# Módulo TQSDWG

Este módulo permite ler e gravar desenhos DWG TQS. Além dos elementos gráficos padrão manipulados, a biblioteca de desenho permite manipular os "objetos inteligentes de desenho". Estes objetos são implementados dentro do desenho através de DLLs externas com chamadas próprias. Dentre estes objetos, está prevista a manipulação de objetos de cotagem associativa e de Ferros Inteligentes. Os ferros inteligentes, por terem um grande número de funções e propriedades, serão mostrados no capítulo seguinte.

Nas entradas e saídas do sistema TQS, as unidades de medidas de projeto são tratadas. Entretanto, por convenção, todos os valores definidos internamente nos programas têm alguma unidade fixa convencionada. Todas as chamadas ao módulo `TQSDwg`, referentes à geometria usam unidades em centímetros e graus.

## Modo de funcionamento

Um objeto `TQSDwg.Dwg` representa um desenho na memória. Ao criar o objeto, o desenho está vazio:

```
From TQS import TQSDwg

dwg = TQSDwg.Dwg ()
```

A partir daí um desenho pode ser lido do disco para dentro deste objeto, elementos gráficos novos podem ser criados, desenhos misturados, blocos inseridos, o desenho ser salvo, etc. A maioria das operações é feita dentro dos subobjetos, que definem categorias de operação.

Múltiplos desenhos podem ser tratados simultaneamente, cada um contido em um objeto criado por `TQSDwg.Dwg`.

A criação de elementos gráficos se dá através da definição de atributos de desenho e chamadas de funções do objeto `Dwg`.

## Operações de arquivos com TQSDwg.File

O objeto tipo `TQSDwg` contém um subobjeto `file` da classe `TQSDwg.File` usado para operações que afetam um desenho como um tudo. Por exemplo, para abrir um desenho:

```
from TQS import TQSDwg

dwg = TQSDwg.Dwg ()

nomedwg = "TESTE.DWG"

if dwg.file.Open (nomedwg) != 0:

print ("Não abri o desenho ", nomedwg)

file .New ()
```

Inicializa novo desenho para ser salvo com `SaveAs`. Fecha desenho atual sem salvar, se existir algum.

```
file .Open (dwgname)
```

Abre e carrega um desenho existente com nome fornecido.

Retorna: `(0)` se teve sucesso na operação ou `(!=0)` se não conseguir carregar.

```
file .Close ()
```

Fecha um desenho aberto, não salva o conteúdo.

```
file.Save()
```

Salva o desenho com nome atual, não fecha o arquivo.

```
file.SaveAs(dwgname)
```

Salva o desenho atual com um nome fornecido, que pode ser diferente do atual. Se o nome do arquivo não tiver tipo, será salvo com tipo .DWG. Outros tipos que podem ser usados são .DXF e .PDF, para salvar nestes formatos.

No caso do formato PDF, o sistema de plotagem é usado, utilizando os parâmetros de plotagem e tabela de plotagem correspondente. Por isto, é possível que uma assinatura com o nome do cliente seja gerada do lado esquerdo do desenho, e um número de revisão seja somado ao nome do arquivo.

```
file.IsModified()
```

Retorna `(1)` se o desenho atual foi modificado desde que foi aberto ou criado.

```
file.Name()
```

Retorna o nome do desenho atual.

```
file.PurgeBlocks()
```

Elimina todas as definições de blocos não utilizados no desenho.

```
file.PurgeLevels()
```

Elimina todos os níveis não utilizados no desenho.

```
file.LoadColors()
```

Carrega a tabela padrão de cores associada ao sistema/subsistema, que devem ser definidos antes (propriedades `systemId` e `subSystemId`). Veja adiante para que servem estas propriedades.

# Desenho com TQSDwg.Draw

O objeto do tipo `TQSDwg` tem o subobjeto `draw` da classe `TQSDwg.Draw` com o objetivo de criação de elementos gráficos. Por exemplo, para desenhar uma linha entre dois pontos, fazemos:

```
dwg.draw.Line (100., 100., 200., 200.)
```

As rotinas de desenho não permitem a construção direta de objetos gráficos; em vez disto são usadas funções e propriedades e os objetos gráficos são criados dentro de `TQSDwg`. Algumas propriedades de objetos gráficos são definidas antes como valores atuais, e usadas automaticamente sempre que um objeto é criado. Por exemplo, nível, estilo e cor. No exemplo, acima, uma linha foi criada nas coordenadas fornecidas, usando nível, estilo e cor atual.

