



# IM

Biblioteca de Acesso à Arquivos de Imagens Bitmaps  
Versão 2.3

---

Data de Impressão: 28-Oct-99

---

Tecgraf/PUC-Rio Rua Marquês de São Vicente, 225 Prédio Velloso - CEP 22453-900 - Rio de Janeiro - Brasil  
Tel. +55 21 529-9424 - Fax. +55 21 259-2232 - E-mail: [tecg@tecgraf.puc-rio.br](mailto:tecg@tecgraf.puc-rio.br) - URL: <http://www.tecgraf.puc-rio.br>

## Sumário

|   |    |
|---|----|
| <i>Visão Geral</i> .....  | 3  |
| <i>Histórico</i> .....  | 3  |
| <i>Pendências</i> .....   | 4  |
| <i>Comunicação de Problemas</i> .....                                 | 4  |
| <i>Agradecimentos</i> .....   | 4  |
| <i>Equipe de Desenvolvimento</i> .....                                | 5  |
| <i>Sobre o IM Online</i> .....  | 5  |
| <b>GUIA</b> .....   | 6  |
| <i>Acesso às Imagens</i> .....  | 6  |
| <i>Exemplos</i> .....   | 6  |
| <i>Variáveis e Tipos</i> .....  | 6  |
| <i>Inicialização</i> .....  | 6  |
| <i>Leitura e Escrita</i> .....  | 7  |
| <i>Cores</i> .....  | 7  |
| <i>Uso com a Biblioteca CD - Canvas Draw</i> .....                    | 7  |
| <i>Implementando um Novo Formato</i> .....                            | 7  |
| <i>Acessando Arquivos Virtuais</i> .....                              | 8  |
| <b>FUNÇÕES</b> .....  | 9  |
| <i>Cor</i> .....  | 9  |
| <i>Informação</i> .....   | 9  |
| <i>I/O</i> .....  | 9  |
| <i>Conversão</i> .....  | 10 |
| <i>Tamanho</i> .....  | 10 |
| <i>Controle</i> .....   | 11 |
| <b>FORMATOS</b> .....   | 12 |
| <i>BMP (Windows 3.x ou OS/2 1.x Device Independent Bitmaps)</i> ..... | 12 |
| <i>PCX (ZSoft PC-PaintBrush Picture)</i> .....                        | 12 |
| <i>GIF (Compuserve Graphics Interchange Format)</i> .....             | 12 |
| <i>TIF (TIFF, Tagged Image File Format)</i> .....                     | 12 |
| <i>SGI (BW/RGB/RGBA, Silicon Graphics)</i> .....                      | 13 |
| <i>RAS (Sun RasterFile)</i> .....                                     | 13 |
| <i>JPG (JPEG, Join Photographic Experts Group)</i> .....              | 13 |
| <i>LED (IUP/LED, Linguagem de Especificação de Diálogos)</i> .....    | 13 |
| <i>TGA (True Vision Targa)</i> .....                                  | 14 |
| <i>Quadro Comparativo</i> .....                                       | 14 |
| <i>Comentários</i> .....  | 15 |
| <b>IDENTIFICADORES</b> .....  | 16 |
| <i>File Format</i> .....  | 16 |
| <i>Format Compression</i> .....                                       | 16 |
| <i>Image Types</i> .....  | 16 |
| <i>Error Codes</i> .....  | 16 |

# IM

## Biblioteca de Acesso à Arquivos de Imagens Bitmaps Versão 2.3

### Visão Geral

O objetivo desta biblioteca é ler e escrever imagens bitmaps em formatos raster, tais como BMP, PCX, GIF, TIF, etc... É dada única importância à imagens RGB de 24 bits por *pixel* (8 bits para cada componente) e imagens de 8 bits por *pixel* indexadas por uma palette. A palette é uma tabela de valores RGB com no máximo 256 elementos. Durante a leitura de determinados formatos são aceitos, dentro do possível, outros tipos de imagem, que são automaticamente convertidos para 24 ou 8 bpp.

A biblioteca foi gerada e testada nas seguintes plataformas: Linux (PC), AIX (RISC), IRIX (Silicon Graphics), Windows (PC), SunOS (SUN), e DOS (PC). No ambiente UNIX é usado o compilador GNU C (gcc), no DOS usa-se o Watcom C/C++ com DOS4GW, e no Windows são geradas bibliotecas para Visual C++ 2.0, 4.0 e 5.0, Borland C++ 4.5 e 5.0, e Watcom C/C++ 10.6. No ambiente Windows é gerada também uma biblioteca dinâmica (DLL).

A IM possui total compatibilidade com imagens cliente do CD (Canvas Draw). Fornecendo assim armazenamento persistente para imagens cliente.

A adoção de formatos de arquivo conhecidos e bem difundidos, e a independência do formato escolhido tornam a IM um poderosa biblioteca de acesso a arquivos de imagem.

### Histórico

**Junho de 98** - Versão 2.3. Corrigido função close do driver de acesso a arquivo em memória. Atualizada biblioteca JPEG para 6b. Também foi corrigido um problema com a leitura de alguns arquivos JPEG.

**Novembro de 97** - Versão 2.2. Modificada a definição da *callback* de contador para informar em um parâmetro, qual o tipo de acesso sendo realizado, leitura ou escrita. Definido o tipo **imCallback** para facilitar o *type casting* ao usar a função **imRegisterCallback**. Corrigido um problema com o makefile no UNIX que gerava erros no link em algumas plataformas.

