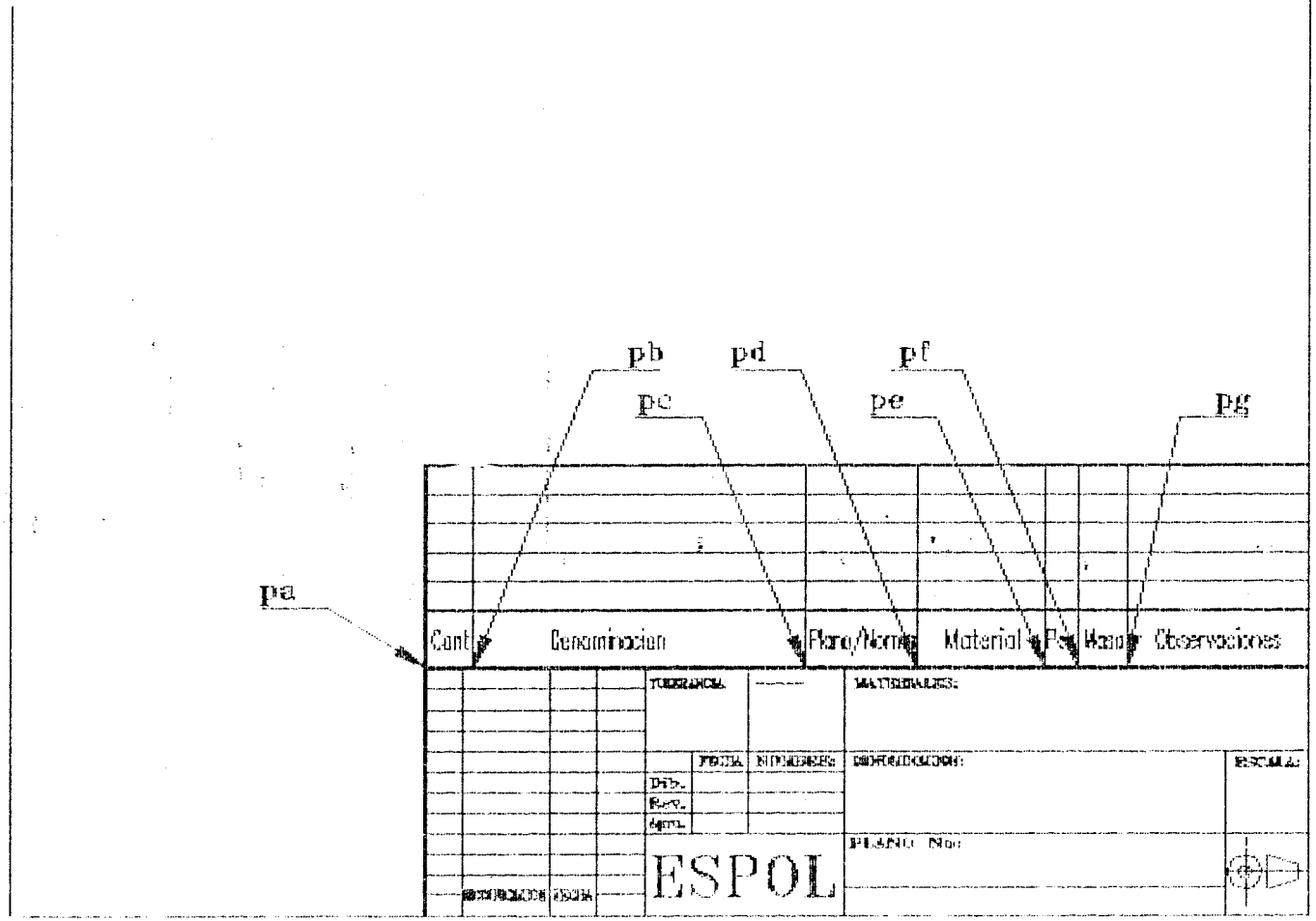


FIGURA 6.3 Cuadro de Rotulacion y Cuadro de Especificacion

cuando sea apropiado usando la función "OK_REAL". Si la función encuentra algún dato nulo o mal introducido, se presentara un mensaje de error para que sea modificado dicho dato.

Si los mosaicos son de casillas de imagen o casilla de recuadro de lista, la función simplemente acepta el icono o la lista del recuadro seleccionado sin comprobarlo, sino simplemente lo acepta y lo ejecuta.

FUNCION SETLOC_<NOMBRE>.- Las funciones que empiezan con los caracteres **<<SETLOC_>>** están diseñadas para inicializar las variables locales, las imágenes, o bien los valores por defectos la primera vez que se ejecuta la rutina.

Generalmente las funciones de este tipo tienen una serie de funciones "IF" que comprueban la existencia de los valores globales, imágenes, e inicializan las variables locales. Además los datos numéricos se convierten en datos de cadenas, usando las funciones "RTOS[3]" e "ITOA[3]" cuando los mosaicos a inicializar son casillas de edición.

FUNCION VERIFY_<NOMBRE>.- Las funciones que empiezan con los caracteres **<<VERIFY_>>** están diseñadas para examinar los datos introducidos hasta que encuentra alguna forma de dato nulo o alguna condición que no se cumpla. Presenta un mensaje de error si encuentra datos no validos. Una variable local especial, **"OK_PARA"**, se pone en nulo cuando encuentra un dato mal introducido, impidiendo de ésta manera poder abandonar el cuadro de diálogo hasta que dichos datos sean modificados por el usuario.

FUNCION IMAGE_ON.- Esta función está diseñada para presentar el cuadro de diálogo de los símbolos de disposición de las vistas (FIGURA 6.8).

FUNCION IMAGEN1_ON.- Esta función está diseñada para presentar el cuadro de diálogo de los dos diferentes tipos rotulación (FIGURA 6.7).

FUNCION C:FORMATO.- Finalmente, cuando toldas las funciones se han definido están preparadas para ser reunidas dentro de una orden de AutoCAD. Las ordenes de AutoCAD se definen igual que las funciones, pero

sus nombres empiezan por C:. Esta técnica especial de denominación indica a AutoCAD que éstas funciones se pueden llamar directamente desde el editor de ordenes.

La función se limita a guardando en variables el color actual, el estilo de texto actual, la capa actual del dibujo, y además desactiva las marcas auxiliares, ecos de ordenes, el modo de arrastre, y el símbolo de icono. A demás llama a todas las funciones anteriormente explicadas en este capitulo y presenta el cuadro de diálogo principal (FIG. 6.5).

La figura 6.4 muestra el diagrama de flujo de las funciones del programa **FORMATO.LSP.**

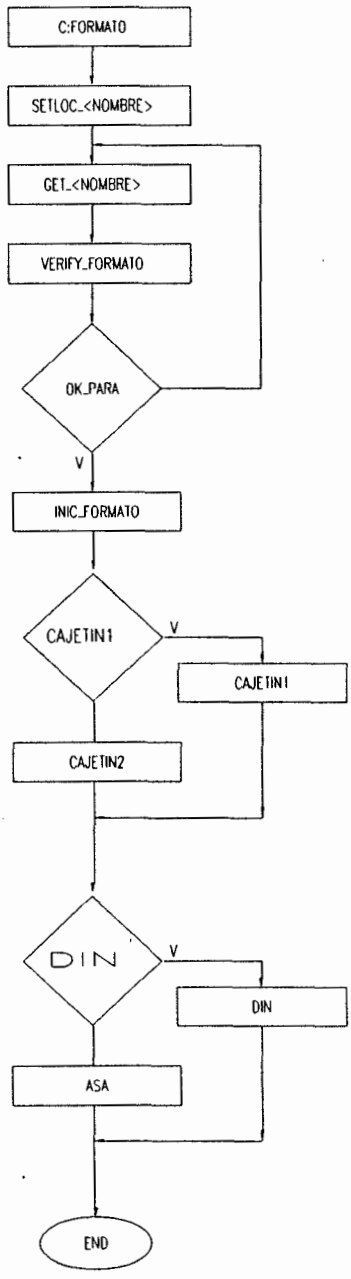


FIGURA 6.4 Diagrama de Flujo del Programa FORMATO.LSP

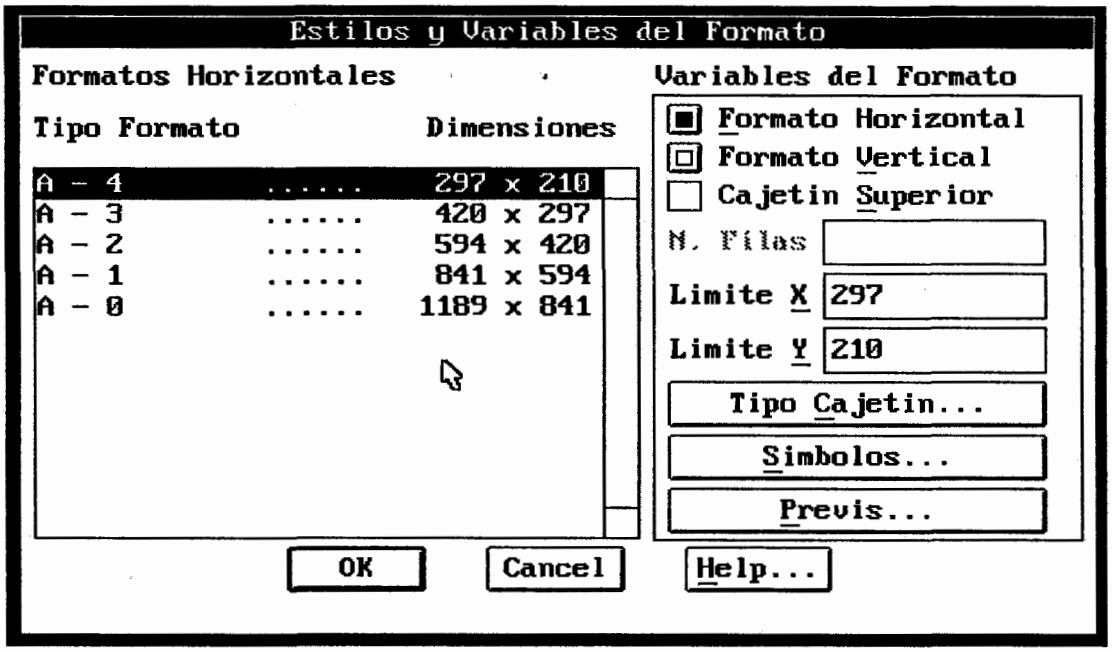


FIGURA 6. 5 Orden FORMATO. Letrero de diálogo

6.3 EXPLICACION DEL USO DEL PROGRAMA.

La orden **FORMATO** utiliza un letrero de diálogo para especificar todos los datos necesario para la graficación del formato de dibujo (FIGURA 6.5).

La secuencia de trabajo:
Por el teclado, a partir del indicador [prompt] **Command, teclee: MENU,** Seleccione TESIS.MNX, y a continuacion **OK.**

En el sistema de Menús desplegables, seleccione:

DRAW/E-MAQUINA/FORMATO (FIGURA 6.6).

El letrero de diálogo de la orden está dividido en dos zonas.
El área de la izquierda visualiza los tipos de formatos normalizados disponibles para el dibujo. La selección del formato a usar se la hace simplemente con el ratón.

En cuanto al las variables del formato, una serie de casillas acceden a la modificación del formato a insertar.

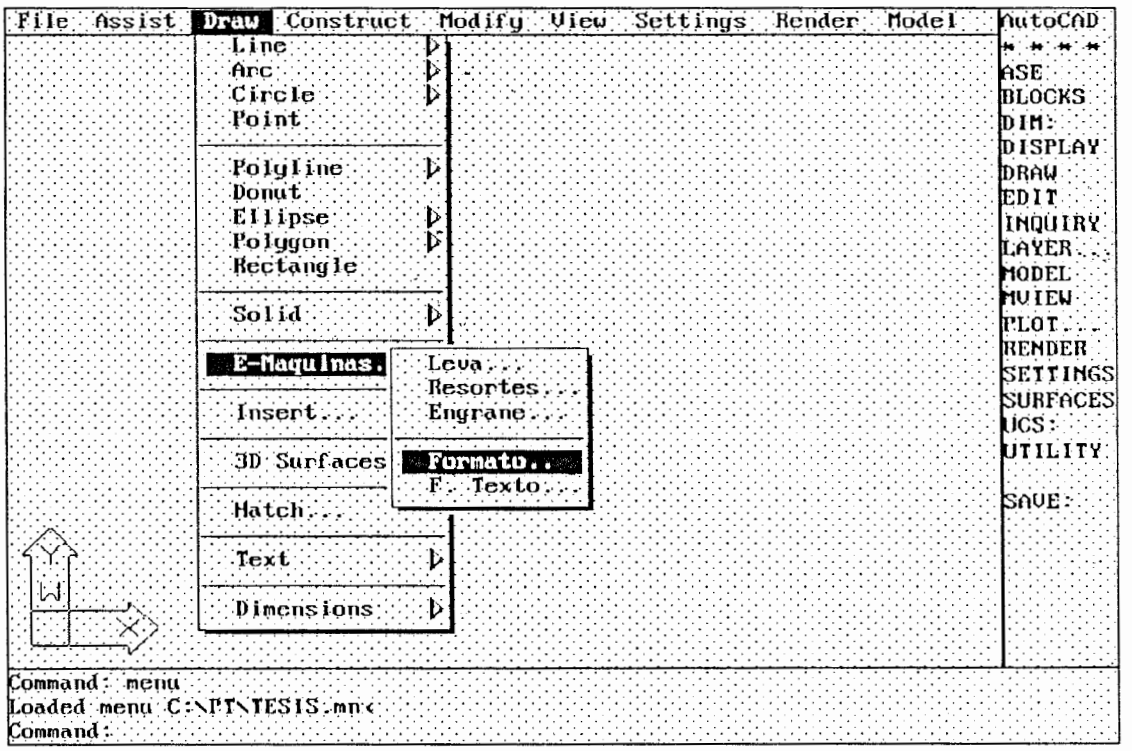


FIGURA 6. 6 Menú desplegable de la Orden FORMATO

- **Formato Horizontal**

  Esta casilla indica que el formato se insertara de forma horizontal, por ejemplo para un formato A4 los limites serian 297 x 210.