## Propriedades de TQSDwg.Draw

Devem ser definidas antes dos elementos gráficos, e se tornam atuais para todos os elementos criados depois.

```
draw.level
```

Nível atual de desenho, sempre na forma numérica

```
draw.style
```

Estilo atual de desenho ou (-1) para usar o estilo associado ao nível

```
draw.color
```

Cor atual de desenho (1..255) ou (-1) para usar a cor associada ao nível de desenho.

## Métodos de TQSDwg.Draw

```
draw.Line(x1, y1, x2, y2)
```

Uma linha entre 2 pontos

```
draw .Rectangle (x1, y1, x2, y2)
```

Um retângulo entre 2 pontos – canto esquerdo inferior e direito superior

```
draw .PolyStart ()
```

Inicia a acumulação de pontos por `PolyEnterPoint`. Os pontos acumulados podem ser usados para desenhar linhas poligonais pelas rotinas `Polyline`, `PolylineFilled` e `PolylineCurve`.

```
draw .PolyEnterPoint (x, y)
```

Acumula um ponto de uma linha poligonal. Este ponto fará parte da poligonal usada pelas rotinas `Polyline`, `PolylineFilled` e `PolylineCurve`.

```
draw .PolyEnterXY (xy)
```

Acumula um vetor de uma polyline, no formato `[ [x1,y1], [x2,y2], ...]`. Usa a rotina `PolyEnterPoint` . Os pontos acumulados podem ser usados em `Polyline`, `PolylineFilled` e `PolylineCurve`.

```
draw .Polyline ()
```

Desenha linha poligonal com os pontos acumulados por `PolyEnterPoint`

```
draw .PolylineFilled ()
```

Linha poligonal preenchida com os pontos acumulados por `PolyEnterPoint.`

```
draw .PolylineCurve ()
```

Linha poligonal aproximada por spline cúbica com os pontos acumulados por `PolyEnterPoint`.

```
draw .Arc (xc, yc, r, angi, angf)
```

Arco no sentido anti-horário com centro `xc,yc` raio `r` e ângulos inicial `angi` e final `angf em` graus.

```
draw .Circle( xc, yc, r)
```

Círculo com centro em `xc,yc` e raio `r`.

```
draw .Text (x, y, h, ang, text)
```

Têxto com ponto de inserção em `x,y`, altura `h` e ângulo `ang` em graus. O string do texto é definido em `text`.

```
draw .BlockOpen (blockname, xbase, ybase)
```

Criação de um novo bloco interno. Todos os elementos gráficos criados a seguir entrarão dentro do bloco, até que o bloco seja fechado por `draw.BlockClose`. `blockname` define o nome do bloco e `xbase,ybase` as coordenadas que coincidirão com o ponto de inserção, quando o bloco for inserido no desenho.

```
draw .BlockClose ()
```

Termina a criação de um bloco, aberto em `draw.BlockOpen.`

```
draw .BlockInsert (blockname, x, y, escx, escy, ang)
```

Inserção de um bloco interno pré-definido de nome `blockname`, com ponto de inserção `x,y`, escala X `escx`, escala Y `escy`, e ângulo `ang` em graus.

```
draw .BlockLoadFromDwg (dwgname)
```

Carrega um arquivo de desenho e transforma em bloco interno com o mesmo nome do desenho. Retorna `(0)` se completou a operação ou `(1)` se houve um erro.

```
draw .DwgMix (dwgname, deltax, deltay)
```

Carrega e mistura o desenho `dwgname` no desenho com o atual, aplica um deslocamento `deltax,deltay` no desenho carregado. Retorna `(0)` se completou a operação, ou `(1)` se houve um erro.

```
draw .XrefInsert (dwgname, deltax, deltay, ang)
```

Insere uma referência externa com nome `dwgname`, deslocamento `deltax,deltay` e ângulo `ang`. Retorna `(0)` se completou a operação ou `(1)` se houve um erro.

# Cotagens com TQSDwg.dim

Cotagens são geradas como objetos de editor gráfico tipo "`IPOCOT`", tratadas por uma DLL diferente da de desenho. Mas a criação destes objetos foi inteiramente embutida dentro da `TQSDwg`.

Assim como nos demais elementos gráficos, a criação dos objetos é feita por funções, e certos atributos de cotagem são definidos como "*atuais*",e são implícitos na criação de cotagens novas.

O objeto do tipo `TQSDwg` tem o subobjeto `dim` da classe `TQSDwg.Dim` para a criação de cotagens. Por exemplo, para uma cotagem horizontal entre dois pontos e passando por um terceiro, temos uma chamada como esta:

```
dwg.dim.DimHorizontal (0, 0, 500, 0, 0, -50)
```

As cotagens são em cm, mas as alturas de texto e dimensões de símbolos são todos multiplicados pela escala atual de desenho. Em um desenho com escala 1:50 e um texto de cotagem de altura 0.2, a altura real do texto será de 0.2*50=10 cm.

