MAKE*CIRC*

Una biblioteca METAPOST para dibujar diagramas circuitales eléctricos

Gustavo S. Bustamante Argañaraz*

Enero de 2004

Índice

1. Introducción

MAKE*CIRC* es una biblioteca para METAPOST que contiene diversos símbolos (en su mayoría eléctricos) para el uso en diagramas circuitales. MAKE*CIRC* intenta brindar una herramienta de alta calidad, con una sintaxis simple.

*Gracias a José Luis Díaz por la gran ayuda que me brindó respondiendo a mis dudas y cuestiones sobre METAPOST, y proporcionando la biblioteca `latex.mp`, la cual es utilizada en esta biblioteca. Gracias también a Sebastián Pablo Sánchez por su ayuda en la versión en inglés de este manual.

A pesar de emplear toda la potencia de METAPOST, MAKE*CIRC* no requiere el aprendizaje de este lenguaje para poder ser utilizado. Sin embargo, el conocimiento del mismo hace aún más fácil la implementación de esta biblioteca.

MAKE*CIRC* se integra completamente con documentos de LaTeX y con otros dibujos/gráficos de METAPOST. Su salida es un archivo POSTSCRIPT.

2. Configuración

MAKE*CIRC* provee tres parámetro que pueden ser configurados externamente:

- `linewd` especifica el grosor de los trazos;
- `lbsep` especifica la separación entre las etiquetas y el símbolo;
- `dimen` especifica el tamaño de los símbolos; y
- `labeling` que puede tomar dos valores, `rotatelabel` para que las etiquetas de los símbolos roten junto con ellos, o `norotatelabel` (por defecto) para que las etiquetas permanezcan en posición normal.

Los tres primeros parámetros se deben escalar para modificar su valor por defecto, es decir, si se quiere que el grosor de los trazos sea el doble que el grosor actual, se debe escribir:

```
linewd:=2linewd;
```

De la misma manera se hará con los los otros dos:

```
lbsep:=2lbsep; dimen:=2dimen;
```

siendo `dimen` un parámetro (dimensión caracterísitca) que varía según el símbolo del que se trate, tal como se muestra en el cuadro 1.

El parámetro `labeling`, como se describió, puede tomar dos valores, los cuales deben ser cambiados especificando `labeling:=valor`, por ejemplo:

```
labeling:=rotatelabel;
```

Los cuatro parámetros de configuración pueden aplicarse bien a elementos determinados, o bien a todos los elementos del circuito. En este caso se deben colocar antes de comenzar a insertar los símbolos.

En el caso de `linewd`, una vez que se escaló su valor un determinado `factor`, para volverlo a su valor normal, tenemos que especificar

```
linewd:=linewd/factor;
```

Cuadro 1: Dimensiones características de los símbolos.

Elemento	**Dimensión característica**
Resistencia	`rstlth`
Capacitor	`platsep`
Inductor	`coil`
Transformador	
Fuente de alimentación	`ssize`
Batería	
Motor	
Generador	
Instrumento de medición	
Lámpara	
Diodo	`diodeht`
Transistor	`bjtlth`
Impedancia	`implth`
Reóstato	`rheolth`
Puesta a tierra	`gndlth`
Unión	`junctiondiam`
Llave	`ssep`

debido a que el valor que tiene en ese momento `linewd` es `factor` veces `linewd`, por lo tanto si lo dividimos por el mismo factor el resultado será `linewd`.

Por ejemplo:

```
elemento1; elemento2;
labeling:=rotatelabel;
elemento3; % en este elemento rotan las etiquetas %
linewd:=2linewd;
elemento4; % en este también y además se incrementa
% el espesor de los trazos al doble %
labeling:=norotatelabel; linewd:=linewd/2;
% vuelven a su valor por defecto %
elemento5; elemento6;
```

Si se quiere aumentar proporcionalmente el tamaño de todos los elementos del circuito junto con el trazo y etiquetas de los mismos, se debe emplear como última instrucción

```
scalecirc factor;
```

con lo que se escala el tamaño del circuito en una cantidad `factor`.

3. Utilización

La metodología que sugiere MAKECIRC para la creación de diagramas circuitales es la siguiente:

a) colocar los símbolos en la posición deseada; y
b) conectarlos utilizando sus pines.

Este método se cree más conveniente, pero el usuario puede implementar cualquier otro que le resulte más simple y eficaz.

3.1. Inicio

Para comenzar se debe especificar la entrada de la biblioteca de la siguiente manera:

```
input makecirc;
```

Luego se debe llamar a la función que inicializa LaTeX, si es que se quiere utilizar este programa para tipografiar las etiquetas (si se usa TeX no es necesario). Esta macro se utiliza para cargar los paquetes o definir los comandos que se necesitarán en la composición de las etiquetas:

```
initlatex("\usepackage{paquete}" &
          "\usepackage{otro_paquete}" &
          "\newcommand{comando}{definicion}");
```

El siguiente paso es comenzar los diagramas circuitos con la declaración `beginfig(n)`, siendo `n` el número de la figura. Cuando el circuito está finalizado se coloca `endfig`. Si se quiere realizar otro circuito se colocará nuevamente:

```
beginfig(k); % k es otro número distinto de n %
% ... aca va el circuito ... %
endfig;
```

Cuando todos los circuitos estén finalizados se colocará `end`.