- **Formato Vertical**

  Esta casilla indica que el formato se insertara en forma vertical, por ejemplo para el formato A4 los limites serian 210 x 297.

- **Cajetín Superior**

  Al activar ésta casilla se habilita la casilla de edición del número de filas.

- **N. Filas**

  Esta casilla de edición acepta el número de filas a crear para el cuadro de especificaciones pero antes debe ser habilitada primero.

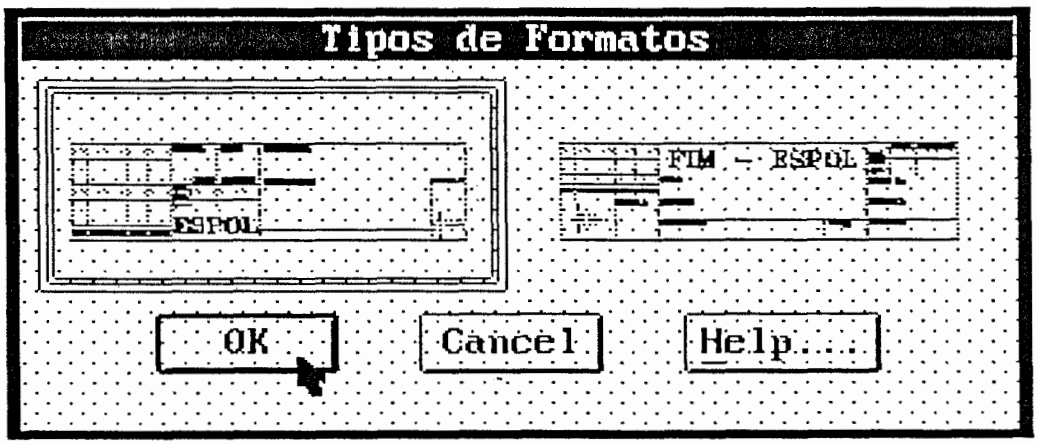


FIGURA 6. 7 Cuadro de Rotulación

- **Limite X, Limite Y**

  Estas dos casillas de edición aceptan cualquier valor para los limites del formato.

- **Tipo Cajetín**

  La casilla "Tipo Cajetín..." accede al letrero de diálogo de los dos diferentes tipos de cuadro de rotulación (FIGURA 6.7)

- **Símbolos**

  La casillas "Símbolos..." accede al letrero de diálogo de los símbolos de disposición de vistas (FIGURA 6.8).

- **Previous**

  La casilla "Previous..." accede a presentar previamente el formato.

Para el llenado automático del cajetin la secuencia de trabajo es:

En el sistema de Menús desplegables, seleccione:

DRAW/E-MAQUINA/F-TEXTO (FIGURA 6.6).

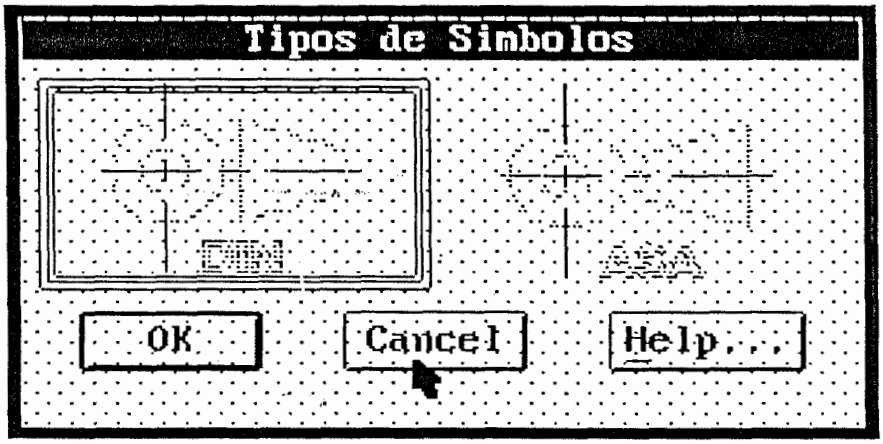


FIGURA 6.8 Símbolos de Disposición de Vistas

CONCLUSIONES Y RECOMENDACIONES

Después del desarrollo de los programas que sirven al diseñador que opera el AutoCAD 12 para elaborar gráficos en computadoras de elementos mecánicos importantes como son: engranajes, levas y resortes. Estos gráficos son diseñados partiendo de datos técnicos como:

- Engranajes de dientes rectos

  Modulo, número de dientes, ángulo de presión, y el diámetro del eje

- Leva de placa plana

  Partiendo del nombre de un archivo con extensión .LEV, el programa se encarga automáticamente de unir los puntos del perfil de la leva para formar el elemento buscado (Leva).

- Resortes Helicoidales

Diámetro del alambre, tipo del resorte, diámetro del resorte, número de espiras y el paso del resorte

Se logró un enlace entre el programa graficador (AutoCAD) y un programa de lenguaje medio (Turbo C++) bajo un archivo

escrito en código ASCII, que dio como resultado de un análisis de diseño que fue realizado en la Facultad de Ingeniería en Mecánica como son los de resortes helicoidales y leva de placa plana. Para mejorar la facilidad de operación del programa se desarrolló un cuadro de diálogo que permite que el usuario tenga una mejor -- visualización y versatilidad para ingresar los parámetros requeridos para obtener el diseño gráfico.

El programa de diseño paramétrico de engranaje de dientes recto está basado en el estudio de la geometría de la involuta y presenta una caja de diálogo que permite un ingreso fácil de los parámetros. El programa calcula los parámetros de los dientes del engranaje y los presenta por pantalla, o si el usuario lo prefiere puede almacenarlo en un archivo de texto para su posterior revisión.

Para mejorar la facilidad y rapidez del llenado del cajetín utilizado en la Facultad de Ingeniería en Mecánica se desarrolló un programa que por medio de una caja de diálogo permite seleccionar el tipo de cajetín, el tamaño del plano, la posición (sea horizontal o vertical), el símbolo DIN o ASA y 24 tipos diferentes de letra.

RECOMENDACIONES

Se debe utilizar como herramienta de ingeniería básica el lenguaje de computación AutoLISP para diseñar gráficamente y en forma automática los elementos mecánicos, ya que AutoLISP tendrá siempre la ventaja de un desarrollo más sencillo y más rápido con un pequeño sacrificio de potencia y velocidad.

Como AutoCAD tiene relación directa con los programas de control numérico y de elemento finito, AutoLISP se convierte en una herramienta poderosa en la Ingeniería Mecánica para el diseño gráfico de elementos de maquinas que servirán para la construcción y producción en serie en la industria manufacturera en el país con una excelente calidad, rapidez y confiabilidad.