## Atributos de cotagem

São os atributos que valem para as cotagens feitas a seguir. Quando não definidos, são assumidos os valores do arquivo de critérios de cotagem.

```
dim .dimtxt
```

Altura do têxto de cotagem.

```
dim .dimexe
```

Extensão da linha de chamada.

```
dim .dimdle
```

Extensão da linha de cotagem.

```
dim .dimexo
```

Folga na linha de chamada.

```
dim .dimtsz
```

Tamanho do símbolo de cotagem.

```
dim .dimlfc
```

Multiplicador de comprimentos.

```
dim .idmniv
```

Nível geral de cotagem.

`dim .idmnic`

Nível da linha de cotagem.

`dim .idmnil`

Nível da linha de chamada.

`dim .idmnib`

Nível do símbolo de cotagem.

`dim .idmcim`

(1) Se têxto abaixo da linha de cotagem

`dim .idmar5`

(1) Se medidas arredondadas de 5 em 5.

`dim .dimblk`

Nome do bloco de cotagem (TICK/DOT/ARROW são pré-definidos).

`dim .dimse1`

(1) Se suprime linha de chamada

`dim .idmcot`

(1) Se suprime linha de cotagem

`dim .dimtxu`

Texto manual de cotagem.

## Constantes de cotagem

`TQSDwg .IDHOR`

Identifica cotagem horizontal.

`TQSDwg .IDVER`

Identifica cotagem vertical.

`TQSDwg .IDANG`

Identifica cotagem alinhada.

`TQSDwg .IDINC`

Identifica cotagem inclinada.

## Métodos de cotagem

`dim .Dim3P (itipo, x1, y1, x2, y2, x3, y3)`

Cotagem do tipo `itipo=TQSDwg.IDHOR/IDVER/IDANG/IDINC` entre os pontos `x1,y1` e `x2,y2` e com linha de cotagem passando por `x3,y3`.

`dim .DimHorizontal (x1, y1, x2, y2, x3, y3)`

Cotagem horizontal entre os pontos `x1,y1` e `x2,y2` e com linha de cotagem passando por `x3,y3`.

`dim .DimVertical (x1, y1, x2, y2, x3, y3)`

Cotagem vertical entre os pontos `x1,y1` e `x2,y2` e com linha de cotagem passando por `x3,y3`.

```
dim .DimAligned (x1, y1, x2, y2, x3, y3)
```

Cotagem alinhada com os pontos `x1,y1` e `x2,y2` e com linha de cotagem passando por `x3,y3`.

```
dim .DimInclined (x1, y1, x2, y2, x3, y3, ang)
```

Cotagem inclinada pelo ângulo `ang` em graus, entre os pontos `x1,y1` e `x2,y2` e com linha de cotagem passando por `x3,y3`.

```
dim .DimContinue (x, y)
```

Continua a última cotagem com ponto adicional.

```
dim .DimRadius (x1, y1, x2, y2)
```

Cotagem de raio de arco ou círculo passando por `x1,y1 e x2,y2.`

```
dim .DimDiameter (x1, y1, x2, y2)
```

Cotagem de diâmetro de raio ou círculo passando por `x1,y1 e x2,y2.`

```
dim .DimAngular (x1, y1, x2, y2, x3, y3, x4, y4, xcota, ycota)
```

Cotagem de ângulo entre as retas ( `x1,y1 => x2,y2`) e ( `x3,y3 => x4,y4`) passando pelo ponto `xcota,ycota.`

```
dim .DimNote ()
```

Desenho da linha com flecha de nota. Use `PolyStart ()` e `PolyEnterPoint (x, y)` para a definição dos pontos. Será desenhada uma linha poligonal com uma flecha na ponta.

# Estados de desenho com TQSDwg.settings

O objeto do tipo `TQSDwg` tem o subobjeto `settings` da classe `TQSDwg.Settings` com o objetivo de controlar modos de funcionamento do desenho que afetam sua edição gráfica. Tratam-se de propriedades que podem ser lidas ou gravadas. Por exemplo, para definir a escala de desenho 1:50 fazemos:

```
dwg.settings.scale = 50.
```

## Constantes que definem o número do sistema

Mostraremos adiante quando usar o número do sistema. Um sistema é uma das aplicações contidas dentro do TQS. As constantes são:

```
TQSDwg .IEDOUT
```