Es importante mencionar que todas las declaraciones que se utilizan en METAPOST —y por tanto en MAKECIRC— se deben finalizar con punto y coma (`;`).

3.2. Símbolos

En este apartado se describirá la sintaxis para la composición de símbolos de MAKECIRC.

La mayoría de los símbolos más comunes se componen de la siguiente manera:

```
elemento.α(z,tipo,angulo,nombre,valor);
```

donde: α es un caracter alfabético (a, b, c, A, B, C, ab, cd, etc.) que luego será utilizado como parte de los pines del elemento;
`z=(x,y)` son las coordenadas del punto de inserción del elemento;
`tipo` es una subcategoría que dependerá del elemento insertado;
`angulo` es el ángulo en grados que estará girado (en sentido antihorario) el elemento a partir de su posición inicial;
`nombre` es el nombre o letra característica del elemento (`L`, `R`, etc.);
y
`valor` es el valor numérico del elemento.

Sin embargo, hay otros elementos que contienen mayor o menor cantidad de parámetros que se irán describiendo a lo largo de este apartado.

Cabe agregar que los parámetros `nombre` y `valor` se deben colocar entre comillas (`"`). Además en estos parámetros se pueden utilizar comandos LaTeX.

El parámetro `nombre` está definido en *modo matemático*, esto es en lenguaje LaTeX entre $ y $. Por lo tanto son válidas todas las funciones y comandos matemáticos de LaTeX. Es decir, se podrá colocar `"L_1^2"` y su resultado será L_1^2, `"v=\hat{V}\sen\omega t"` y el resultado será $v = \hat{V}\,\mathrm{sen}\,\omega t$, etc. Esto se debe a que en la mayoría de los casos, el nombre de un símbolo será una constante o variable matemática (L, R, Z, v, i, etc.).

En contraste, el parámetro `valor` está definido en modo texto, por requerirse éste en la mayoría de los casos. (`"10 V"` $\Longrightarrow$ 10 V, `"50 pF"` $\Longrightarrow$ 50 pF, etc.). Una excepción en este caso —que requiere escribirse en modo matemático—, es la unidad de resistencia eléctrica (el ohm Ω). Para facilitar este trabajo a aquellos que no tienen conocimientos de LaTeX, MAKE*CIRC* incluye los comandos `\ohm` y `\kohm`. Por lo tanto, si se escribe `"10\ohm"`, el resultado será 10 Ω, `"42\kohm"` y el resultado será 42 kΩ. Nótese que no hay espacios entre el texto y los comandos.

También, MAKE*CIRC* incluye otro comando `\modarg` para escribir números complejos en la forma módulo-argumento.[1] Por ejemplo si se escribe `"\modarg{220}{30}"` el resultado será $220\angle 30°$.

Es importante mencionar que si el texto de las etiquetas contiene errores, LaTeX no podrá completar la compilación. Esto hará imposible correr METAPOST en el archivo fuente —aún si los errores son corregidos— hasta que se borre el archivo auxiliar (el cual tendrá una extensión `.mpt`).

A continuación se describirán e ilustrarán todos los símbolos disponibles en MAKE*CIRC*. También se indicará cuál es el pin que se posiciona en el punto `z` de inserción del elemento.

[1] Gracias a Javier Bezos por esto.