APENDICE A

LISTADO DE LOS PROGRAMAS

LISTADO 3.1 La rutina MUELLE.LSP

165

```
;=============================================================================
;              NOMBRE DE ARCHIVO:    MUELLE.LSP - AutoCAD Versión 12
;                      Escrito por:     Renato Parodi
;
;=============================================================================

(DEFUN inic_muelle ()
  (GRAPHSCR)
  (PROMPT "Este programa dibuja muelles") (TERPRI)
  (IF (NOT op)
    (SETQ g:op "Simple")
    (SETQ g:op op)
  )
  (INITGET "Simple Rebajado Escuadra Gancho Aplan-es")
  (IF (SETQ op (GETKWORD
                 (STRCAT "Gancho/Escuadra/Rebajado/Aplan-escuad/Simple"
                         "<" g:op ">: ")))
      () (SETQ op g:op)
  ) (TERPRI)
  (IF (NOT pin)
      (SETQ x "0.0000" y "0.0000")
      (SETQ x (RTOS (CAR pin) 2) y (RTOS (CADR pin) 2))
  )
  (IF (SETQ pin (GETPOINT
                  (STRCAT "Punto de inicio del alambre <" x "," y "> :")))
      () (SETQ pin (LIST (ATOF x) (ATOF y)))
  ) (TERPRI)
  (IF (NOT dal)
      (SETQ g:dal "0.2000")
      (SETQ g:dal (RTOS dal 2))
  )
  (INITGET 6)
  (IF (SETQ dal (GETREAL
                  (STRCAT "Diametro del alambre <" g:dal ">: ")))
      () (SETQ dal (ATOF g:dal))
  ) (TERPRI)
  (IF (NOT dres)
      (SETQ g:dres "0.7000")
      (SETQ g:dres (RTOS dres 2))
  )
  (INITGET 6)
  (IF (SETQ dres (GETREAL
                   (STRCAT "Diametro del resorte <" g:dres ">: ")))
      () (SETQ dres (ATOF g:dres))
  ) (TERPRI)
  (IF (NOT pso)
      (SETQ g:pso "0.3500")
      (SETQ g:pso (RTOS pso 2))
  )
  (INITGET 6)
  (IF (SETQ pso (GETREAL
                  (STRCAT "Paso del resorte <" g:pso ">: ")))
      () (SETQ pso (ATOF g:pso))
  ) (TERPRI)
  (WHILE (< pso dal)
         (PROMPT "El paso debe ser mayor al diametro del alambre") (TERPRI)
         (INITGET 6)
         (IF (SETQ pso (GETREAL
                         (STRCAT "Paso del resorte <" g:pso ">: ")))
             () (SETQ pso (ATOF g:pso))
         ) (TERPRI)
  )
  (IF (NOT ns)
      (SETQ g:ns "5")
      (SETQ g:ns (ITOA ns))
  )
 (INITGET 6)
  (IF (SETQ ns (GETINT
```

```
                (STRCAT "Numero de espiras <" g:ns ">: ")))
      () (SETQ ns (ATOI g:ns))
  ) (TERPRI)
  (IF (= op "Gancho")
      (PROGN
        (IF (NOT rgan)
            (SETQ g:rgan "0.7000")
            (SETQ g:rgan (RTOS rgan))
        )
        (INITGET 6)
        (IF (SETQ rgan (GETREAL
                         (STRCAT "Radio del gancho <" g:rgan ">: ")))
            () (SETQ rgan (ATOF g:rgan))
        ) (TERPRI)
        (WHILE (< rgan (/ dres 2.0))
          (PROMPT "El radio del gancho debe ser mayor al radio del resorte") (TERPRI)
          (INITGET 6)
          (IF (SETQ rgan (GETREAL
                           (STRCAT "Radio del gancho <" g:rgan ">: ")))
              () (SETQ rgan (ATOF g:rgan))
          ) (TERPRI)
        )
      )
  )
)
;-------------------------------------------------------------------------
(DEFUN para_file ()
  (SETQ pin (GETPOINT "Punto de inicio del resorte: ")) (TERPRI)
  (SETQ fich (OPEN fich "r"))
  (SETQ n 1 r 1)
  (WHILE r
     (SETQ cad (READ-LINE fich))
     (IF (/= cad nil)
         (PROGN
            (SETQ expr (STRCAT
                         "(SETQ p"(ITOA n)" (ATOF (SETQ q (SUBSTR cad 27))))")
            )
            (EVAL (READ expr))
            (SETQ n (+ n 1))
         )
         (SETQ r nil)
     )
  )
  (CLOSE fich)
  (SETQ op p1 dal p2 dres p3 pso p4 ns p5 rgan p6 mat q)
  (COND ( (= op 1.0) (SETQ op "Simple"))
        ( (= op 2.0) (SETQ op "Rebajado"))
        ( (= op 3.0) (SETQ op "Escuadra"))
        ( (= op 4.0) (SETQ op "Aplan-es"))
        ( (= op 5.0) (SETQ op "Gancho"))
  )
  (SETQ d (SQRT (+ (EXPT dres 2) (EXPT (- pso dal) 2))))
  (SETQ ang (ATAN (/ (- pso) 2.0) dres))
  (SETQ p1 (POLAR pin (/ PI 2.0) (/ dal 2.0)))
  (SETQ p2 (POLAR p1 (- (/ PI 2.0)) dal))
  (SETQ p3 (POLAR p1 (+ PI (- ang)) d))
  (SETQ p4 (POLAR p3 (/ (- PI) 2.0) dal))
  (SETQ pf (POLAR pin (+ PI (- ang)) d))
  (SETQ n 0)
)
;-------------------------------------------------------------------------
(DEFUN int ()
  (SETQ p5 (POLAR pf (/ PI 2.0) (/ dal 2.0)))
  (SETQ p6 (POLAR p5 (- (/ PI 2.0)) dal))
  (SETQ p7 (POLAR p5 (+ PI (- ang)) d))
  (SETQ p8 (POLAR p6 (+ PI (- ang)) d))
  (SETQ i1 (INTERS P2 P4 P3 P5 nil))
  (SETQ i2 (INTERS p4 p6 p5 p7 nil))
)
;-------------------------------------------------------------------------
(DEFUN act ()
  (SETQ pin pf p1 p5 p2 p6 p3 p7 p4 p8)
```

```
  (SETQ n (+ n 1))
)

;-------------------------------------------------------------------------
(DEFUN prog ()
  (REPEAT (FIX ns)
     (COMMAND "arc" "c" pin p2 p1)
     (SETQ pin pf)
     (SETQ pf (POLAR pin ang d))
     (int)
     (COMMAND "line" p1 p3 "")
     (COMMAND "line" p2 p4 "")
     (COMMAND "arc" "c" pin p3 p4)
     (IF (/= (+ n 1) ns)
         (PROGN
            (COMMAND "line" i1 p5 "")
            (COMMAND "line" p4 i2 "")
            (act)
            (SETQ pf (POLAR pin (+ PI (- ang)) d))
         )
     )
 )
)
;-------------------------------------------------------------------------
(DEFUN simp ()
  (SETQ d (SQRT (+ (EXPT dres 2) (EXPT (- pso dal) 2))))
  (SETQ ang (ATAN (/ (- pso) 2.0) dres))
  (SETQ p1 (POLAR pin (/ PI 2.0) (/ dal 2.0)))
  (SETQ p2 (POLAR p1 (- (/ PI 2.0)) dal))
  (SETQ p3 (POLAR p1 (+ PI (- ang)) d))
  (SETQ P4 (POLAR p3 (/ (- PI) 2.0) dal))
  (SETQ pf (POLAR pin (+ PI (- ang)) d))
  (SETQ n 0)
  (IF (= op "Simple")
      (PROGN
         (SETQ pa (POLAR p1 (+ PI ang) d))
         (SETQ pb (POLAR pa (- (/ PI 2.0)) dal))
         (SETQ i1 (INTERS pb p2 p1 p3 nil))
         (COMMAND "line" pa p1 "")
         (COMMAND "line" pb i1 "")
         (COMMAND "circle" "2p" pa pb)
         (prog)
         (COMMAND "line" i1 p5 "")
         (COMMAND "line" p4 p6 "")
         (COMMAND "circle" "2p" p5 p6)
      )
  )
  (IF (= op "Rebajado")
      (PROGN
         (SETQ ns (- ns 1))
         (SETQ pa (POLAR p1 PI (+ dres (/ dal 2.0))))
         (SETQ cen (POLAR p1 PI dres))
         (SETQ pb (POLAR cen (/ (- PI) 2.0) (/ dal 2.0)))
         (SETQ pc (POLAR p1 PI (/ dres 2.0)))
         (SETQ i1 (INTERS pb p2 p1 p3 nil))
         (COMMAND "line" p1 pa "")
         (COMMAND "line" pb i1 "")
         (COMMAND "arc" "c" cen pa pb)
         (COMMAND "line" pb pc "")
         (prog)
         (SETQ pa (POLAR p4 0 (+ dres (/ dal 2.0))))
         (SETQ cen (POLAR p4 0 dres))
         (SETQ pb (POLAR cen (/ PI 2.0) (/ dal 2.0)))
         (SETQ pc (POLAR p4 0 (/ dres 2.0)))
         (SETQ i1 (INTERS pb p3 p2 p4))
         (COMMAND "line" p4 pa "")
         (COMMAND "line" pb i1 "")
         (COMMAND "arc" "c" cen pa pb)
         (COMMAND "line" pb pc "")

  )
```

```
(IF (= op "Escuadra")
    (PROGN
       (SETQ ns (- ns 1))
       (SETQ pa (POLAR p1 (+ PI ang) d))
       (SETQ cen (POLAR pa (/ (- PI) 2.0) (/ dal 2.0)))
       (SETQ pb (POLAR pa (- (/ PI 2.0)) dal))
       (SETQ pc (POLAR pa 0 (/ dres 2.0)))
       (SETQ pd (POLAR pb 0 (/ dres 2.0)))
       (SETQ i1 (INTERS p1 pa pc pd nil))
       (SETQ i2 (INTERS pb p2 p1 p3 nil))
       (COMMAND "arc" "c" cen pa pb)
       (COMMAND "line" pa pc "")
       (COMMAND "line" pb pd "")
       (COMMAND "line" pc pd "")
       (COMMAND "line" p1 i1 "")
       (COMMAND "line" pb i2 "")
       (prog)
       (SETQ cen (POLAR p5 (/ (- PI) 2.0) (/ dal 2.0)))
       (SETQ pc (POLAR p5 PI (/ dres 2.0)))
       (SETQ pd (POLAR P6 PI (/ dres 2.0)))
       (SETQ i1 (INTERS p4 p6 pc pd))
       (SETQ i2 (INTERS p3 p5 p2 p4))
       (COMMAND "line" i2 p5 "")
       (COMMAND "line" p4 i1 "")
       (COMMAND "arc" "c" cen p6 p5)
       (COMMAND "line" p5 pc "")
       (COMMAND "line" p6 pd "")
       (COMMAND "line" pc pd "")
    )
)
(IF (= op "Aplan-es")
    (PROGN
       (SETQ pa (POLAR p1 PI (+ dres (/ dal 2.0))))
       (SETQ cen (POLAR p1 PI dres))
       (SETQ pb (POLAR cen (/ (- PI) 2.0) (/ dal 2.0)))
       (SETQ i1 (INTERS pb p2 p1 p3 nil))
       (COMMAND "line" p1 pa "")
       (COMMAND "line" pb i1 "")
       (COMMAND "arc" "c" cen pa pb)
       (COMMAND "line" pb p1 "")
       (prog)
       (SETQ pa (POLAR p4 0 (+ dres (/ dal 2.0))))
       (SETQ cen (POLAR p4 0 dres))
       (SETQ pb (POLAR cen (/ PI 2.0) (/ dal 2.0)))
       (SETQ pc (POLAR p4 0 (/ dres 2.0)))
       (SETQ i1 (INTERS pb p3 p2 p4))
       (COMMAND "line" p4 pa "")
       (COMMAND "line" pb i1 "")
       (COMMAND "arc" "c" cen pa pb)
       (COMMAND "line" pb p4 "")
    )
  )
(IF (= op "Gancho")
    (PROGN
       (SETQ pa (POLAR p1 (+ PI ang) (/ d 2.0)))
       (SETQ cen (POLAR pa (/ PI 2.0) rgan))
       (SETQ dang 0.1)
       (WHILE dang
          (SETQ pb (POLAR cen (+ PI dang) (+ rgan dal)))
          (SETQ i1 (INTERS cen pb p3 p1))
          (IF (/= i1 nil) (SETQ dang nil) (SETQ dang (+ 0.01 dang)))
       )
       (SETQ pn1 (POLAR cen 0 rgan))
       (SETQ pn2 (POLAR cen 0 (+ rgan dal)))
       (COMMAND "line" pn1 pn2 "")
       (COMMAND "line" p1 pa "")
       (COMMAND "arc" "c" cen pn1 pa)
       (COMMAND "arc" "c" cen pn2 pb)
       (prog)
       (SETQ pa (POLAR p4 ang (/ d 2.0)))
       (SETQ cen (POLAR pa (/ (- PI) 2.0) rgan))
       (SETQ dang 0.1)
```

```
        (WHILE dang
           (SETQ pb (POLAR cen ang (+ rgan dal)))
           (SETQ il (INTERS cen pb p2 p4))
           (IF (/= il nil) (SETQ dang nil) (SETQ dang (+ 0.01 dang)))
        )
        (SETQ pn1 (POLAR cen PI rgan))
        (SETQ pn2 (POLAR cen PI (+ rgan dal)))
        (COMMAND "line" pn1 pn2 "")
        (COMMAND "line" p4 pa "")
        (COMMAND "arc" "c" cen pn1 pa)
        (COMMAND "arc" "c" cen pn2 pb)
      )
  )
)

;-------------------------------------------------------------------------------
(DEFUN texto (tx)
  (COMMAND "text" pins (* 0.015 escl) 0 tx)
  (SETQ pins (LIST (CAR pins) (- (CADR pins) (* 0.04 escl))))
)
;-------------------------------------------------------------------------------
(DEFUN cuadro ()
  (SETQ p1 (GETPOINT "Punto de insercion del cuadro: ")) (TERPRI)
  (SETQ esc (* 4 dres))
  (IF (SETQ escl (GETREAL (STRCAT "Escala del cuadro <"(RTOS esc 2 2)">: ")))
      () (SETQ escl esc)) (TERPRI)
  (COMMAND "insert" "format" p1 1 "" 0)
  (COMMAND "insert" "*cuadro2" p1 escl "")
  (SETQ pins (POLAR p1 0 (* 0.37 escl)))
  (COMMAND "color" "1")
  (COMMAND "layer" "s" "formato" "")
  (IF (NOT dato) (SETQ mat (GETSTRING "Material del muelle: ")))
  (texto mat)
  (texto (STRCAT (RTOS dal 2 2) " " "mm"))
  (texto (STRCAT (RTOS dres 2 2) " " "mm"))
  (texto (STRCAT (RTOS pso 2 2) " " "mm"))
  (texto (STRCAT (RTOS ns 2 2)))
  (IF (= op "Gancho") (texto (STRCAT (RTOS rgan 2 2))))
)
```

LISTADO 3.2 Un archivo DCL para DMUELLE.LSP

```
//=====================================================================
//                MUELLE.DCL - Cuadro de dialogo DMUELLE.LSP
//                        Escrito por: Renato Parodi
//=====================================================================

muelle : dialog {
    label = "Parametros del Resorte";
    : row {
        : column {
            : row {
              : boxed_column {
                label = "Pto de Insercion";
                    : button {
                        label = "Pto Desig.<";
                        key = "pin";
                        mnemonic = "P";
                        fixed_width = true;
                    }
                    : row {
                    : edit_box {
                        key = "x_pt";
                        label = "X:";
                        edit_width = true;
                        mnemonic = "X";
                        value = "0.0000";
                    }
                    : edit_box {
                        key = "y_pt";
                        label = "Y:";
                        edit_width = true;
                        mnemonic = "Y";
                        value = "0.0000";
                    }
                    }

              }
            }
            : edit_box {
                key = "dal";
                label = "Dtro Alambre:";
                edit_width = 16;
                mnemonic = "D";
            }
            : edit_box {
                key = "dres";
                label = "Dtro Resorte:";
                edit_width = 16;
                mnemonic = "r";
            }
            : edit_box {
                key = "pso";
                label = "Paso:";
                edit_width = 16;
                mnemonic = "P";
            }
            : edit_box {
                key = "ns";
                label = "N. Espiras:";
                edit_width = 16;
                mnemonic = "N";
            }
            : edit_box {
                key = "rgan";
                label = "R. Gancho:";
                edit_width = 16;
                mnemonic = "R";
                is_enabled = false;
```

```
        }

    }
    : column {
        : boxed_row {
            label = "Tipos de Resortes";
            : radio_column {
                key ="noi_cluster";
                : radio_button {
                    key = "simple";
                    label ="Simple";
                    mnemonic = "S";
                }
                : radio_button {
                    key = "rebajado";
                    label = "Rebajado";
                    mnemonic = "R";
                }
                : radio_button {
                    key = "escuadra";
                    label = "Escuadra";
                    mnemonic ="E";
                }
                : radio_button {
                    key = "esc-apla";
                    label = "Esc-Apla";
                    mnemonic ="s";
                }
                : radio_button {
                    key = "gancho";
                    label = "Gancho";
                    mnemonic = "G";
                }

            }
            : icon_image {
                key          = "icono";
            }

        }
        spacer_1;
        : toggle {
            label = "Datos por Archivos";
            key = "dato";
            mnemonic = "D";
        }
        : toggle {
            key = "pantalla";
            label = "Datos por Pantalla";
            mnemonic = "D";
        }
        : toggle {
            label = "Cuadro de Datos";
            key = "cuad";
            mnemonic = "C";
        }
    }
}
: row {
    : button {
        label = "Archivo...";
        key = "archivo";
        mnemonic = "A";
        fixed_width = true;
        is_enabled = false;
     }
     : edit_box {
         key = "path";
         width = 40;
         alignment = left;
         is_enabled = false;
```

```
            allow_accept = true;
        }
    }

    spacer;
    ok_cancel_help_errtile;
}
```