Genérico

```
TQSDwg .IEDFOR
```

Formas

```
TQSDwg .IEDLAJ
```

Lajes

```
TQSDwg .IEDFUS
```

Fundações - sapatas

```
TQSDwg .IEDFUE
```

Fundações - blocos

```
TQSDwg .IEDFUT
```

Fundações - tubulões

```
TQSDwg  .IEDVIG
```

Vigas

```
TQSDwg  .IEDPIL
```

Pilares

```
TQSDwg  .IEDTAB
```

Armação genérica AGC & DP

```
TQSDwg  .IEDMAD
```

Madeira

```
TQSDwg  .IEDCORB
```

G-Bar

```
TQSDwg  .IEDALV
```

Alvenaria Estrutural

```
TQSDwg  .IEDPOR
```

Pórtico TQS

```
TQSDwg  .IEDGRE
```

Grelha TQS

```
TQSDwg  .IEDPLW
```

IGV

```
TQSDwg  .IEDESC
```

Escadas

```
TQSDwg  .IEDISE
```

Interação solo-estrutura

```
TQSDwg  .IEDPRE
```

Pré-moldados

```
TQSDwg  .IEDRER
```

Reservatórios e elementos especiais

```
TQSDwg  .IEDPAR
```

Paredes de concreto

## Propriedades de sistema

Todo desenho recebe as propriedades de sistema e subsistema:

```
settings  .systemId
```

Número do sistema

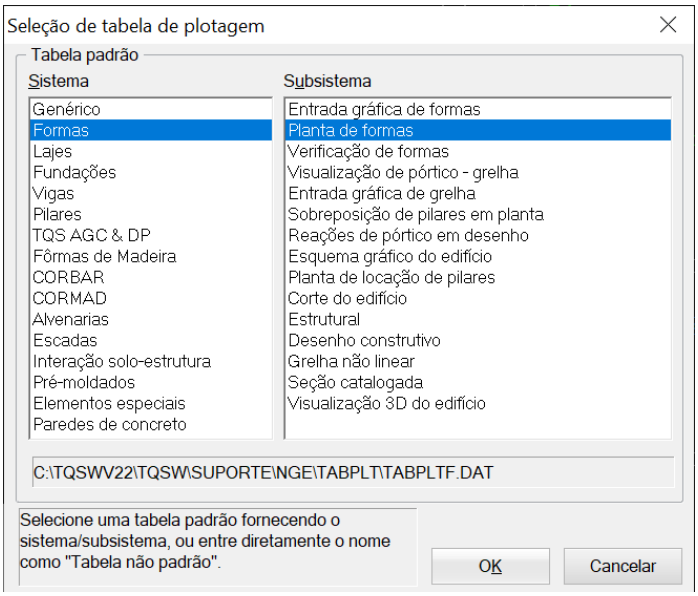
```
settings  .subSystemId
```

Número do subsistema

O arquivo TQSW\SUPORTE\NGE\GAPLIC.DAT relaciona o par (sistema, subsistema) com:

Menu do editor gráfico associado
Nome da tabela de cores de níveis
Desenho com armaduras ou não

Por exemplo, um desenho de planta de formas pertence ao sistema Formas ( `TQSDwg .IEDFOR` ) e subsistema 2, será associado com o menu de edição gráfica EAGFOR.MEN e tabela de cores DESFOR.COR. Ao dar um duplo-clique sobre um desenho de formas no gerenciador, o editor gráfico lê primeiro o desenho, determina o sistema e subsistema, e chama o menu EAGFOR.MEN associado. O número do subsistema pode ser obtido no gerenciador, menu de plotagem, "Edição de critérios, Tabelas, Tabelas de plotagem". Ao selecionar à esquerda o sistema, vemos os subsistemas à direita:




O número do subsistema é a ordem do item selecionado à direita, começando em 1.

## Outras propriedades de estado

`settings .scale`

Retorna valor que divide uma unidade e resulta em centímetros de plotagem

`settings .xBase`

Retorna X do ponto que coincidirá com o cursor se este desenho for inserido dentro de outro como um bloco, referência ou mistura de desenhos.

`settings .yBase`

Retorna Y do ponto que coincidirá com o cursor se este desenho for inserido dentro de outro como um bloco, referência ou mistura de desenhos.

`settings .blockReadDelimiterMode`

(1) Se os delimitadores `DWGTYPE_BLOCK` e `DWGTYPE_BLOCKEND` de início e fim de bloco devem ser lidos

`settings .blockReadElementsMode`

(1) Para ler os elementos dentro do bloco inserido ou (0) apenas para os delimitadores `DWGTYPE_BLOCK` e `DWGTYPE_BLOCKEND`.