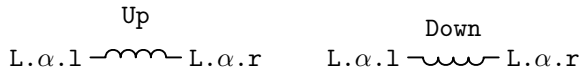
Elemento: Inductor

Sintaxis: `inductor.`α`(z,tipo,angulo,nombre,valor)`

Tipos Disponibles: `Up|Down`

Pines: `L.`α`.l|L.`α`.r`

Pin de posicionamiento en `z`: `L.`α`.l`



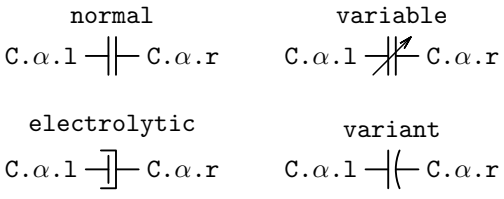
Elemento: Capacitor

Sintaxis: `capacitor.`α`(z,tipo,angulo,nombre,valor)`

Tipos Disponibles: `normal|variable|electrolytic|variant`

Pines: `C.`α`.l|C.`α`.r`

Pin de posicionamiento en `z`: `C.`α`.l`



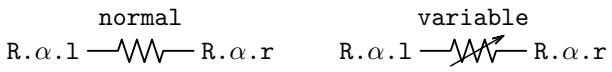
Elemento: Resistor

Sintaxis: `resistor.`α`(z,tipo,angulo,nombre,valor)`

Tipos Disponibles: `normal|variable`

Pines: `R.`α`.l|R.`α`.r`

Pin de posicionamiento en `z`: `R.`α`.l`



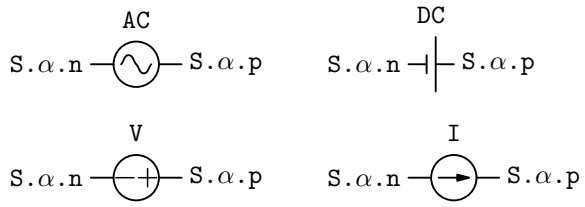
Elemento: Fuente de alimentación

Sintaxis: `source.`α`(z,tipo,angulo,nombre,valor)`

Tipos Disponibles: `AC|DC|V|I`

Pines: `S.`α`.n|S.`α`.p`

Pin de posicionamiento en z: S.α.n



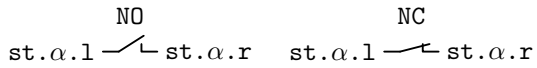
Elemento: Llave

Sintaxis: switch.α(z,tipo,angulo,nombre,valor)

Tipos Disponibles: NO|NC

Pines: st.α.l|st.α.r

Pin de posicionamiento en z: st.α.l

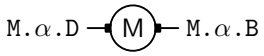


Elemento: Motor

Sintaxis: motor.α(z,angulo,nombre,valor)

Pines: M.α.D|M.α.B

Pin de posicionamiento en z: M.α.D

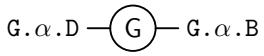


Elemento: Generador

Sintaxis: generator.α(z,angulo,nombre,valor)

Pines: G.α.D|G.α.B

Pin de posicionamiento en z: G.α.D

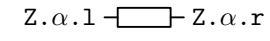


Elemento: Impedancia

Sintaxis: `impedance.`α`(z,angulo,nombre,valor)`

Pines: `Z.`α`.l|Z.`α`.r`

Pin de posicionamiento en `z`: `Z.`α`.l`

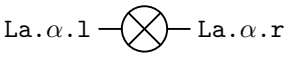


Elemento: Lámpara

Sintaxis: `lamp.`α`(z,angulo,nombre,valor)`

Pines: `La.`α`.l|La.`α`.r`

Pin de posicionamiento en `z`: `La.`α`.l`



Elemento: Batería

Sintaxis: `battery.`α`(z,angulo,nombre,valor)`

Pines: `B.`α`.n|B.`α`.p`

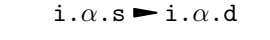
Pin de posicionamiento en `z`: `B.`α`.n`