LISTADO 3.3. La rutina DMUELLE.LSP

```
;=============================================================================
;        NOMBRE DE ARCHIVO:    DMUELLE.LSP - AutoCAD Versión 12
;                      Escrito por:    Renato Parodi
;=============================================================================


(DEFUN *error* (cad)
   (PRINC "ERROR: ") (PRINC cad) (TERPRI)
   (UNLOAD_DIALOG dcl_id)
   (restore_muelle)
   (PRIN1)
)
;-----------------------------------------------------------------------
(DEFUN ai_abort_muelle (app msg)
   (IF msg
       (ALERT (STRCAT "Error de aplicacion: "
                      app
                      "  \n\n  "
                      msg
                       " \n "
              )
       )
   )
   (EXIT)
)
(COND ( (NOT (FINDFILE "muelle.dcl"))
        (ai_abort_muelle "PROGRAMA MUELLE"
                  (STRCAT "No se encuentra el archivo MUELLE.DCL"
                          "\n Buscar el el directorio" " " (GETVAR "dwgprefix"))))
      ( (NOT (FINDFILE "muelle.lsp"))
        (ai_abort_muelle "PROGRAMA MUELLE"
                  (STRCAT "No se encuentra el archivo MUELLE.LSP"
                          "\n Buscar el el directorio" " " (GETVAR "dwgprefix"))))
)
;-----------------------------------------------------------------------
(DEFUN ok_real (var vartipo)
   (COND ((DISTOF var 2) var)
         (T (SET_TILE "error" (STRCAT  "Invalido valor " vartipo)) nil)
   )
)
;-----------------------------------------------------------------------
(DEFUN off_mosaico ()
   (MODE_TILE "dal" 1)  (MODE_TILE "dres" 1) (MODE_TILE "pso" 1)
   (MODE_TILE "ns" 1) (MODE_TILE "rgan" 1) (MODE_TILE "simple" 1)
   (MODE_TILE "rebajado" 1) (MODE_TILE "escuadra" 1) (MODE_TILE "gancho" 1)
   (MODE_TILE "x_pt" 1) (MODE_TILE "y_pt" 1) (MODE_TILE "pin" 1)
   (MODE_TILE "archivo" 0) (MODE_TILE "path" 0)
)
;-----------------------------------------------------------------------
(DEFUN on_mosaico ()
   (MODE_TILE "dal" 0)  (MODE_TILE "dres" 0) (MODE_TILE "pso" 0)
   (MODE_TILE "ns" 0) (MODE_TILE "rgan" 0) (MODE_TILE "simple" 0)
   (MODE_TILE "rebajado" 0) (MODE_TILE "escuadra" 0) (MODE_TILE "gancho" 0)
   (MODE_TILE "x_pt" 0) (MODE_TILE "y_pt" 0) (MODE_TILE "pin" 0)
   (MODE_TILE "archivo" 1) (MODE_TILE "path" 1)
)
;-----------------------------------------------------------------------
(DEFUN file_muelle ()
   (SET_TILE "error" "")
   (SETQ fich (GETFILED "Fichero de menu a cargar" "" "?" 14))
   (IF (= fich nil)
       (IF (SETQ fich (GET_TILE "path")) (SET_TILE "path" fich))
       (SET_TILE "path" fich)
   )
)
;-----------------------------------------------------------------------
(DEFUN path_muelle ()
   (SET_TILE "error" "")
```

```
   (SETQ fich (GET_TILE "path"))
  ; (SETQ arch (FINDFILE fich))
)
;-------------------------------------------------------------------------
(DEFUN mosaico_muelle ()
   (IF (= "0" (SETQ dato (GET_TILE "dato")))
       (on_mosaico) (off_mosaico)
   )
)
;-------------------------------------------------------------------------
(DEFUN l_pantalla ()
   (IF (= "1" (SETQ pantalla (GET_TILE "pantalla")))
       (PROGN
          (off_mosaico)
          (MODE_TILE "archivo" 1)  (MODE_TILE "path" 1)
          (MODE_TILE "dato" 1) (SET_TILE "dato" "0")
       )
       (PROGN
          (on_mosaico)
          (MODE_TILE "archivo" 0)  (MODE_TILE "path" 0)
          (MODE_TILE "dato" 0)
       )
   )
   (SETQ ok_para T)
)
;-------------------------------------------------------------------------
(DEFUN get_icono (ico)
   (START_IMAGE "icono")
   (FILL_IMAGE 0 0 (DIMX_TILE "icono") (DIMY_TILE "icono") -2)
   (SLIDE_IMAGE 0 0 (DIMX_TILE "icono") (DIMY_TILE "icono")
                (STRCAT "tesis(" ico ")")
   )
   (END_IMAGE)
)
;-------------------------------------------------------------------------
(DEFUN get_simple ()
   (get_icono (SETQ op "Simple"))
   (MODE_TILE "rgan" 1)
)
;-------------------------------------------------------------------------
(DEFUN get_rebajado ()
   (get_icono (SETQ op "Rebajado"))
   (MODE_TILE "rgan" 1)
)
;-------------------------------------------------------------------------
(DEFUN get_escuadra ()
   (get_icono (SETQ op "Escuadra"))
   (MODE_TILE "rgan" 1)
)
;-------------------------------------------------------------------------
(DEFUN get_esc-apla ()
   (get_icono (SETQ op "Aplan-es"))
   (MODE_TILE "rgan" 1)
)
;-------------------------------------------------------------------------
(DEFUN get_gancho ()
   (get_icono (SETQ op "Gancho"))
   (MODE_TILE "rgan" 0)
)
;-------------------------------------------------------------------------
(DEFUN get_x_muelle ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ x_pt (GET_TILE "x_pt")) "x cordenada")
       (SETQ pin (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
   )
)
;-------------------------------------------------------------------------
(DEFUN get_y_muelle ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ y_pt (GET_TILE "y_pt")) "y cordenada")
```

```
        (SETQ pin (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
    )
)
;-------------------------------------------------------------------------
(DEFUN get_dal ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:dal (GET_TILE "dal")) "del diametro del alambre")
        (SETQ dal (DISTOF g:dal))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_dres ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:dres (GET_TILE "dres")) "del diametro del resorte")
        (SETQ dres (DISTOF g:dres))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_pso ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:pso (GET_TILE "pso")) "del paso")
        (SETQ pso (DISTOF g:pso))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_ns ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:ns (GET_TILE "ns")) "del numero de espiras")
        (SETQ ns (ATOI g:ns))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_rgan ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:rgan (GET_TILE "rgan")) "del radio del gancho")
        (SETQ rgan (DISTOF g:rgan))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN setloc_muelle ()
    (IF (NOT fich)
        (SETQ fich ""))
    (SET_TILE "path" (SETQ g:fich fich))
    (IF pin
        (SETQ x_pt (RTOS (CAR pin) 2 4)
              y_pt (RTOS (CADR pin) 2 4)
        )
        (SETQ pin (LIST
                    (DISTOF (SETQ x_pt "0.0000"))
                    (DISTOF (SETQ y_pt "0.0000"))
                  )
        )
    )
    (SET_TILE "x_pt" x_pt)
    (SET_TILE "y_pt" y_pt)
    (IF (NOT dal)
        (SETQ dal 0.2))
    (SET_TILE "dal" (SETQ g:dal (RTOS dal 2)))
    (IF (NOT dres)
        (SETQ dres 0.7))
    (SET_TILE "dres" (SETQ g:dres (RTOS dres 2)))
    (IF (NOT pso)
```

```
        (SETQ pso 0.35))
    (SET_TILE "pso" (SETQ g:pso (RTOS pso 2)))
    (IF (NOT ns)
        (SETQ ns 5))
    (SET_TILE "ns" (SETQ g:ns (ITOA ns)))
    (IF (NOT rgan)
        (SETQ rgan 0.7))
    (SET_TILE "rgan" (SETQ g:rgan (RTOS rgan)))
    (IF (NOT op)
        (SETQ op "Simple"))
    (COND ( (= op "Simple")
            (SET_TILE "simple" "1") (get_icono "simple"))
          ( (= op "Rebajado")
            (SET_TILE "rebajado" "1") (get_icono "rebajado"))
          ( (= op "Escuadra")
            (SET_TILE "escuadra" "1") (get_icono "escuadra"))
          ( (= op "Gancho")
            (SET_TILE "gancho" "1") (get_icono "gancho") (MODE_TILE "rgan" 0))
    )
)

;-------------------------------------------------------------------------
(DEFUN verify_muelle ()
    (COND ( (AND (AND (SETQ ok_para T) (= nil (FINDFILE fich))) (= "1" dato))
            (MODE_TILE "path" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Archivo no encontrado"))
          ( (AND ok_para (/= (TYPE pin) 'LIST))
            (SETQ ok_para nil)
            (SET_TILE "error" "Invalido punto de inicio"))
          ( (AND ok_para (<= dal 0))
            (MODE_TILE "dal" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto"))
          ( (AND ok_para (<= dres 0))
            (MODE_TILE "dres" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto"))
          ( (AND ok_para (< pso dal))
            (MODE_TILE "pso" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto del paso (Mayor o igual al
D.Aambre)"))
          ( (AND ok_para (<= ns 1))
            (MODE_TILE "ns" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto (2 o mas)"))
          ( (AND (= op "Gancho") (AND ok_para (< rgan (/ dres 2.0))))
            (MODE_TILE "rgan" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto del gancho (Mayor o = al R.Resorte)"))
          ( (AND ok_para (NOT (ok_real g:dal "del diametro del alambre.")))
            (MODE_TILE "dal" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real g:dres "del diametro del resorte.")))
            (MODE_TILE "dres" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real g:pso "del paso.")))
            (MODE_TILE "pso" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real g:ns "del numero de espiras.")))
            (MODE_TILE "ns" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real g:rgan "del radio del gancho.")))
            (MODE_TILE "dal" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real x_pt "X cordenada.")))
            (MODE_TILE "x_pt" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real y_pt "Y cordenada.")))
            (MODE_TILE "y_pt" 2) (SETQ ok_para nil))
    )
)
;-------------------------------------------------------------------------
(DEFUN restore_muelle ()
    (COMMAND "layer" "s" oldlayer "")
    (COMMAND "color" oldcolor)
    (SETVAR "blipmode" oldblp)
    (SETVAR "cmdecho" oldech)
)
;-------------------------------------------------------------------------
(DEFUN c:muelle ( / pantalla fich dato cuad)
    (SETQ oldblp (GETVAR "blipmode")
          oldech (GETVAR "cmdecho")
```

```
        oldlayer (GETVAR "clayer")
        oldcolor (GETVAR "cecolor")
    )
    (SETVAR "blipmode" 0) (SETVAR "cmdecho" 0)
    (SETQ what_next 5)
    (SETQ dcl_id (LOAD_DIALOG "muelle"))
    (WHILE (< 2 what_next)
       (IF (NOT (NEW_DIALOG "muelle" dcl_id)) (EXIT))
       (setloc_muelle)
       (ACTION_TILE "archivo" "(file_muelle)")
       (ACTION_TILE "path" "(path_muelle)")
       (ACTION_TILE "simple" "(get_simple)")
       (ACTION_TILE "rebajado" "(get_rebajado)")
       (ACTION_TILE "escuadra" "(get_escuadra)")
       (ACTION_TILE "esc-apla" "(get_esc-apla)")
       (ACTION_TILE "gancho" "(get_gancho)")
       (ACTION_TILE "pin" "(done_dialog 4)")
       (ACTION_TILE "x_pt" "(get_x_muelle)")
       (ACTION_TILE "y_pt" "(get_y_muelle)")
       (ACTION_TILE "dal" "(get_dal)")
       (ACTION_TILE "dres" "(get_dres)")
       (ACTION_TILE "pso" "(get_pso)")
       (ACTION_TILE "ns" "(get_ns)")
       (ACTION_TILE "rgan" "(get_rgan)")
       (ACTION_TILE "dato" "(mosaico_muelle)")
       (ACTION_TILE "pantalla" "(l_pantalla)")
       (ACTION_TILE "cuad" "(setq cuad $value)")
       (ACTION_TILE "accept" "(verify_muelle) (if ok_para (done_dialog 1))")
       (SETQ what_next (START_DIALOG))
       (COND ( (= what_next 4)
               (SETQ pin (GETPOINT "P. de inicio del alambre: ")) (TERPRI))
             ( (AND (= what_next 1) (= dato "1"))
               (LOAD "muelle") (para_file) (simp)
               (IF (= cuad "1") (cuadro)))
             ( (AND (= what_next 1) (= "1" pantalla))
               (LOAD "muelle") (inic_muelle) (simp)
               (IF (= cuad "1") (cuadro)))
             ( (= what_next 1)
               (LOAD "muelle") (simp)
               (IF (= cuad "1") (cuadro)))
       )
    )
    (UNLOAD_DIALOG dcl_id)
    (restore_muelle)
    (PRIN1)
)
(PRINC "     DMUELLE cargado.    ")
(PRINC)
```