`settings .levelsReadMode`

(1) Para ler apenas elementos em níveis ligados ou (0) para todos os elementos

`settings` **`.captureMode`**

(0) Captura definida no gerenciador (1) desligado (2) todos ligados (3) ligados somente nos níveis com captura

`settings` **`.gridMode`**

(0) Captura (1) Grade (2) desligado

`settings` **`.gridOriginX`**

Origem X da grade

`settings` **`.gridOriginY`**

Origem Y da grade

`settings` **`.gridSpacingX`**

Espaçamento X da grade

`settings` **`.gridSpacingY`**

Espaçamento Y da grade

`settings` **`.gridAngle`**

Ângulo da grade em graus

`settings` **`.handle_pedmv`**

Handle de desenho para uso em outras bibliotecas. É o valor passado diretamente para as DLLs C++ TQS.

`settings` **`.originalColorsMode`**

(1) se o desenho deve ser plotado com cores originais e não da tabela de plotagem

`settings` **`.comment`**

Comentário associado ao desenho

`settings` **`.revComment`**

Comentários da revisão do desenho

`settings` **`.blueprintMode`**

(1) Modo de visualização de plantas

# Limites de desenho com TQSDwg.limits

O objeto do tipo `TQSDwg` tem o subobjeto `limits` da classe `TQSDwg.Limits` com o objetivo de ler e calcular os limites do desenho atual. Por exemplo, para obter os limites do desenho atual, fazemos:

```
xmin, ymin, xmax, ymax = dwg.limits.DwgLimits()
```

## Métodos de TQSDwg.limits

`limits` **`.FileLimits`** `(dwgname)`

Retorna os limites de um desenho sem carregá-lo na memória.

Retorna: `xmin, ymin, xmax, ymax, istat`

`istat == (0)` se a operação foi bem sucedida ou `(1)` se não leu o arquivo

`limits` **`.DwgLimits`** `()`

Le os limites do desenho atual.

Retorna: `xmin, ymin, xmax, ymax`

`limits .UpdateLimits ()`

Recalcula os limites do desenho atual.

`limits .WindowLimits()`

Lê a janela inicial do desenho nos editores gráficos.

Retorna: `xmin, ymin, xmax, ymax.`

`limits .DefineWindowLimits (xmin, ymin, xmax, ymax)`

Define a janela inicial de desenho nos editores gráficos.

# Leitura de desenho com TQSDWg.iterator

O objeto do tipo `TQSDwg` tem o subobjeto `iterator` da classe `TQSDwg.Iterator` com o objetivo de ler e interpretar o conteúdo de um desenho. Por exemplo, uma iteração de leitura por todos os objetos seria como esta:

```
dwg.iterator.Begin()
while True:
itipo = dwg.iterator.Next()
if itipo == TQSDwg.DWGTYPE_EOF:
break
.....
```

A iteração pelo desenho não retorna objetos, mas valores simples como propriedades da classe `Iterator()`. As funções são usadas para controlar a iteração.

## Tipos de elemento de desenho

Todo elemento gráfico tem um tipo, definido por uma das constantes:

`TQSDwg .DWGTYPE_EOF`

Fim de arquivo

`TQSDwg .DWGTYPE_LINE`

Linha

`TQSDwg .DWGTYPE_TEXT`

Texto

`TQSDwg .DWGTYPE_POLYLINE`

Linha poligonal, preenchida ou não.

`TQSDwg .DWGTYPE_BLOCK`

Inserção de bloco

`TQSDwg .DWGTYPE_BLOCKEND`

Fim de bloco

`TQSDwg .DWGTYPE_BLOCKBEGIN`

Inicio de bloco

`TQSDwg` **`.DWGTYPE_CIRCLE`**

Circulo

`TQSDwg` **`.DWGTYPE_ARC`**

Arco

`TQSDwg` **`.DWGTYPE_CURVE`**

Curva

`QSDwg` **`.DWGTYPE_OBJECT`**

Objeto de desenho (como cotagem associativa ou Ferro Inteligente)

`TQSDwg` **`.DWGTYPE_OBJECTEND`**

Fim de leitura de um objeto de desenho

## Propriedades de um objeto gráfico lido

As propriedades a seguir ficam disponíveis conforme o tipo de elemento gráfico ( `TQSDwg.DWGTYPE_xxx`), após a chamada da função `iterator.Next()`. Estas propriedades são somente de leitura.

`iterator` **`.itype`**

Tipo `TQSDwg.DWGTYPE_xxxx` do elemento gráfico lido.

`iterator` **`.elementName`**

Nome do tipo do elemento gráfico lido.