Elemento: Corriente

Sintaxis: `current.`α`(z,angulo,nombre,valor)`

Pines: `i.`α`.s|i.`α`.d`

Pin de posicionamiento en `z`: `i.`α`.s`



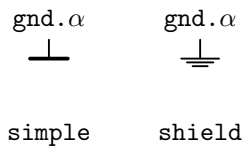
Elemento: Puesta a tierra

Sintaxis: `ground.`α`(z,tipo,angulo)`

Tipos Disponibles: `simple|shield`

Pines: `gnd.`α

Pin de posicionamiento en `z`: `gnd.`α



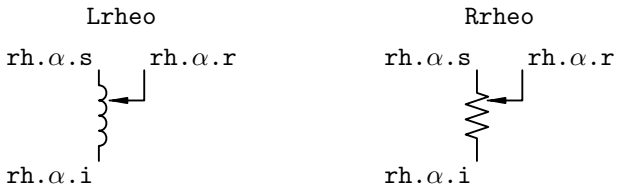
Elemento: Reóstato

Sintaxis: `rheostat.`α`(z,tipo,angulo)`

Tipos Disponibles: `Lrheo|Rrheo`

Pines: `rh.`α`.i|rh.`α`.s|rh.`α`.r`

Pin de posicionamiento en `z`: `rh.`α`.i`



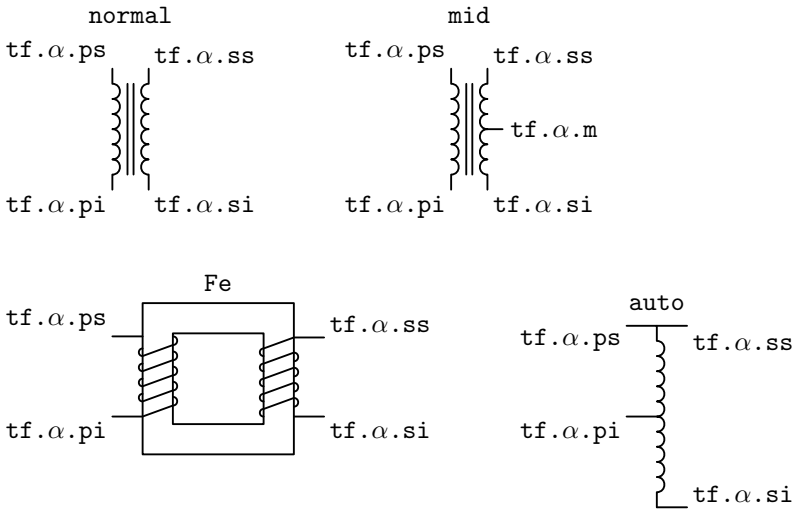
Elemento: Transformador

Sintaxis: `transformer.`α`(z,tipo,angulo)`

Tipos Disponibles: `normal|mid|Fe|auto`

Pines: `tf.`α`.pi|tf.`α`.ps|tf.`α`.si|tf.`α`.ss|tf.`α`.m`

Pin de posicionamiento en `z`: `tf.`α`.pi`



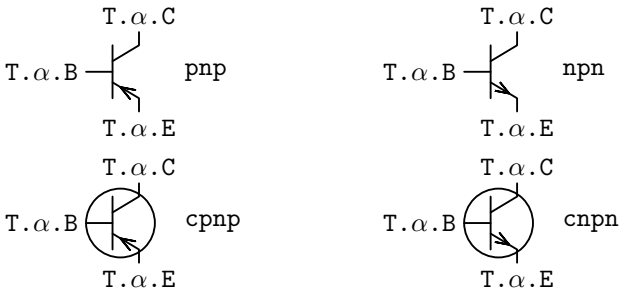
Elemento: Transistor

Sintaxis: `transistor.α(z,tipo,angulo)`

Tipos Disponibles: `pnp|npn|cpnp|cnpn`

Pines: `T.α.B|T.α.C|T.α.E`

Pin de posicionamiento en `z`: `T.α.B`



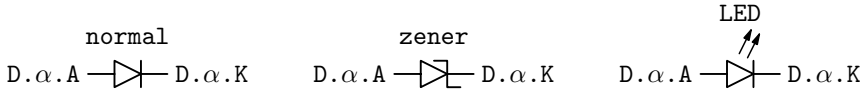
Elemento: Diodo

Sintaxis: `diode.α(z,tipo,angulo,pin,nombre,valor)`

Tipos Disponibles: `normal|zener|LED`

Pines: `D.α.A|D.α.K`

Pin de posicionamiento en `z`: `D.α.A (pin=pinA)|D.α.K (pin=pinK)`



El diodo es un elemento especial, por lo que requiere una sintaxis especial. Esto se debe a que un circuito que contiene un diodo se comporta de diferentes maneras cuando este último está en polarización directa o en polarización inversa, es decir, interesa el sentido de colocación del diodo. Es por eso que se agrega un parámetro `pin`, el cual toma dos valores: `pinA` o `pinK`. Esto indica qué pin es el que se quiere ubicar en la posición `z`, si el ánodo (`pinA`) o el cátodo (`pinK`).

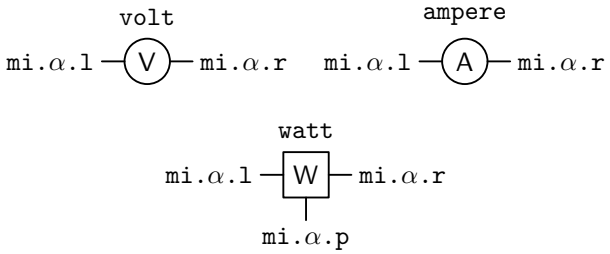
Elemento: Instrumento de medición

Sintaxis: `meains.`α`(z,tipo,angulo,texto)`

Tipos Disponibles: `volt|ampere|watt`

Pines: `mi.`α`.l|mi.`α`.r|mi.`α`.p`

Pin de posicionamiento en `z`: `mi.`α`.l`

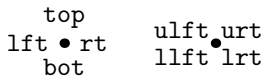


Elemento: Conexión

Sintaxis: `junction.`α`(z,texto)(pos)`

Posición (`pos`): `top|bot|lft|rt|ulft|urt|llft|lrt`

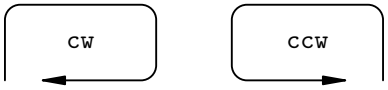
Pines: `J.`α



Elemento: Corriente de malla

Sintaxis: `imesh(centro,ancho,alto,sentido,angulo,nombre)`

Sentido: `cw|ccw`



3.2.1. Posicionamiento

La colocación de los símbolos puede hacerse utilizando coordenadas relativas (lo más usual) o absolutas. Las relativas generalmente se referirán

a pines de elementos colocados anteriormente.