LISTADO 4.1 La rutina LEVA.LSP

```
;=====================================================================
;          NOMBRE DE ARCHIVO:  LEVA.LSP - AutoCAD Versión 12
;                      Escrito por:  Renato Parodi
;=====================================================================

(DEFUN inic_leva ()
    (INITGET 1)
    (SETQ pcen (GETPOINT "Punto de inicio de la leva: ")) (TERPRI)
    (INITGET 6 "Si No")
    (IF (SETQ op (GETKWORD "Graficar el chavetero No/<Si>: "))
        () (SETQ op "Si")
    )
    (IF (= op "Si")
      (PROGN
        (INITGET 7)
        (SETQ dia (GETREAL "\nDiametro del eje: "))
        (COND ( (< dia 6)
                  (SETQ b 2.0) (SETQ t1 0.6)
                )
                ( (AND (>= dia 6) (< dia 8))
                  (SETQ b 2.0) (SETQ t1 0.6)
                )
                ( (AND (>= dia 8) (< dia 10))
                  (SETQ b 3.0) (SETQ t1 1.4)
                )
                ( (AND (>= dia 10) (< dia 12))
                  (SETQ b 4.0) (SETQ t1 1.8)
                )
                ( (AND (>= dia 12) (< dia 17))
                  (SETQ b 5.0) (SETQ t1 2.3)
                )
                ( (AND (>= dia 17) (< dia 22))
                  (SETQ b 6.0) (SETQ t1 2.8)
                )
                ( (AND (>= dia 22) (< dia 30))
                  (SETQ b 8.0) (SETQ t1 3.1)
                )
                ( (AND (>= dia 30) (< dia 38))
                  (SETQ b 10.0) (SETQ t1 3.3)
                )
                ( (AND (>= dia 38) (< dia 44))
                  (SETQ b 12.0) (SETQ t1 3.6)
                )
                ( (AND (>= dia 44) (< dia 50))
                  (SETQ b 14.0) (SETQ t1 3.8)
                )
                ( (AND (>= dia 50) (< dia 58))
                  (SETQ b 16.0) (SETQ t1 4.3)
                )
                ( (AND (>= dia 58) (< dia 65))
                  (SETQ b 18.0) (SETQ t1 4.4)
                )
                ( (AND (>= dia 65) (< dia 75))
                  (SETQ b 20.0) (SETQ t1 4.9)
                )
                ( (AND (>= dia 75) (< dia 85))
                  (SETQ b 22.0) (SETQ t1 5.4)
                )
                ( T
                  (SETQ b 22.0) (SETQ t1 5.4)
                )
        )
      )
    )
)
;---------------------------------------------------------------------
(DEFUN lptos ()
```

```
    (SETQ xo (CAR pcen) yo (CADR pcen))
    (SETQ arch (OPEN fich "r"))
    (SETQ n 1 r T)
        (SETQ cad (READ-LINE arch))
        (SETQ x (+ (ATOF (SUBSTR cad 2 24)) xo))
        (SETQ y (+ (ATOF (SUBSTR cad 29)) yo))
        (SETQ p (LIST x y))
        (COMMAND "pline" p)
    (WHILE r
        (SETQ cad (READ-LINE arch))
        (IF (/= cad nil)
            (PROGN
               (SETQ x (+ (ATOF (SUBSTR cad 2 24)) xo))
               (SETQ y (+ (ATOF (SUBSTR cad 29)) yo))
               (SETQ p (LIST x y))
               (COMMAND p)
               (SETQ n (+ n 1))
            )
            (SETQ r nil)
        )
    )
    (COMMAND "c")
    (CLOSE arch)
)
;-------------------------------------------------------------------------
(DEFUN chavet ()
    (IF (OR (= op "Si") (= chavetero "1"))
        (PROGN
           (SETQ h (SQRT (- (EXPT (/ dia 2.0) 2) (EXPT (/ b 2.0) 2))))
           (SETQ pa (POLAR pcen (/ PI 2) h))
           (SETQ p1 (POLAR pa PI (/ b 2.0)))
           (SETQ p2 (POLAR p1 (/ PI 2) t1))
           (SETQ p3 (POLAR p2 0 b))
           (SETQ p4 (POLAR p3 (- (/ PI 2)) t1))
           (COMMAND "pline" p1 "a" "ce" pcen p4 "l" p3 p2 p1 "")
        )
    )
)
```

LISTADO 4.2 Un archivo DCL para DLEVA.LSP

```
//=============================================================================
//                LEVA.DCL - Cuadro de dialogo DLEVA.DCL
//                     Escrito por: Renato Parodi
//=============================================================================

leva    : dialog {
    label = "Parametros de Leva";
    : boxed_column {
        label = "Seleccione el Nombre del Archivo";
        : row {
            : button {
                label = "Archivo...";
                key = "archivo";
                mnemonic = "A";
                width = 10;
                fixed_width = true;
            }
            : edit_box {
                key = "path";
                width = 36;
            }
        }
    }
    spacer;
    spacer;
    : boxed_column {
        label = "Opciones";
        : row {
            : toggle {
                label = "Ingreso de Datos";
                key = "pantalla";
                mnemonic = "I";
                fixed_width = true;
            }
        }
        spacer;
        spacer;
        : row {
            : boxed_column {
                label = "Centro Leva";
                :button {
                   key = "pcen";
                   label = "Pto Desig. <";
                   mnemonic = "P";
                   fixed_width = true;
                   alignment = left;
                   is_enabled = false;
                }
                : edit_box {
                    label = "X:";
                    mnemonic ="X";
                    key = "x_pt";
                    fixed_width = true;
                    is_enabled = false;
                }
                : edit_box {
                    label ="Y:";
                    mnemonic ="Y";
                    key = "y_pt";
                    fixed_width = true;
                    is_enabled = false;
                }
            }
            : boxed_column {
                label = "Diametro del Arbol";
                : toggle {
                    label = "Chavetero";
```

```
                    key = "chavetero";
                    mnemonic = "C";
                    fixed_width = true;
                }
                spacer;
                : edit_box {
                    label = "Dtro del Arbol:";
                    mnemonic ="D";
                    key = "dia";
                    fixed_width = true;
                    is_enabled = false;
                }
                : popup_list {
                    label = "Chaveta: ";
                    key = "chave";
                    mnemonic = "h";
                    edit_width = 7;
                    is_enabled = false;
                }
            }
        }
    }
    ok_cancel_help_errtile;
}
```

LISTADO 4.3 La rutina DLEVA.LSP

```
;=============================================================================
;          NOMBRE DE ARCHIVO:  DLEVA.LSP - AutoCAD Version 12
;                    Escrito por:     Renato Parodi
;=============================================================================


(DEFUN *error* (cad)
  (PRINC "Error: ") (PRINC cad) (TERPRI)
  (UNLOAD_DIALOG dcl_id)
  (restore_leva)
  (PRIN1)
)
;-----------------------------------------------------------------------
(DEFUN ai_abort_leva (app msg)
   (IF msg
       (ALERT (STRCAT "Error de aplicación: "
                      app
                     "   \n \n  "
                      msg
                     "  \n  "
              )
       )
   )
   (EXIT)
)
(COND ( (NOT (FINDFILE "leva.lsp"))
        (ai_abort_leva "PROGRAMA LEVA"
                  (STRCAT "No se encuentra el archivo LEVA.LSP"
                          "\n  Buscar en el directorio" " " (GETVAR "dwgprefix"))))
      ( (NOT (FINDFILE "leva.dcl"))
        (ai_abort_leva "LEVA"
                  (STRCAT "No se encuentra el archivo LEVA.DCL"
                          "\n  Buscar en el directorio" " " (GETVAR "dwgprefix"))))
)
;-----------------------------------------------------------------------
(DEFUN ok_real (var vartipo)
   (COND ((DISTOF var 2) var)
         (T (SET_TILE "error" (STRCAT  "Invalido valor " vartipo)) nil)
   )
)
;-----------------------------------------------------------------------
(DEFUN file_leva ()
   (SET_TILE "error" "")
   (SETQ fich (GETFILED "Fichero de menu a cargar" "" "LEV" 10))
   (IF (= fich nil)
       (PROGN
          (IF (SETQ fich (GET_TILE "path")) (SET_TILE "path" fich) (SET_TILE "path"
"" ))
       )
          (SET_TILE "path" fich)
   )
)
;-----------------------------------------------------------------------
(DEFUN mosaico_leva ()
   (IF (= "0" (SETQ pantalla (GET_TILE "pantalla")))
       (PROGN
          (MODE_TILE "x_pt" 0) (MODE_TILE "y_pt" 0)
          (MODE_TILE "pcen" 0) (MODE_TILE "chavetero" 0)
       )
       (PROGN
          (MODE_TILE "x_pt" 1) (MODE_TILE "y_pt" 1)
          (MODE_TILE "pcen" 1) (MODE_TILE "chavetero" 1)
          (MODE_TILE "dia" 1) (MODE_TILE "chave" 1)
          (SET_TILE "chavetero" "0")
       )
   )
)
;-----------------------------------------------------------------------
```

```
(DEFUN mosa_leva ()
   (COND ( (= "1" (SETQ chavetero (GET_TILE "chavetero")))
           (MODE_TILE "dia" 0) (MODE_TILE "chave" 0))
         ( T
           (MODE_TILE "dia" 1) (MODE_TILE "chave" 1))
   )
)
;-------------------------------------------------------------------------
(DEFUN path_leva ()
   (SET_TILE "error" "")
   (SETQ fich (GET_TILE "path"))
   (COND ( (WCMATCH fich "*.lev,*.LEV")
           (SETQ g:fich (FINDFILE fich))
         )
         ( T
           (SETQ fich (STRCAT fich ".LEV"))
           (SETQ g:fich (FINDFILE fich))
         )
   )
)
;-------------------------------------------------------------------------
(DEFUN get_chave_leva ()
   (SETQ list1 (LIST " 6 - 8" " 8 - 10" "10 - 12" "12 - 17" "17 - 22"
                     "22 - 30" "30 - 38" "38 - 44" "44 - 50"
                     "50 - 58" "58 - 65" "65 - 75" "75 - 85"))
   (START_LIST "chave")
   (MAPCAR 'ADD_LIST list1)
   (END_LIST)
)
;-------------------------------------------------------------------------
(DEFUN get_x_leva ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ x_pt (GET_TILE "x_pt")) "x cordenada")
       (SETQ pcen (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
   )
   nil
)
;-------------------------------------------------------------------------
(DEFUN get_y_leva ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ y_pt (GET_TILE "y_pt")) "y cordenada")
       (SETQ pcen (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
   )
   nil
)
;-------------------------------------------------------------------------
(DEFUN get_dia ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ g:dia (GET_TILE "dia")) "del diametro")
       (SETQ dia (DISTOF g:dia))
   )
   (COND ( (>= dia 85)
           (SET_TILE "chave" (SETQ chave "12")))
         ( (<= dia 8)
           (SET_TILE "chave" (SETQ chave "0")))
         ( (AND (> dia 8) (< dia 85))
           (SETQ vmin 6 vmax 8)
           (SETQ n 0)
           (WHILE (NOT (AND (>= dia vmin) (<= dia vmax)))
              (SETQ vmin (ATOI (SUBSTR (NTH n list1) 1 3)))
              (SETQ vmax (ATOI (SUBSTR (NTH n list1) 5)))
              (SETQ n (+ n 1))
           )
           (SET_TILE "chave" (SETQ chave (ITOA (- n 1)))))
   )
)
;-------------------------------------------------------------------------
(DEFUN setloc_leva ()
   (IF (NOT fich)
```

```
        (SETQ fich ""))
    (SET_TILE "path" (SETQ g:fich fich))
    (IF pcen
        (SETQ x_pt (RTOS (CAR pcen) 2 4)
              y_pt (RTOS (CADR pcen) 2 4)
        )
        (SETQ pcen (LIST
                     (DISTOF (SETQ x_pt "0.0000"))
                     (DISTOF (SETQ y_pt "0.0000"))
                   )
        )
    )
    (SET_TILE "x_pt" x_pt)
    (SET_TILE "y_pt" y_pt)
    (IF (NOT pantalla)
        (SETQ pantalla "0"))
    (SET_TILE "pantalla" (SETQ g:pantalla pantalla))
    (IF (NOT dia)
        (SETQ dia 6.0000))
    (SET_TILE "dia" (SETQ g:dia (RTOS dia 2)))
    (IF (NOT chavetero)
        (SETQ chavetero "0"))
    (SET_TILE "chavetero" (SETQ g:chavetero chavetero))
    (COND ( (= "0" pantalla)
            (MODE_TILE "pcen" 0) (MODE_TILE "x_pt" 0)
            (MODE_TILE "y_pt" 0) (MODE_TILE "chaaveta" 0))
          ( T
            (MODE_TILE "pcen" 1) (MODE_TILE "x_pt" 1)
            (MODE_TILE "y_pt" 1) (MODE_TILE "chaveta" 1))
    )
    (COND ( (= "1" chavetero)
            (MODE_TILE "dia" 0) (MODE_TILE "chave" 0))
          ( T
            (MODE_TILE "dia" 1) (MODE_TILE "chave" 1))
    )
)
;-------------------------------------------------------------------------
(DEFUN verify_leva ()
    (COND ( (AND (SETQ ok_para T) (OR (= fich "") (= fich ".LEV")))
            (MODE_TILE "path" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Invalida especificacion de archivo"))
          ( (AND SETQ ok_para (= nil (FINDFILE fich)))
            (MODE_TILE "path" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Archivo no encontrado"))
          ( (AND ok_para (/= (TYPE pcen) 'LIST))
            (MODE_TILE "path" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Invalido punto central"))
          ( (AND ok_para (< dia 6))
            (MODE_TILE "dia" 2) (SETQ ok_para nil)
            (SET_TILE "error" "Valor incorrecto (6 o mas)"))
          ( (AND ok_para (NOT (ok_real g:dia "del diametro.")))
            (MODE_TILE "dia" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real x_pt "Y cordenada.")))
            (MODE_TILE "x_pt" 2) (SETQ ok_para nil))
          ( (AND ok_para (NOT (ok_real y_pt "Y cordenada.")))
            (MODE_TILE "y_pt" 2) (SETQ ok_para nil))
    )
)
;-------------------------------------------------------------------------
(DEFUN restore_leva ()
    (SETVAR "blipmode" oldblp)
    (SETVAR "cmdecho" oldech)
)
;-------------------------------------------------------------------------
(DEFUN c:leva ( / ok_para pcen x_pt y_pt fich chavetero pantalla dia)
    (SETQ oldblp (GETVAR "blipmode")
          oldech (GETVAR "cmdecho")
    )
    (SETVAR "blipmode" 0) (SETVAR "cmdecho" 0)
    (SETQ what_next 5)
    (SETQ dcl_id (LOAD_DIALOG "leva"))
    (WHILE (< 2 what_next)
```

```
    (IF (NOT (NEW_DIALOG "leva" dcl_id)) (EXIT))
    (get_chave_leva)
    (setloc_leva)
    (ACTION_TILE "archivo" "(file_leva)")
    (ACTION_TILE "path" "(path_leva)")
    (ACTION_TILE "pantalla" "(mosaico_leva)")
    (ACTION_TILE "chavetero" "(mosa_leva)")
    (ACTION_TILE "pcen" "(done_dialog 4)")
    (ACTION_TILE "x_pt" "(get_x_leva)")
    (ACTION_TILE "y_pt" "(get_y_leva)")
    (ACTION_TILE "dia" "(get_dia)")
    (ACTION_TILE "chave" "(SETQ chave $value)")
    (ACTION_TILE "accept" "(verify_leva) (if ok_para (done_dialog 1))")
    (SETQ what_next (START_DIALOG))
    (COND ( (= what_next 4)
            (SETQ pcen (GETPOINT "P. CENTRO: ")) (TERPRI))
          ( (AND (= what_next 1) (= "1" chavetero))
            (COND ( (= chave "0") (SETQ b 2.0 t1 0.6))
                  ( (= chave "1") (SETQ b 3.0 t1 1.4))
                  ( (= chave "2") (SETQ b 4.0 t1 1.8))
                  ( (= chave "3") (SETQ b 5.0 t1 2.3))
                  ( (= chave "4") (SETQ b 6.0 t1 2.8))
                  ( (= chave "5") (SETQ b 8.0 t1 3.1))
                  ( (= chave "6") (SETQ b 10.0 t1 3.3))
                  ( (= chave "7") (SETQ b 12.0 t1 3.6))
                  ( (= chave "8") (SETQ b 14.0 t1 3.8))
                  ( (= chave "9") (SETQ b 16.0 t1 4.3))
                  ( (= chave "10") (SETQ b 18.0 t1 4.4))
                  ( (= chave "11") (SETQ b 20.0 t1 4.5))
                  ( (= chave "12") (SETQ b 22.0 t1 5.4))
            )
            (LOAD "leva") (lptos) (chavet))
          ( (AND (= what_next 1) (= "0" pantalla))
            (LOAD "leva") (lptos))
          ( (= what_next 1)
            (LOAD "leva") (inic_leva) (lptos) (chavet))
    )
  )
  (UNLOAD_DIALOG dcl_id)
  (restore_leva)
  (PRIN1)
)
(PRINC "    DLEVA cargado    ")
(PRINC)
```