`iterator` **`.x1`**

X1 de um elemento gráfico

`iterator` **`.y1`**

Y1 de um elemento gráfico

`iterator` **`.x2`**

X2 de um elemento gráfico

`iterator` **`.y2`**

Y2 de um elemento gráfico

`iterator` **`.xySize`**

Número de pontos de um elemento gráfico

`iterator` **`.GetPolylinePt`** `(ipt)`

Retorna `x,y` de um elemento gráfico com `ipt` entre `0` e `iterator.xySize-1.`

`iterator` **`.xy`**

Leitura da matriz de pontos inteira no formato `xy [[0][1]]`

`iterator` **`.isFilled`**

`(1)` se a linha poligonal atual é do tipo preenchida.

`iterator` **`.xc`**

X do centro de arco ou círculo.

```
iterator .yc
```

Y do centro de arco ou círculo.

```
iterator .radius
```

Raio de um arco ou círculo.

```
iterator .startAngle
```

Ângulo inicial de um arco em graus.

```
iterator .endAngle
```

Ângulo final de um arco em graus.

```
iterator .textHeight
```

Altura de texto

```
iterator .textAngle
```

Ângulo de texto, em graus

```
iterator .textLength
```

Número de caracteres do texto.

```
iterator .text
```

String de texto.

```
iterator .blockName
```

Nome do bloco inserido.

```
iterator .xScale
```

Escala X do bloco inserido.

```
iterator .yScale
```

Escala Y do bloco inserido.

```
iterator .insertAngle
```

Ângulo do bloco inserido em graus.

```
iterator .level
```

Nível do elemento

```
iterator .color
```

Cor (0..255) do elemento, possivelmente herdado do nível.

```
iterator .colorRGB
```

Cor RGB opcional do elemento

```
iterator .style
```

Estilo do elemento, possivelmente herdado do nível.

```
iterator .levelLock
```

(1) se o elemento é travado no nível.

`iterator` **`.captureable`**

(1) se o elemento é capturável.

`iterator` **`.hasPlotData`**

(1) se informações de plotagem independentes disponíveis

`iterator` **`.plotPen`**

Pena de plotagem se disponível.

`iterator` **`.plotWeight`**

Peso de plotagem se disponível.

`iterator` **`.plotStyle`**

Estilo de plotagem se disponível.

`iterator` **`.plotFont`**

Fonte de plotagem se disponível.

`iterator` **`.plotHatch`**

Hachura de plotagem se disponível.

`iterator` **`.objectName`**

Nome do objeto de desenho, se objeto inteligente do editor.

`iterator` **`.objectPointer`**

Apontador do objeto de desenho, se objeto inteligente do editor.

`iterator` **`.smartRebar`**

Objeto `SmartRebar` se o objeto extraído é um Ferro Inteligente, ou `None`.

`iterator` **`.inBlock`**

(1) Se o elemento gráfico lido está dentro de um bloco.

`iterator` **`.inXref`**

(1) se o elemento gráfico lido está dentro de uma referência externa.

`iterator` **`.isOpenObject`**

(1) se o elemento gráfico lido está dentro de um objeto inteligente.

## Métodos de TQSDwg.iterator

`iterator` **`.Begin`** `()`

Posiciona o desenho para leitura no início.

`iterator` **`.Next`** `()`

Lê o próximo elemento, retorna o tipo `iterator.DWGTYPE_xxxx`, com `DWGTYPE_EOF` no final.

`iterator` **`.GetElementReadPosition`** `()`

Retorna a posição `addr` do último elemento lido

```
iterator.SetPosition (addr)
```

Posiciona para leitura na posição `addr`.

```
iterator .GetElementWritePosition ()
```

Retorna a posição do último elemento gráfico gravado.

```
iterator .GetReadPosition ()
```

Retorna a posição do próximo elemento a ser lido.

```
iterator .GetWritePosition ()
```

Retorna a posição do próximo elemento a ser gravado

```
iterator .SetBlockPosition( blockname)
```

Posiciona para a leitura dos elementos internos ao bloco `blockname`. Retorna: `(0)` se teve sucesso ou `(1)` se o bloco não existe.

```
iterator .CopyReadAtribbutes ()
```

Copia os atributos do último elemento lido para uso no próximo elemento a ser gravado.