El primer símbolo siempre se coloca con coordenadas absolutas y generalmente en el origen de coordenadas (`origin=(0,0)`). Por ejemplo:

```
inductor.a(origin,Up,0,"L_1","10 H");
```

A partir de este elemento se colocan los restantes utilizando las coordenadas relativas en la forma `pin.ref+(x,y)`, lo cual significa que el próximo elemento se ubicará a una distancia `x` hacia la derecha e `y` hacia arriba, a partir del pin `pin.ref`. Por ejemplo si el punto de inserción de mi próximo símbolo es `L.a.r+(1cm,5mm)` significa que estará a `1cm` a la derecha y `5mm` hacia arriba del pin `L.a.r`, mientras que la declaración `L.a.l+(-1cm,5mm)` establece el punto a `1cm` hacia la izquierda y `5mm` hacia arriba del pin `L.a.l`.

Sin embargo, esto no cubre todas las necesidades del usuario, porque éste puede querer centrar un elemento entre dos puntos (o pines) y no conocer las coordenadas de dichos puntos. Para estos casos MAKE*CIRC* provee la siguiente declaración:

```
centreof.α(pin1,pin2,sym);
```

siendo `sym` el símbolo que queremos centrar. Para indicar esto se utilizará una abreviación conveniente de los elementos (véase cuadro 2).

Cuadro 2: Abreviaciones para los elementos.

Elemento	Abreviación
Inductor	`ind`
Capacitor	`cap`
Motor	`mot`
Generador	`gen`
Transformador	`tra`
Fuente de alterna	`sac`
Fuente de continua	`sdc`
Fuente de corriente	`si`
Fuente de tensión	`sv`
Batería	`bat`
Resistencia	`res`
Diodos	`dio`
Transistores	`bjt`
Instrumentos de medición	`ins`
Impedancia	`imp`
Lámpara	`lam`
Llaves	`swt`
Corriente	`cur`

Esta declaración devuelve el punto de inserción del elemento (`c.`α) y el ángulo `phi.`α cuya dirección es la recta que une los pines `pin1` y `pin2`.

Para aclarar un poco estos aspectos de posicionamiento se pondrá un ejemplo. Incluyo una resistencia R de $10\,\Omega$ a 90 grados en el origen de coordenadas (véase apartado 3.2):

```
resistor.a(origin,normal,90,"R","10\ohm");
```

A continuación coloco una bobina L situada a 2 centímetros a la derecha de la resistencia:

```
inductor.a(R.a.r+(2cm,0),Up,-90,"L",);
```

Ahora quiero centrar un capacitor entre estos dos elementos. Entonces coloco:

```
centreof.A(R.a.r,L.a.l,cap);
```

lo que devuelve el punto de inserción `c.A` donde se debe colocar el capacitor, y el ángulo `phi.A` que estará girado. Ahora coloco el capacitor C de la siguiente manera:

```
capacitor.a(c.A,normal,phi.A,"C",);
```

Es decir que el código completo sería:

```
beginfig(1);
        resistor.a(origin,normal,90,"R","10\ohm");
        inductor.a(R.a.r+(2cm,0),Up,-90,"L","");
        centreof.A(R.a.r,L.a.l,cap);
        capacitor.a(c.A,normal,phi.A,"C","");
endfig;
```

cuyo resultado es el que muestra la figura 1.

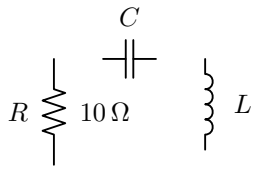


Figura 1: Ejemplo de posicionamiento.

Otra posibilidad que puede presentarse en el momento de posicionar un elemento, es que se quiere colocar un símbolo debajo o al lado de otro, centrado con este último. Para ello MAKECIRC provee la siguiente declaración:

```
centerto.α(pin1,pin2,dist,sym)
```

siendo `dist` la distancia de separación desde el elemento cuyos pines son `pin1` y `pin2` hasta el elemento que queremos colocar. Esta distancia puede ser un valor positivo — si se quiere desplazar el elemento hacia la derecha o hacia arriba— o negativo —si se lo quiere desplazar hacia la izquierda o abajo—. Por ejemplo, si se toma el caso anterior, puede verse que la bobina no está centrada con la resistencia. Para hacerlo se procede de la siguiente manera:

```
beginfig(2);
        resistor.a(origin,normal,90,"R","10\ohm");
        centerto.A(R.a.l,R.a.r,2cm,ind);
        inductor.a(A,Up,90,"L","");
endfig;
```

El resultado de ingresar este código se muestra en la figura 2 .

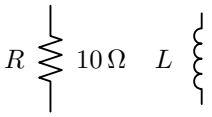


Figura 2: Ejemplo de centrado de elementos.

Si quisieramos ahora, como en el caso anterior, centrar el capacitor entre estos dos elementos tal como están, el resultado sería el que muestra la figura 3 , es decir, el capacitor está ligeramente girado en la parte derecha. Esto se debe a que los pines utilizados para el centrado están a diferentes alturas, por lo tanto el ángulo `phi` de la declaración `centreof` tendrá la dirección de la recta que une estos dos puntos.