LISTADO 5.1 La rutina ENGRANE.LSP

```
;=============================================================================
;        NOMBRE DEL ARCHIVO:     ENGRANE.LSP - AutoCAD Version 12
;                     Escrito por:    Renato Parodi
;=============================================================================

(DEFUN rad_deg (rad)
   (/ (* rad 180) PI)
)
(DEFUN inic_engrane ()
    (GRAPHSCR)
    (PROMPT "Este programa dibuja engranajes") (TERPRI)
    (INITGET 1)
    (SETQ pcen (GETPOINT "Punto inicial del pinon: ")) (TERPRI)
    (INITGET 7)
    (SETQ mod  (GETREAL "Entre el modulo: ")) (TERPRI)
    (INITGET 7)
    (SETQ nd (GETINT "Numero de dientes: ")) (TERPRI)
    (INITGET 2)
    (IF (SETQ angp (GETANGLE "Angulo de presion <25°>: ")) ()
        (SETQ angp 0.436332)) (TERPRI)
    (INITGET 6 "Si No")
    (IF (SETQ opl (GETKWORD "Greficar el chavetero No/<Si>: ")) ()
        (SETQ opl "Si")) (TERPRI)
    (IF (= opl "Si")
      (PROGN
        (INITGET 7)
        (SETQ diam (GETREAL "Diametro del eje: ")) (TERPRI)
        (SETQ di (* mod (- nd 2.334)))
        (WHILE (> diam di)
          (PRINC (STRCAT "*Error* El diametro del eje es mayor al\n"
                         "        diametro interior del engranaje\n")
          )
          (INITGET 7)
          (SETQ diam (GETREAL "Diametro del eje: ")) (TERPRI)
        )
        (COND ( (< diam 6)
                (SETQ b 2.0) (SETQ t1 0.6)
              )
              ( (AND (>= diam 6) (< diam 8))
                (SETQ b 2.0) (SETQ t1 0.6)
              )
              ( (AND (>= diam 8) (< diam 10))
                (SETQ b 3.0) (SETQ t1 1.4)
              )
              ( (AND (>= diam 10) (< diam 12))
                (SETQ b 4.0) (SETQ t1 1.8)
              )
              ( (AND (>= diam 12) (< diam 17))
                (SETQ b 5.0) (SETQ t1 2.3)
              )
              ( (AND (>= diam 17) (< diam 22))
                (SETQ b 6.0) (SETQ t1 2.8)
              )
              ( (AND (>= diam 22) (< diam 30))
                (SETQ b 8.0) (SETQ t1 3.1)
              )
              ( (AND (>= diam 30) (< diam 38))
                (SETQ b 10.0) (SETQ t1 3.3)
              )
              ( (AND (>= diam 38) (< diam 44))
                (SETQ b 12.0) (SETQ t1 3.6)
              )
              ( (AND (>= diam 44) (< diam 50))
                (SETQ b 14.0) (SETQ t1 3.8)
              )
              ( (AND (>= diam 50) (< diam 58))
                (SETQ b 16.0) (SETQ t1 4.3)
```

```
                )
                ( (AND (>= diam 58) (< diam 65))
                  (SETQ b 18.0) (SETQ t1 4.4)
                )
                ( (AND (>= diam 65) (< diam 75))
                  (SETQ b 20.0) (SETQ t1 4.9)
                )
                ( (AND (>= diam 75) (< diam 85))
                  (SETQ b 22.0) (SETQ t1 5.4)
                )
                ( T
                  (SETQ b 22.0) (SETQ t1 5.4)
                )
            )
        )
    )
)
;-------------------------------------------------------------------------
(DEFUN cal ( )
    (SETQ rad (* 0.3 mod))
    (SETQ di (* mod (- nd 2.334)))
    (SETQ dp (* mod nd))
    (SETQ de (* mod (+ nd 2)))
    (SETQ db (* dp (COS angp)))
    (SETQ do (+ di (* 0.334 mod)))
    (SETQ paso (* mod PI))
    (SETQ hdie (* 2.167 mod))
    (SETQ hcab mod)
    (SETQ hpie (* 1.167 mod))
    (SETQ disd (* 1.5708 mod))
    (SETQ espd disd)
)
;-------------------------------------------------------------------------
(DEFUN puntos ()
    (SETQ prx (/ (- de db) 10.0))
    (IF (>= db di) (SETQ dia db) (SETQ dia do db do))
    (SETQ dang1 (/ (SQRT (- (EXPT dp 2) (EXPT db 2))) db))
    (SETQ r 1  n 1 dia db)
    (WHILE r
      (SETQ dang2 (/ (SQRT (- (EXPT dia 2) (EXPT db 2))) db))
      (SETQ ang (- (+ (/ 1.5708 nd) dang1 (ATAN dang2)) (+ (ATAN dang1) dang2)))
      (SETQ expr1 (STRCAT "(SETQ p"(ITOA n)" (POLAR pcen (+ (- ang) ang1) (/ dia
2.0)))"))
        (SETQ expr2 (STRCAT "(SETQ q"(ITOA n)" (POLAR pcen (+ ang ang1)(/ dia
2.0)))"))
        (EVAL (READ expr1))
        (EVAL (READ expr2))
        (IF (= n 1) (SETQ ang0 ang))
        (SETQ dia (+ dia prx))
        (SETQ n (+ n 1))
        (IF (>= dia de) (SETQ r nil))
    )
    (SETQ pa (POLAR pcen (ANGLE pcen p1) (/ do 2.0)))
    (SETQ pb (POLAR pcen (ANGLE pcen q1) (/ do 2.0)))
    (SETQ cen (POLAR pa (+ (- 1.571) ang1) rad))
    (SETQ cen1 (POLAR pb (+ 1.571 ang1) rad))
    (SETQ r 1 ) (SETQ ang (ANGLE cen pa))
    (WHILE r
        (SETQ pf (POLAR cen ang rad))
        (SETQ x (CAR pf) y (CADR pf))
        (SETQ ra (SQRT (+ (EXPT (- x (CAR pcen)) 2) (EXPT (- y (CADR pcen)) 2))))
        (SETQ ang (+ ang 0.1))
        (IF (<= ra (/ di 2.0)) (SETQ r nil))
    )
    (SETQ pf2 (POLAR cen (- ang 0.275) rad))
    (SETQ pf3 (POLAR cen1 (- (ANGLE cen1 pb) 0.275) rad))
    (SETQ ang2 (+ ang0 (- (/ PI nd) ang0 )))
    (SETQ pin (POLAR pcen (+ (- ang2) ang1) (/ di 2.0)))
    (SETQ ang3 (- (/ PI nd) (- (ANGLE pcen pf) (ANGLE pcen pin))))
    (SETQ pf1 (POLAR pcen (+ (ANGLE pcen pf) (* 2 ang3)) (/ di 2.0)))
    (SETQ pin1 (POLAR pcen (+ ang2 ang1) (/ di 2.0)))
```

```
)
;-------------------------------------------------------------------------

(DEFUN envol ()
    (SETQ r 3 s (- n 3) expr1 "p2" expr2 (STRCAT "q"(ITOA (- n 2))))
    (REPEAT (- n 3)
        (SETQ expr1 (STRCAT expr1 " " "p"(ITOA r)))
        (SETQ expr2 (STRCAT expr2 " " "q"(ITOA s)))
        (SETQ s (- s 1))
        (SETQ r (+ r 1))
    )
    (SETQ exp (STRCAT "(COMMAND "(CHR 34) "pline" (CHR 34)" pin "(CHR 34)
        "a" (CHR 34)" "(CHR 34) "ce" (CHR 34)" pcen pf "(CHR 34)
        "s" (CHR 34)" pf2 pa "(CHR 34) "1" (CHR 34)" "expr1" "(CHR 34)
        "a" (CHR 34)" "(CHR 34) "ce" (CHR 34)" pcen q"(ITOA (- n 1))" "(CHR 34)
        "1" (CHR 34)" "expr2" pb "(CHR 34) "a" (CHR 34)" "(CHR 34)
        "s" (CHR 34)" pf3 pf1 "(CHR 34) "ce" (CHR 34)" pcen pin1 "(CHR 34) (CHR
34)")"))
    (EVAL (READ exp))
)
;-------------------------------------------------------------------------
(DEFUN dib ()
    (SETQ angl 0)
    (REPEAT nd
        (cal)
        (puntos)
        (envol)
        (SETQ angl (+ angl (/ (* 2 PI) nd)))
    )
)
;-------------------------------------------------------------------------

(DEFUN nulo ()
    (SETQ r 1 s 1)
    (REPEAT n
        (SETQ expr1 (STRCAT "(SETQ p"(ITOA r)" nil)"))
        (SETQ expr2 (STRCAT "(SETQ q"(ITOA s)" nil)"))
        (EVAL (READ expr1))
        (EVAL (READ expr2))
        (SETQ r (+ r 1) s (+ s 1))
    )
)
;-------------------------------------------------------------------------
(DEFUN chavet ()
    (IF (OR (= op1 "Si") (= chavetero "1"))
        (PROGN
           (SETQ h (SQRT (- (EXPT (/ diam 2.0) 2) (EXPT (/ b 2.0) 2))))
           (SETQ pa (POLAR pcen (/ PI 2.0) h))
           (SETQ p1 (POLAR pa PI (/ b 2.0)))
           (SETQ p2 (POLAR p1 (/ PI 2.0) t1))
           (SETQ p3 (POLAR p2 0 b))
           (SETQ p4 (POLAR p3 (- (/ PI 2.0)) t1))
           (COMMAND "pline" p1 "a" "ce" pcen p4 "1" p3 p2 p1 "")
        )
    )
)
;-------------------------------------------------------------------------
(DEFUN texto (tx / )
    (COMMAND "text" pins (* 0.015 escl) 0 tx)
    (SETQ pins (LIST (CAR pins) (- (CADR pins) (* 0.04 escl))))
)
;-------------------------------------------------------------------------


(DEFUN cuadro ()
   (IF (= cua "1")
       (PROGN
          (SETQ p1 (GETPOINT "Punto de insercion del cuadro: ")) (TERPRI)
          (SETQ limt (CAR (GETVAR "limmax")))
          (SETQ esc de)
          (IF (SETQ escl (GETREAL (STRCAT "Escala del cuadro <"(RTOS esc 2 2)">: ")))
              () (SETQ escl esc)) (TERPRI)
```

```
            (COMMAND "insert" "format" p1 1 "" 0)
            (COMMAND "insert" "*cuadro1" p1 esc1 "")
            (COMMAND "layer" "s" "formato" "")
            (COMMAND "color" 1)
            (SETQ pins (POLAR p1 0 (* 0.37 esc1)))
            (texto (RTOS mod 2 2))
            (texto (RTOS nd 2 2))
            (texto (STRCAT (RTOS dp 2 2) "" "mm"))
            (texto (STRCAT (RTOS paso 2 2) "mm"))
            (texto (STRCAT (RTOS (rad_deg angp) 2 2) "" "%%D"))
            (texto (STRCAT (RTOS hdie 2 2) "" "mm"))
            (texto (STRCAT (RTOS hpie 2 2) "" "mm"))
            (texto (STRCAT (RTOS rad 2 2) "" "mm"))
        )
    )
)
;=============================================================================
(DEFUN list_resultado ()
   (SETQ list1 '("\t   Modulo" "\t   Angulo de Presion" "\t   Numero de Dientes" "\t
Paso"
                 "\t   Diametro Interior" "\t   Diametro Primitivo" "\t   Diametro
Base" "\t   Diametro Exterior"
                 "\t   Espacio entre Diente" "\t   Espesor del Diente" "\t   Radio de
Entalle"
                 "\t   Adendo" "\t   Dedendo" "\t   Altura del Diente"))
   (SETQ list2 '("\t\t=" "\t=" "\t=" "\t\t\t=" "\t=" "\t="
                 "\t=" "\t=" "\t=" "\t=" "\t=" "\t\t="
                 "\t\t=" "\t="))
   (SETQ list3 '("\t" "\t" "\t" "\t" "\t" "\t" "\t" "\t" "\t" " \t" "\t"
                 "\t" "\t" "\t"))
   (SETQ list4 (LIST (RTOS mod 2 2) (RTOS (rad_deg angp) 2 2) (RTOS nd 2 2)
                     (RTOS paso 2 2) (RTOS di 2 2) (RTOS dp 2 2) (RTOS db 2 2)
                     (RTOS de 2 2) (RTOS disd 2 2) (RTOS espd 2 2) (RTOS rad 2 2)
                     (RTOS hpie 2 2) (RTOS hcab 2 2) (RTOS hdie 2 2)))
   (SETQ list5 '("\n" "  grados\n" "  diente\n" "  mm\n" "  mm\n"
                 "  mm\n" "  mm\n" "  mm\n" "  mm\n" "  mm\n" "  mm\n"
                 "  mm\n" "  mm\n" "  mm\n"))
  (IF (= rea "1")
      (PROGN
        (SETQ file (OPEN narch "w"))
        (SETQ ldato (MAPCAR 'strcat list1 list2 list3 list4 list5))
        (PRINC "\tTABLA DE RESULTADOS DE LOS PARAMETROS DEL ENGRANAJE\n" file)
        (PRINC "\t=================================================\n\n" file)
        (MAPCAR '(LAMBDA (lista) (PRINC lista file)) ldato)
        (PRINC "\t=================================================\n\n" file)
        (CLOSE file)
      )
   )
   (IF (= rep "1")
      (PROGN
        (TEXTPAGE)
        (PRINC "\tTABLA DE RESULTADOS DE LOS PARAMETROS DEL ENGRANAJE\n")
        (PRINC "\t=================================================\n\n")
        (MAPCAR 'princ (MAPCAR 'strcat list1 list2 list3 list4 list5))
        (PRINC "\t=================================================\n\n")
      )
   )
)
```