# Tabela de níveis com TQSDwg.levelstable

O objeto do tipo `TQSDwg` tem o subobjeto `levelstable` da classe `TQSDwg.LevelsTable` que permite listar e alterar algumas propriedades da tabela de níveis de um desenho. Por exemplo, para atribuir a cor 4 ao nível 3, fazemos:

```
dwg.levelstable.SetColor(3, 4)
```

```
levelstable .count
```

Número total de níveis numerados de 0 a N-1.

```
levelstable .IsDefined(level)
```

Retorna (1) se o nível está definido

```
levelstable .GetStyle (level)
```

Retorna o estilo associado ao nível ou (-1) para estilo padrão

```
levelstable .SetStyle (level, ival)
```

Define o estilo associado ao nível ou (-1) para estilo padrão

```
levelstable .IsOn (level)
```

Retorna estado ligado (1) ou desligado (0) de um nível

```
levelstable .TurnOn (level, ival)
```

Define o estado ligado (1) ou desligado (0) de um nível

```
levelstable .GetColor (level)
```

Retorna a cor (0..255) de um nível

```
levelstable .SetColor (level, ival)
```

Define a cor (0..255) de um nível

```
levelstable .GetLock (level)
```

Retorna o estado de trava de um nível (1) sim (0) não

```
levelstable .SetLock (level, ival)
```

Define o estado de trava de um nível (1) sim (0) não

```
levelstable .GetCapture (level)
```

Retorna o estado de captura de um nível (1) sim (0) não

```
levelstable .SetCapture (level, ival)
```

Define o estado de captura de um nível (1) sim (0) não

```
levelstable .Create (level)
```

Cria um nível numérico ou alfanumérico e retorna o correspondente numérico para uso em outros métodos

```
levelstable .Name (level)
```

Retorna o nível em formato alfanumérico

# Tabela de blocos com TQSDwg.blockstable

O objeto do tipo `TQSDwg` tem o subobjeto `blockstable` da classe `TQSDwg. BlocksTable` que permite listar a biblioteca de blocos interna.

```
blockstable .count
```

Retorna o número de blocos da biblioteca interna

```
blockstable .Name (index)
```

Retorna o nome do bloco com índice entre 0 e `blockstable.count-1.`

```
blockstable .IsDefined (name)
```

Retorna (1) se o bloco `name` está definido

# Referências externas e o TQSDwg.xreference

Referências externas são codificadas no DWG como blocos com nome especial, contendo um retângulo envolvente e um texto com o nome do arquivo que representa a referência. A inserção de uma referência externa é feita pela função `draw. XrefInsert`. A listagem e o controle de alguns parâmetros são feitos pelo objeto `xreference` da classe `XReference`.

```
xreference .count
```

Retorna o número de referências externas definidas e inseridas.

```
xreference .Read (index)
```

Lê os dados de uma referência externa efetivamente inserida no desenho `(index==0.. xreference.count-1)`

Retorna: `dwgname` (o desenho associado), `ion` (1 se referência ligada) e `blockname` (nome do bloco que representa a referência no DWG).

```
xreference .TurnOn (index, ival)
```

Defina ( `ival==1`) para ligar a visualização de uma referência externa.

```
index==0.. xreference.count-1)
```

```
xreference .Disable (ival)
```

Defina (0) para habilitar ou (1) para inibir a interpretação de referências externas no desenho.

# Edição de elementos com TQSDwg.edit

Elementos gráficos podem ser modificados depois de criados. Para isto é necessário de um endereço de objeto gráfico que pode ser obtido com o `TQSDwg.iterator`, ou dentro do editor gráfico, com as rotinas interativas de seleção do editor. Por exemplo,

```
addr = iterator.GetElementReadPosition()
```

O objeto do tipo `TQSDwg` tem o subobjeto `edit` da classe `TQSDwg.Edit` que permite editar elementos gráficos. Nas rotinas a seguir, a variável `addr` corresponde a um endereço mostrado acima.

```
edit .ModifyLevel (addr, level)
```

Modifica o nível de um elemento.

```
edit .ModifyColor (addr, icolor)
```

Modifica a cor (0..255) de um elemento.

```
edit .ModifyStyle (addr, istyle)
```

Modifica o estilo de um elemento.

```
edit .Erase (addr)
```

Apaga um elemento, retorna (!=0) se erro.

```
edit .Recover (addr)
```

Recupera um elemento apagado, retorna (!=0) se erro.

```
edit .ModifyPoint (addr, index, x, y)
```

Modifica as coordenadas de um ponto de um elemento

```
index== 0..iterator. xySize-1
```

Retorna (!=0) se erro.

```
edit .ModifyText (addr, h, ang)
```

Modifica altura `h` e o ângulo `ang` em graus de um texto. Para modificar o seu conteúdo é necessário apagar e criar outro.

```
edit .ModifyArc (addr, r, angi, angf)
```

Modifica dados de um arco: raio, ângulo inicial e ângulo final, em graus.

```
edit .ModifyCircle (addr, r)
```

Modifica o raio de um círculo.

```
edit .ModifyBlock (addr, escx, escy, ang)
```

Modifica dados de um bloco inserido: escala X e Y e ângulo em graus. retorna (!=0) se erro.

```
edit .Move (addr, dx, dy)
```

Movimenta um elemento qualquer por um deslocamento dx,dy. Retorna (!=0) se erro.