Código fuente de la figura 3

```
1  beginfig(3);
2          resistor.a(origin,normal,90,"R","10\ohm");
3          centerto.A(R.a.l,R.a.r,2cm,ind);
4          inductor.a(A,Up,90,"L","");
5          centreof.B(R.a.r,L.a.r,cap);
6          capacitor.a(c.B,normal,phi.B,"C","");
7  endfig;
```

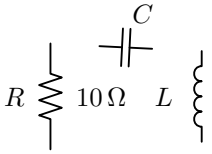


Figura 3: Capacitor girado por haberse centrado entre dos pines cuyas alturas son diferentes.

Para evitar esto, podría especificarse en lugar del ángulo `phi` que devuelve la declaración `centreof`, el ángulo cero (0), con lo que el capacitor

quedaría correctamente posicionado (horizontal), pero esto traerá problemas más tarde cuando se realicen las uniones de los elementos. En lugar de eso se deben utilizar las declaraciones `xpart` e `ypart`, que son propias de METAPOST y no de MAKE*CIRC*.

Estas declaraciones son útiles para especificar la coordenada absoluta horizontal (`xpart`) o vertical (`ypart`) de un punto. En este caso, es necesario conocer las coordenadas del punto z_1 (véase figura 4), el cual está a la misma altura que el pin de la resistencia, y por tanto permitirá realizar un centrado correcto del capacitor. Dichas coordenadas no se conocen explícitamente, pero están implícitas, porque este punto tiene la misma coordenada en x (horizontal) que el pin `L.a.r` y la misma coordenada en y (vertical) que el punto `R.a.r`, es decir que

$$z_1 = (\texttt{xpart L.a.r}, \texttt{ypart R.a.r})$$

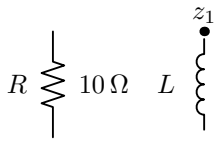


Figura 4: Punto z_1 a la misma altura que el pin de la resistencia.

Este es el punto que se debe utilizar como `pin2` para el centrado, es decir que a la línea 5 del código fuente de la figura 3 se la debe reemplazar por la siguiente:

```
centreof.B(R.a.r,(xpart L.a.r,ypart R.a.r),cap);
```

También se puede definir un punto `z1` que contenga esas coordenadas y luego especificarlo en el centrado,[2] es decir:

```
z1=(xpart L.a.r,ypart R.a.r);
centreof.B(R.a.r,z1,cap);
```

Si se adopta esta última forma, el código corregido cuyo resultado se muestra en la figura 5 es el siguiente:

Código fuente de la figura 5

```
beginfig(5);
        resistor.a(origin,normal,90,"R","10\ohm");
        centerto.A(R.a.l,R.a.r,2cm,ind);
        inductor.a(A,Up,90,"L","");
        z1=(xpart L.a.r,ypart R.a.r);
        centreof.B(R.a.r,z1,cap);
        capacitor.a(c.B,normal,phi.B,"C","");
endfig;
```

[2]Esto también es propio de METAPOST.

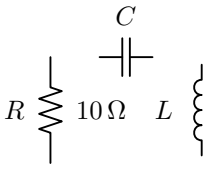


Figura 5: Utilización de las declaraciones `xpart` e `ypart` para realizar un correcto centrado del capacitor.

3.2.2. Conexionado

Para las uniones de los elementos entre sí, se utilizan dos declaraciones:

```
wire(pin1,pin2,tipo); y wireU(pin1,pin2,dist,tipo);
```

Los tipos disponibles para estas declaraciones se muestran en el cuadro 3.

Cuadro 3: Tipos disponibles para las declaraciones de unión de elementos.

Declaración	Tipos disponibles
`wire`	`nsq|udsq|rlsq`
`wireU`	`udsq|rlsq`

Si los pines están a la misma altura el tipo especificado debe ser `nsq`, con lo que la unión se realizará en línea recta, desde el `pin1` hasta el `pin2`. Esto también es válido para cualquier unión que se quiera realizar en línea recta. Si los pines, en cambio, están a diferente altura se debe especificar `udsq`, con lo que la unión se realizará en escuadra, partiendo desde el `pin1` hacia arriba o abajo y luego hasta el `pin2`, o bien `rlsq` y la unión se realizará en escuadra pero partiendo desde el `pin1` hacia la derecha o izquierda y luego hasta el `pin2`.

Estos dos últimos son válidos para la declaración `wireU`, pero en ésta se agrega un parámetro más, porque esta declaración es para cuando la unión se debe realizar con tres trazos (en forma de `U`), es decir para unir dos ramas en paralelo. En este caso, la dirección arriba, abajo, derecha o izquierda está dada por el signo del parámetro `dist` (véase cuadro 4).

Cuadro 4: Dirección según el signo de `dist`.