LISTADO 5.2 Un archivo DCL para DENGRANE.LSP

```
//=============================================================================
//                ENGRANE.DCL - Cuadro de dialogo DENGRANE.LSP
//                      Escrito por:  Renato Parodi
//=============================================================================

engrane  : dialog {
    label = "Parametros del Engranaje";
  : boxed_row {
     label = "Opciones";
     : column {
        : toggle {
           label = "Resultados en Archivo";
           key = "rea";
           mnemonic = "R";
        }
        : edit_box {
           label = "File:";
           mnemonic ="F";
           key = "file";
           fixed_width = true;
           width = 25;
           is_enabled = false;
           edit_limit = 12;
        }
     }
     : column {
        : toggle {
           label = "Resultados en Pantalla";
           key = "rep";
           mnemonic = "P";
        }
        : toggle {
           label = "Cuadro de Corte";
           key = "cua";
           mnemonic = "C";
        }
     }
  }
  : boxed_column {
     : toggle {
        label = "Ingreso de Datos";
        key = "pantalla";
        mnemonic = "I";
        fixed_width = true;
     }
  : row {
      : boxed_column {
      label = "Centro Engrane";
          : button {
              key = "pcen";
              label = "Pto Desig. <";
              mnemonic = "P";
              fixed_width = true;
              alignment = left;
          }
          : edit_box {
              label = "X:";
              mnemonic ="X";
              key = "x_pt";
              fixed_width = true;
          }
          : edit_box {
              label ="Y:";
              mnemonic ="Y";
              key = "y_pt";
              fixed_width = true;
          }
       }
```

```
      spacer_1;
      : boxed_column {
        label = "Geom. del Diente";
          : edit_box {
              label ="Módulo:";
              mnemonic = "M";
              key = "mod";
              fixed_width = true;
          }
          : edit_box {
              label = "Dientes:";
              mnemonic = "D";
              key = "nd";
              fixed_width = true;
          }
          : popup_list {
              label = "Angulo";
              key = "angp";
              mnemonic = "A";
              edit_width = 7;
          }
      }
      : boxed_column {
        label = "Diam. del Arbol";
          : toggle {
              label = "Chavetero";
              key = "chavetero";
              mnemonic = "C";
              fixed_width = true;
          }
          : edit_box {
              label = "Diametro:";
              mnemonic ="D";
              key = "dia";
              fixed_width = true;
              is_enabled = false;
          }
          : popup_list {
              label = "Chaveta:";
              key = "chave";
              mnemonic ="h";
              edit_width = 7;
              value = "7";
              is_enabled = false;
          }
      }


  }
 }
 ok_cancel_help_errtile;
}

ok_error : dialog {
           label = "ALERTA";
    : text {
      value = "error de parametros";
    }
    ok_button;
}
//-------------------------------------------------------------------------
alerta : dialog {
  label = "Alert AutoCAD";
  : paragraph {
       : text_part {
           label = "";
       }
       : text_part {
           key = "text1";
           width = 33;
       }
       : text_part {
```

```
            key = "text2";
            width = 33;
        }
    }
    : row {
        fixed_width = true;
        alignment = centered;
        : button {
            label = "Si";
            key = "si";
            mnemonic = "S";
            is_default = true;
            fixed_width = true;
        }
        : spacer {width = 2;}
        : button {
            label = "No";
            key = "no";
            mnemonic = "N";
            is_default = true;
            fixed_width = true;
        }
    }
}
```