# Leitura da tabela de plotagem associada com TQSDwg.plotting

O objeto do tipo `TQSDwg` tem o subobjeto `plotting` da classe `TQSDwg. Plotting` que permite listar o conteúdo da tabela de plotagem associada a um desenho.

```
plotting .LoadPlottingTable ()
```

Torna disponíveis os dados de tabela de plotagem associados a este desenho.

Retorna: o nome da tabela e o status de leitura (!=0) se erro.

```
plotting .AttributeRead (nivel)
```

Lê os atributos de um determinado nível da tabela de plotagem

Retorna:

`Pena` Índice da pena

`peso` Índice do peso

`estilo` Índice do estilo

`hachura` Índice da hachura

`fonte` Índice do fonte

`titulo` Título do nível

# Programa TSTDwg.py de teste do TQSDwg

Este programa testa a quase totalidade de recursos do módulo `TQSDwg`, com exceção dos ferros inteligentes. O programa abre uma janela de mensagens TQS e lista os resultados nesta janela. No final, é gravado o desenho `TESTE1.DWG`, com o resultado do teste.

O desenho é feito na instância de objeto `dwg`:

```
Dwg = TQSDwg.Dwg()
```

## Função TestarDesenho ()

Ela começa atribuindo sistema `TQSDwg.IEDFOR` e subsistema 2 (planta de formas), e depois testa a maioria dos atributos e funções deste objeto: desenho de linhas, linhas poligonais, arcos, círculo, curva, poligonal preenchida, criação e inserção de blocos, textos, mistura de desenho e inserção de referência externa.

Tanto o desenho misturado quanto a referência externa são obtidos do desenho `\TQSW\USUARIO\TESTE\COBERTURA.DWG`, que é distribuído com o TQS. A pasta onde está o desenho é obtida por uma função de TQSUtil.

## Função TestarEdit()

A função desenha alguns elementos e obtém seu endereço para alterá-los em seguida, demostrando a capacidade de edição da biblioteca.

## Função TestarCotagem()

Testa diversas propriedades e funções de cotagem.

```
Teste de cotagens
Altura do texto de cotagem 0.31
Extensão de linha de cotagem 0.32
Extensão da linha de chamada 0.33
Folga da linha de cotagem 0.34
....
```

## Função TestarLeitura()

Esta função itera na leitura de todos os elementos de desenho, incluindo a mistura de desenhos e a referência externa, e lista os atributos de todos os elementos gráficos. Isto resulta em uma listagem extensa, onde podem ser verificados os atributos de cada objeto gráfico.

```
Elemento [Linha] (tipo 1) em bloco 0 xref 0 obj 0
nivel 0 cor 7/000000 estilo 0 ipreench 0 lock 0 captura 0
plotdata 0 pen 0 peso 0 estilo 0 fonte 0 hatch 0
pt1 0,0 pt2 500,0
.....
```

## Função TestarGlobais()

Testa a leitura e atribuição de algumas variáveis de estado definidas em TQSDwg.settings.

```
Teste de variaveis globais
Aplicação TQS ..... 2/2
Subaplicação TQS..... 2/2
Escala ..... 50/75
....
```

## Função TestarLerNiveis()

Aqui listamos a tabela de níveis de desenho.

```
Tabela de níveis
Nível [ ] Def Estilo Ligado Cor Lock Captura
0 [ 0] 1 0 1 7 0 0
1 [ 1] 1 0 1 15 0 1
....
```

## Função TestarLerBlocos()

Listamos todos os blocos inseridos no desenho.

```
Tabela de blocos (4)
Bloco [ X] definido 1
Bloco [ $LAJPRE] definido 1
Bloco [ DOT2] definido 1
Bloco [ $REFE0001] definido 1
```

## Função TestarXRef()

Lista as referências externas inseridas. Somente uma neste exemplo.

```
Tabela de referências externas (1)
Referência [C:\TQSWV22\TQSW\USUARIO\TESTE\COBERTURA.DWG] ligada 1 bloco [
$REFE0001]
```

## Função TestarLerTabPlt()

Mostra todos os atributos da tabela de plotagem associada a este desenho.

```
Teste de tabela de plotagem [C:\TQSWV22\TQSW\SUPORTE\NGE\TABPLT\TABPLTF.DAT]

ível Pena Peso Estilo Hachura Fonte Título

0 0 0 0 0 0 [Uso geral]

1 3 0 0 0 0 [Contorno de vigas]

2 6 0 0 6 0 [Pilares que nascem]
```