Tipo	Signo de `dist`	
	+	−
`udsq`	arriba	abajo
`rlsq`	derecha	izquierda

Para ilustrar esto, se tomarán los ejemplos anteriores y se le practicarán las uniones. Para el ejemplo de la figura 1 se utilizarán tres declaraciones de unión: una para unir la resistencia con el capacitor, otra para el capacitor

con la bobina, y una más para la bobina con la resistencia. Para la primera y segunda unión, como están los pines a la misma altura, será:

```
wire(R.a.r,C.a.l,nsq);
wire(C.a.r,L.a.l,nsq);
```

y para la tercera, como los pines no están a la misma altura puden darse dos casos:

si se parte desde la bobina deberá utilizarse el tipo `udsq` debido a que el pin de la bobina está más arriba que el pin de la resistencia, por lo tanto primero deberá realizarse un trazo hacia abajo y luego hacia el pin de la resistencia; en cambio

si se parte desde la resistencia deberá ser `rlsq` debido a que el pin de la resistencia está más abajo que el de la bobina, por lo tanto deberá hacerse un trazo primero hacia la derecha y luego hacia el pin de la bobina.

```
wire(L.a.r,R.a.l,udsq);
% o bien %
wire(R.a.l,L.a.r,rlsq);
```

Ambas intrucciones dan el mismo resultado.

Agregando las líneas de unión al código fuente de la figura 1 se obiene el siguiente código completo, cuyo resultado se muestra en la figura 6 .

```
beginfig(6);
        resistor.a(origin,normal,90,"R","10\ohm");
        inductor.a(R.a.r+(2cm,0),Up,-90,"L","");
        centreof.A(R.a.r,L.a.l,cap);
        capacitor.a(c.A,normal,phi.A,"C","");
        wire(R.a.r,C.a.l,nsq);
        wire(C.a.r,L.a.l,nsq);
        wire(L.a.r,R.a.l,udsq);
        % o bien
        % wire(R.a.l,L.a.r,rlsq);
endfig;
```

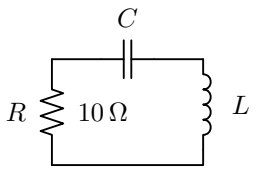


Figura 6: Ejemplo de unión de elementos.

Si tomamos ahora el ejemplo de la figura 5 las uniones serán:

```
wire(R.a.r,C.a.l,nsq);
% resistencia y capacitor, pines a la misma altura %
wire(C.a.r,L.a.r,rlsq);
% capacitor y bobina, pin C.a.r más alto que L.a.r %
wire(L.a.l,R.a.l,udsq);
% bobina y resistencia, pin L.a.l más alto que R.a.l %
```

Colocando estas líneas en el código fuente de la figura 5 resulta:

Código fuente de la figura 7

```
1  beginfig(7);
2          resistor.a(origin,normal,90,"R","10\ohm");
3          centerto.A(R.a.l,R.a.r,2cm,ind);
4          inductor.a(A,Up,90,"L","");
5          z1=(xpart L.a.r,ypart R.a.r);
6          centreof.B(R.a.r,z1,cap);
7          capacitor.a(c.B,normal,phi.B,"C","");
8          wire(R.a.r,C.a.l,nsq);
9          % resistencia y capacitor, pines a la misma altura %
10         wire(C.a.r,L.a.r,rlsq);
11         % capacitor y bobina, pin C.a.r más alto que L.a.r %
12         wire(L.a.l,R.a.l,udsq);
13         % bobina y resistencia, pin L.a.l más alto que R.a.l %
14 endfig;
```

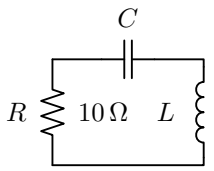


Figura 7: Otro ejemplo de unión.

Para mostrar la utilidad de la declaración `wireU` se tomará el código fuente de la figura 7 y se cambiará la línea 7 por la siguiente:

```
capacitor.a(c.B+(0,5mm),normal,phi.B,"C","");
```

es decir, en lugar de posicionar el capacitor en el punto `c.B` se lo coloca 5 milímetros más arriba (véase apartado 3.2.1). Haciendo esto se debe alterar también la línea 8 porque los pines de la resistencia y el capacitor ya no están a la misma altura:

```
wire(R.a.r,C.a.l,udsq);
```

y además, si se quiere que el circuito quede simétrico, en la línea 12 ya no se puede utilizar la declaración `wire` porque se necesita que el trazo baje 5 milímetros (los mismos que subió el capacitor) y recién vaya hasta el pin de la bobina. Para ello haremos:

```
wireU(L.a.l,R.a.l,-5mm,udsq);
```

Agregando estas líneas en el código fuente resulta:

Código fuente de la figura 8

```
beginfig(8);
        resistor.a(origin,normal,90,"R","10\ohm");
        centerto.A(R.a.l,R.a.r,2cm,ind);
        inductor.a(A,Up,90,"L","");
        z1=(xpart L.a.r,ypart R.a.r);
        centreof.B(R.a.r,z1,cap);
        capacitor.a(c.B+(0,5mm),normal,phi.B,"C","");
        wire(R.a.r,C.a.l,udsq);
        % resistencia y capacitor, pines a la misma altura %
        wire(C.a.r,L.a.r,rlsq);
        % capacitor y bobina, pin C.a.r más alto que L.a.r %
        wire(L.a.l,R.a.l,-5mm,udsq);
        % bobina y resistencia, pin L.a.l más alto que R.a.l %
endfig;
```

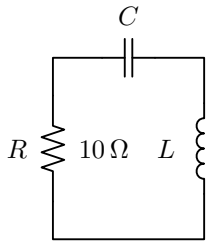


Figura 8: Ejemplo de unión utilizando `wireU`.