LISTADO 5.3 La rutina DENGRANE.LSP

```
;=============================================================================
;        NOMBRE DEL ARCHIVO:    DENGRANE.LSP - AutoCAD Version 12
;                Escrito por:          Renato Parodi.
;=============================================================================


(DEFUN *error* (cad)
   (PRINC "Error: ") (PRINC cad) (TERPRI)
   (UNLOAD_DIALOG dcl_id)
   (restore_engrane)
   (PRIN1)
)
;-----------------------------------------------------------------------------
(DEFUN ai_abort_engrane (app msg)
   (IF msg
     (ALERT (STRCAT "Error de aplicacion: "
                    app
                    "  \n\n  "
                    msg
                    "   \n"
            )
     )
   )
   (EXIT)
)
(COND ( (NOT (FINDFILE "engrane.lsp"))
        (ai_abort_engrane "PROGRAMA ENGRANJE"
                  (STRCAT "No se encuentra el archivo ENGRANE.LSP"
                      "\n Cheque el directorio" " " (GETVAR "dwgprefix"))))
      ( (NOT (FINDFILE "engrane.dcl"))
        (ai_abort_engrane "ENGRANJE"
                  (STRCAT "No se encuentra el archivo ENGRANE.DCL"
                          "\n Cheque el directorio" " " (GETVAR "dwgprefix"))))
      ( (NOT (FINDFILE "cuadro.dwg"))
        (ai_abort_engrane "ENGRANJE"
                  (STRCAT "No se encuentra el archivo CUADRO.DWG"
                          "\n Cheque el directorio" " " (GETVAR "dwprefix"))))

)
;-----------------------------------------------------------------------------
(DEFUN ok_real (var vartipo)
   (COND ((DISTOF var 2) var)
         (T (SET_TILE "error" (STRCAT  "Invalido valor " vartipo)) nil)
   )
)
;-----------------------------------------------------------------------------
(DEFUN get_alerta ()
   (SETQ dcl_id (LOAD_DIALOG "engrane"))
   (IF (NOT (NEW_DIALOG "alerta" dcl_id)) (EXIT))
   (SET_TILE "text1" (STRCAT "Atencion: el archivo "(STRCASE narch)))
   (SET_TILE "text2" "existe, desea remplazarlo?")
   (ACTION_TILE "si" "(SETQ op $key) (done_dialog 1)")
   (ACTION_TILE "no" "(done_dialog)")
   (SETQ what_next1 (START_DIALOG))
   (IF (= what_next1 0)
       (PROGN (SETQ ok_para nil) (MODE_TILE "file" 2))
   )
)
;-----------------------------------------------------------------------------

(DEFUN get_pantalla ()
   (SET_TILE "error" "")
   (IF (= "0" (SETQ pantalla (GET_TILE "pantalla")))
       (PROGN
          (MODE_TILE "x_pt" 0) (MODE_TILE "y_pt" 0)
          (MODE_TILE "pcen" 0) (MODE_TILE "angp" 0)
          (MODE_TILE "nd" 0) (MODE_TILE "mod" 0)
          (MODE_TILE "chavetero" 0) (MODE_TILE "cua" 0)
```

```
        )
        (PROGN
           (MODE_TILE "x_pt" 1) (MODE_TILE "y_pt" 1)
           (MODE_TILE "pcen" 1) (MODE_TILE "angp" 1)
           (MODE_TILE "nd" 1)   (MODE_TILE "mod" 1)
           (MODE_TILE "dia" 1) (MODE_TILE "chave" 1)
           (MODE_TILE "chavetero" 1) (MODE_TILE "cua" 1)

        )
    )
)
;-------------------------------------------------------------------------
(DEFUN get_chave_engrane ()
    (SET_TILE "error" "")
    (IF (= "1" (SETQ chavetero (GET_TILE "chavetero")))
        (PROGN
           (MODE_TILE "dia" 0) (MODE_TILE "chave" 0)
        )
        (PROGN
           (MODE_TILE "dia" 1) (MODE_TILE "chave" 1)
        )
    )
)
;-------------------------------------------------------------------------
(DEFUN get_list1 ()
    (SETQ list1 (LIST "  14«ø" "  20ø" "  25ø" "  30ø"))
    (START_LIST "angp")
    (MAPCAR 'ADD_LIST list1)
    (END_LIST)
)
;-------------------------------------------------------------------------
(DEFUN get_list2 ()
    (SETQ list2 (LIST " 6 - 8" " 8 - 10" "10 - 12" "12 - 17" "17 - 22"
                      "22 - 30" "30 - 38" "38 - 44" "44 - 50" "50 - 58"
                      "58 - 65" "65 - 75" "75 - 85"))
    (START_LIST "chave")
    (MAPCAR 'ADD_LIST list2)
    (END_LIST)
)
;-------------------------------------------------------------------------
(DEFUN get_x_engrane ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ x_pt (GET_TILE "x_pt")) "x cordenada")
        (SETQ pcen (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_y_engrane ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ y_pt (GET_TILE "y_pt")) "y cordenada")
        (SETQ pcen (LIST (DISTOF (GET_TILE "x_pt")) (DISTOF (GET_TILE "y_pt"))))
    )
    nil
)
;-------------------------------------------------------------------------
(DEFUN get_modulo ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:mod (GET_TILE "mod")) "del modulo")
        (SETQ mod (DISTOF g:mod))
    )
)
;-------------------------------------------------------------------------
(DEFUN get_diente ()
    (SET_TILE "error" "")
    (IF (ok_real
            (SETQ g:nd (GET_TILE "nd")) "del numero de dientes")
        (SETQ nd (ATOI g:nd))
```

```
    )
)
;-------------------------------------------------------------------------
(DEFUN get_arbol ()
   (SET_TILE "error" "")
   (IF (ok_real
          (SETQ g:diam (GET_TILE "dia")) "del diametro")
       (SETQ diam (DISTOF g:diam))
   )
   (COND ( (>= diam 85)
           (SET_TILE "chave" (SETQ chave "12")))
         ( (<= diam 8)
           (SET_TILE "chave" (SETQ chave "0")))
         ( (AND (> diam 8) (< diam 85))
           (SETQ vmin 6 vmax 8)
           (SETQ n 0)
           (WHILE (NOT (AND (>= diam vmin) (<= diam vmax)))
              (SETQ vmin  (ATOI (SUBSTR (NTH n list2) 1 3)))
              (SETQ vmax (ATOI (SUBSTR (NTH n list2) 5)))
              (SETQ n (+ n 1))
           )
           (SET_TILE "chave" (SETQ chave (ITOA (- n 1))))
         )
   )
)
;-------------------------------------------------------------------------
(DEFUN get_rea ()
   (SET_TILE "error" "")
   (SETQ rea (GET_TILE "rea"))
   (IF (= rea "1")
       (PROGN
          (MODE_TILE "file" 0)
          (MODE_TILE "file" 2)
       )
       (MODE_TILE "file" 1)
   )
)
;-------------------------------------------------------------------------
(DEFUN setloc_engrane ()
   (IF pcen
       (SETQ x_pt (RTOS (CAR pcen) 2 4)
             y_pt (RTOS (CADR pcen) 2 4)
       )
       (SETQ pcen (LIST
                        (DISTOF (SETQ x_pt "0.0000"))
                        (DISTOF (SETQ y_pt "0.0000"))
                  )
       )
   )
   (SET_TILE "x_pt" x_pt)
   (SET_TILE "y_pt" y_pt)
   (IF (NOT mod)
       (SETQ mod 1))
   (SET_TILE "mod" (SETQ g:mod (RTOS mod 2)))
   (IF (NOT nd)
       (SETQ nd 12))
   (SET_TILE "nd" (SETQ g:nd (ITOA nd)))
   (IF (NOT angp)
       (SETQ angp "2"))
   (SET_TILE "angp" (SETQ g:angp angp))
   (IF (NOT diam)
       (SETQ diam 6))
   (SET_TILE "dia" (SETQ g:diam (RTOS diam 2)))
   (IF (NOT chave)
       (SETQ chave "0"))
   (SET_TILE "chave" (SETQ g:chave chave))
)
;-------------------------------------------------------------------------
(DEFUN verify_engrane ()
   (SETQ di (* mod (- nd 2.334)))
   (COND ( (AND (SETQ ok_para T) (/= (TYPE pcen) 'LIST))
           (SETQ ok_para nil)
```

```
          (SET_TILE "error" "Invalido punto central"))
        ( (AND ok_para (<= mod 0))
          (MODE_TILE "mod" 2) (SETQ ok_para nil)
          (SET_TILE "error" "Invalido valor del modulo (Mayor a 0)"))
        ( (AND ok_para (< nd 12))
          (MODE_TILE "nd" 2) (SETQ ok_para nil)
          (SET_TILE "error" "Invalido numero de dientes (12 o mas)"))
        ( (AND ok_para (< diam 6))
          (MODE_TILE "dia" 2) (SETQ ok_para nil)
          (SET_TILE "error" "Invalido valor del diametro (6 o mas)"))
        ( (AND ok_para (> diam (* 0.8 di)) (= chavetero "1"))
          (MODE_TILE "dia" 2) (SETQ ok_para nil)
          (ALERT (STRCAT "                    ***ATENCION***          \n"
                         "            El valor del DIAMETRO DEL ARBOL "
                         "\nExcede al valor del DIAMETRO INTERIOR del engranaje."))
          (SET_TILE "error" (STRCAT "Incorrecto valor del diametro del arbol "
                                    "(MAXIMO " (ITOA (FIX (* 0.8 di)))"mm)")))
        ( (AND  ok_para (NOT (ok_real g:mod "del modulo")))
          (MODE_TILE "mod" 2) (SETQ ok_para nil))
        ( (AND ok_para (NOT (ok_real g:nd "del numero de diente")))
          (MODE_TILE "nd" 2) (SETQ ok_para nil))
        ( (AND ok_para (NOT (ok_real g:diam "del diametro")))
          (MODE_TILE "dia" 2) (SETQ ok_para nil))
        ( (AND ok_para (NOT (ok_real x_pt "X cordenada")))
          (MODE_TILE "x_pt" 2) (SETQ ok_para nil))
        ( (AND ok_para (NOT (ok_real y_pt "Y cordenada")))
          (MODE_TILE "y_pt" 2) (SETQ ok_para nil))
        ( (= rea "1")
          (COND ( (AND ok_para (= narch ""))
                  (SET_TILE "error" "Incorrecto nombre de archivo.")
                  (MODE_TILE "file" 2) (SETQ ok_para nil))
                ( (AND ok_para (WCMATCH narch "*.rpz,*.RPZ"))
                  (SETQ narch (SUBSTR narch 1 12))
                  (IF (FINDFILE narch) (get_alerta)))
                ( (AND ok_para (WCMATCH narch "*.*"))
                  (SET_TILE "error" "Incorrecto nombre de archivo.")
                  (MODE_TILE "file" 2) (SETQ ok_para nil))
                ( T
                  (SETQ narch (STRCAT (SUBSTR narch 1 8) ".rpz"))
                  (IF (FINDFILE narch) (get_alerta)))
          )
        )
    )
)
;-----------------------------------------------------------------------------
(DEFUN restore_engrane ()
    (COMMAND "layer" "s" oldlayer "")
    (COMMAND "color" oldcolor)
    (SETVAR "blipmode" oldblp)
    (SETVAR "cmdecho" oldech)
)
;-----------------------------------------------------------------------------
(DEFUN c:engrane (/  ok_para cua rea rep narch chavetero pcen mod nd angp chave diam
chavetero pantalla)
    (SETQ oldblp (GETVAR "blipmode")
          oldech (GETVAR "cmdecho")
          oldcolor (GETVAR "cecolor")
          oldlayer (GETVAR "clayer")
    )
    (SETVAR "blipmode" 0) (SETVAR "cmdecho" 0)
    (SETQ what_next 5)
    (SETQ dcl_id (LOAD_DIALOG "engrane"))
    (WHILE (< 2 what_next)
       (IF (NOT (NEW_DIALOG "engrane" dcl_id)) (EXIT))
       (get_list1) (get_list2)
       (setloc_engrane)
       (ACTION_TILE "pantalla" "(get_pantalla)")
       (ACTION_TILE "pcen" "(done_dialog 3)")
       (ACTION_TILE "x_pt" "(get_x_engrane)")
       (ACTION_TILE "y_pt" "(get_y_engrane)")
       (ACTION_TILE "mod" "(get_modulo)")
       (ACTION_TILE "nd" "(get_diente)")
```

APENDICE B

INSTALACION DEL PROGRAMA TESIS

B.1 NTRODUCCION

En este apéndice se presenta, de una manera sencilla y fácil de comprender la instalación del PROGRAMA TESIS que le será de gran utilidad para los diseñadores mecánicos.

El disquete anexo contiene dicho programa TESIS, el mismo que debe ser ejecutado del editor de dibujo de AutoCAD. Este programa le ayudará a diseñar gráficamente y en forma rápida tres elementos mecánicos importantes como son: RESORTES DE TENSION, ENGRANAJES DE DIENTES RECTOS y LEVA DE PLACA PLANA. A demás el programa TESIS contiene una rutina que le permitirá la graficación de forma automática del cuadro de rotulación normalizado que se emplea en Ingeniería Mecánica.

B.2 INSTALACION

El programa incluye un archivo instalador llamado INSTALAR, por lo que solo habrá que seguir los pasos indicados en él. El procedimiento habitual es situarse en la unidad de disco flexible A: o B: y de allí, llamar al instalador.

```
A:\>instalar
```

Aparece una pantalla de dialogo con el menú principal como se muestra a continuación:

***** MENU PRINCIPAL *****

1. Instalar en un Nuevo Subdirectorio de AutoCAD

2. Instalar en el Subdirectorio Support

3. Exit

Opción ?:

- **Instalar en un Nuevo Subdirectorio de AutoCAD:** Esta opción presenta una nueva pantalla de diálogo en la cual se visualiza tres casillas. En la primera casilla se coloca el nombre del directorio en el cual se encuentra instalado AutoCAD. Ejemp: ACAD12.

  En la segunda casilla se coloca el nombre del nuevo subdirectorio que se va a crear. Ejemp: LIBRERIA. En este subdirectorio se guardará todos los archivos del programa TESIS.

  La tercera casilla solo es de lectura. La cual muestra e indica el camino dónde se lee los archivos originales y el camino donde se copian cada archivo del programa Tesis.

  Al terminar la instalación, el instalador lista y modifica el contenido de un archivo llamado ACADR12.BAT que contiene los valores adecuados de variables de entorno de AutoCAD, y que se puede ejecutar para establecerlo.

- **Instalar en el Subdirectorio Support:** Esta opción presenta una nueva pantalla de diálogo en la cual se visualiza tres casillas. En la primera casilla se coloca el nombre del directorio en el cual se encuentra instalado AutoCAD. Ejemp: ACAD12.

  En la segunda casilla es de solo lectura, o sea el usuario no tiene acceso a ella. En esta casilla se visualiza el subdirectorio SUPPORT el cual fue creado cuando se instaló AutoCAD. En el subdirectorio SUPPORT se guardará todos los archivos del programa TESIS.

  La tercera casilla sclo es de lectura. La cual muestra e indica el camino donde se lee los archivos originales y el camino donde sé copian cada archivo del programa Tesis.

- **EXIT:** Esta opción sirve para salir del programa instalador y regresar al prompt de DOS.

BIBLIOGRAFIA

1.- QUEINNEC C.; "PROGRAMACION EN LISP"; EDITORIAL PARANINFO, PRIMERA EDICION EN ESPAÑOL, 1987

2.- COGOLLOR JOSE L.; "ENCICLOPEDIA DE AutoCAD 11"; EDITORIAL ADDISON-WESLEY, 1992

3.- COGOLLOR JOSE L.; "PROGRAMACION EN AutoLISP"; EDITORIAL MACROBIT™, 1991

4.- JAMSA KRIS; "DOS MANUAL DE REFERENCIA"; EDITORIAL Mc.GRAW-HILL, PRIMERA EDICION, 1988

5.- OMURA GEORGE; "AutoCAD REFERENCIA INSTANTANEA"; EDITORIAL MACROBIT™, 1990

6.- VILLACIS EDMUNDO; "DISEÑO DE RESORTES HELICOIDALES" ; FIM-ESPOL, 1993

7.- SCHILDT HERBERT; "PROGRAMACION EN TURBO C++"; EDITORIAL Mc.GRAW-HILL, SEGUNDA EDICION, 1991

8.- TORO LOAISA; "DISEÑO DE LEVAS DE PLACA PLANA"; FIM-SPOL, 1993

9.- MARKS; "MANUAL DEL INGENIERO MECANICO"; EDITORIAL Mc. GRAW-HILL, SEGUNDA EDICION, 1983

10.-MABIE HAMILTON H.;"MECANISMOS Y DINAMICA DE MAQUINARIA" EDITORIAL LIMUSA, 1988

11.-CASILLAS A.L.; "MAQUINAS CALCULO DE TALLER"