3.3. Agregado de texto a los circuitos

En ocaciones es necesario contar con una herramienta que permita añadir texto en los circuitos, porque aveces no basta con la información que se pueda incluir en las «etiquetas» de los símbolos. Para ello MAKECIRC provee dos declaraciones:

`puttext`⟨*pos*⟩`(texto,z);` y `ctext`⟨*pos*⟩`(pin1,pin2,texto,tipo);`

siendo `z` el punto en donde se colocará el texto y ⟨*pos*⟩ la posición que tomará el texto respecto del punto `z`:

⟨*pos*⟩ → ⟨*vacío*⟩`|top|bot|lft|rt|ulft|urt|llft|lrt`

Si el argumento ⟨*pos*⟩ se deja vacío, el texto se colocará exactamente en el punto `z`, de lo contrario se colocará arriba (`top`), abajo (`bot`), a la izquierda (`lft`), derecha (`rt`), arriba-izquierda (`ulft`), arriba-derecha (`urt`), abajo-izquierda (`llft`) o abajo-derecha (`lrt`).

La declaración `ctext` es para centrar texto entre los pines `pin1` y `pin2`, y los tipos disponibles son: `witharrow` para que dibuje una flecha doble desde el `pin1` hasta el `pin2`, y `noarrow` para solamente se centre entre estos dos puntos.

En ambas declaraciones el texto debe ir colocado entre comillas (").

Para mostrar un ejemplo se dibujará un transformador y se le colocarán las etiquetas V_1 y V_2 en el primario y secundario respectivamente, y también el número de espiras N_1 en el bobinado primario y N_2 en el bobinado secundario (véase figura 9).

Código fuente de la figura 9

```
beginfig(9);
        transformer.a(origin,normal,0);

        junction.a(tf.a.ps-(1cm,0),"1a")(top);
        junction.b(tf.a.pi-(1cm,0),"1b")(bot);
        junction.c(tf.a.ss+(1cm,0),"2a")(top);
        junction.d(tf.a.si+(1cm,0),"2b")(bot);

        wire(tf.a.ps,J.a,nsq);
        wire(tf.a.pi,J.b,nsq);
        wire(tf.a.ss,J.c,nsq);
        wire(tf.a.si,J.d,nsq);

        puttext.bot("$N_1$",tf.a.pi);
        puttext.bot("$N_2$",tf.a.si);
        ctext.lft(J.a,J.b,"$V_1$",witharrow);
        ctext(J.c,J.d,"$V_2$",noarrow);
endfig;
```

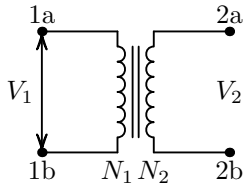


Figura 9: Transformador con texto.

4. Compilación

Cuando todo el trabajo está terminado, llega el momento de compilar el archivo para obtener y visualizar los resultados. Para ello, lo primero que se debe hacer es guardar el archivo, el cual tendrá la extensión `mp`, por ejemplo: `circuitos.mp`.

Luego, se debe compilar **dos veces** con METAPOST para asegurarse de que las etiquetas compuestas por LaTeX salgan correctamente, escribiendo en una ventana de comandos (MSDOS en Windows) lo siguiente:

```
mp circuitos
mp circuitos
```

Esto creará archivos con el mismo nombre que el archivo `mp` pero con extensión numérica: `circuitos.1, circuitos.2,..., circuitos.n`, siendo `n` el número que se especificó en `beginfig` (véase apartado 3.1).

Ahora solo basta incluir los archivos `.n` en un documento de LaTeX, que guardaremos con el nombre `circuitos.tex`:

Documento mínimo de LaTeX para visualizar los resultados

```
\documentclass{article}
\usepackage{graphicx} % parquete para incluir gráficos %

\begin{document}
\centering
\includegraphics{circuitos.1}
\includegraphics{circuitos.2}
⋮
\includegraphics{circuitos.n}
\end{document}
```

y compilarlo con LaTeX y `dvips` para obtener el archivo POSTSCRIPT final:

```
latex circuitos
dvips circuitos
```

Ahora se puede visualizar el archivo `circuitos.ps` con Ghostview.

Otra opción es, si solamente se quieren visualizar los circuitos, utilizar lo siguiente: [3]

```
tex mproof circuitos.1 circuitos.2 circuitos.3
dvips mproof
```

y luego visualizar con Ghosview el archivo `mproof.ps`.

[3] Generalmente el archivo `mproof.tex` viene junto con METAPOST en las distribuciones.