**Outubro de 97** - Versão 2.1. Corrigido um problema de liberação de memória interna durante a leitura de imagens **Map** em arquivos **TIFF**. A conversão **RGB para Map** agora é feita usando o algoritmo implementado pela LibJPEG. O algoritmo da **imResize** foi melhorado para o caso em que o tamanho está sendo reduzido e não ampliado. Corrigido um problema com as funções **imImageInfo** e **imFileFormat** quando o arquivo não fornecido não está em um formato reconhecido pela IM, havia um erro no formato **TGA** que fazia com essas funções acessassem uma área de memória inválida.

**Setembro de 97** - Versão 2.0, a biblioteca foi praticamente reescrita para implementar uma nova estrutura que permita uma maior flexibilidade a mesma,

simplificando o acréscimo de novos formatos. Os formatos **TGA**, **PCL**, **JPEG** e **LED** foram acrescentados a lista de formatos suportados e foram acrescentadas novas funções: **imMap2RGB**, **imRGB2Gray**, **imMap2Gray**, **imResize**, **imStretch**.

**Junho de 96** - Versão 1.1, pequenas correções para melhorar a portabilidade. Códigos de retorno foram modificados. Foram criados identificadores para códigos de retorno -definidos. Manual on-line concluído.

**Outubro de 95** - Versão 1.0 concluída

## Pendências

Concluir os formatos XBM e PCL.

Corrigir um bug na compressão RLE do TGA e do PCX para imagens de 512x512.

Estão em nossos planos:

- Os formatos: Windows Icons (.ICO), MacPaint, RLE, PNG, PPM e RAW (este último usa um arquivo adicional para se saber o tipo da imagem e onde ela está guardada no arquivo).
- Leitura e escrita parcial de imagens.
- Leitura e escrita de múltiplas imagens em um mesmo arquivo, para os formatos que suportam essa funcionalidade.
- Leitura de uma versão reduzida da imagem no arquivo (thumbnail).
- Outras callbacks específicas de cada formato. (SGI - image name, TIF - image description, GIF - background color, GeoTIFF - vários)
- Implementar os algoritmos de compressão/descompressão dos formatos RAS e SGI que

## Comunicação de Problemas

Ao detectar algum erro, envie mensagem para Antonio E. Scuri ([scuri@tecgraf.puc-rio.br](mailto:scuri@tecgraf.puc-rio.br)) com informações sobre o erro explicitando o(s) formato(s) e a(s) plataforma(s) em que ocorreu.

## Agradecimentos

Este trabalho foi desenvolvido no TeCGraf, através do convênio PETROBRAS/CENPES.

O acesso ao formato TIFF é implementado usando-se a biblioteca LibTIFF, versão 3.4 beta 37, desenvolvida por Sam Leffer da Silicon Graphics, ver em <http://www-mipl.jpl.nasa.gov/~ndr/tiff/> ou em <ftp://sgi.com/graphics/tiff>.

O acesso ao formato JPEG é implementado usando-se a biblioteca LibJPEG, versão 6b, desenvolvida pelo Independent JPEG Group, ver em <ftp://ftp.uu.net/graphics/jpeg>.

The Graphics Interchange Format is the Copyright property of CompuServe Incorporated. GIF is a Service Mark property of CompuServe Incorporated.

A Antonio Nabuco Tartarini pelo desenvolvimento dos formatos JPEG, TGA e XBM. A Erick de Moura Ferreira pela versão preliminar do driver PCL.

## Equipe de Desenvolvimento

- Antonio E. Scuri ([scuri@tecgraf.puc-rio.br](mailto:scuri@tecgraf.puc-rio.br)) - Arquitetura e Implementação dos formatos BMP, PCX, TIFF, LED, GIF, SGI e RAS.

## Sobre o IM Online

Esta página está disponível em <http://www.tecgraf.puc-rio.br/manuais/im>. Estas páginas foram criadas utilizando HTML 3.2 e JavaScript 1.1. Elas são melhor vistas e percorridas com Internet Explorer 4 (ou posterior) ou Netscape 4 (ou posterior). Embora sejam vistas no Netscape 3, não são vistas no Internet Explorer 3.

Usando o Microsoft Word 97 geramos uma versão impressa deste manual, que também está disponível em formato Adobe Acrobat ([IM.PDF](#) 1Mb).

Este manual foi criado usando o toolkit de criação de manuais ManJS, que pode ser encontrado em <http://www.tecgraf.puc-rio.br/~scuri/manjs>.

# Guia

## Acesso às Imagens

As imagens RGB são tratadas como componentes independentes, utilizando para tanto um ponteiro para cada componente. No caso de imagens Indexadas existe apenas um ponteiro para os índices e um ponteiro para a palette.

O tipo dos ponteiros para as componentes devem ser `UNSIGNED CHAR`. O ponteiro para a palette é `UNSIGNED LONG`. O tamanho em bytes dos ponteiros para as componentes é sempre Linhas x Colunas. A localização de cada pixel é feita deslocando-se o ponteiro de um *offset* calculado pela fórmula:

`pixel(x,y) = Buffer[y * Width + x]`

assim, nota-se que a primeira linha da imagem fica no fim do ponteiro. As cores na palette são armazenadas como um valor de 32 bits compacto, são fornecidas duas funções para codificação e decodificação deste valor nas componentes RGB.

## Exemplos

Foram implementados 4 exemplos que demonstram a funcionalidade da biblioteca. O programa [iminfo](#) mostra informações sobre a imagem contida em um determinado arquivo. O programa [imconvert](#) converte imagens de um formato em outro. O programa [imquantize](#) converte uma imagem 24 bpp em uma imagem de 8 bpp tentando manter a qualidade o melhor possível. O programa [imview](#) visualiza uma imagem em uma janela usando as bibliotecas IUP e CD. Todos os 4 exemplos são portáteis e foram testados em todos os ambientes, exceto em Windows 3.xx + Win32s onde **imconvert**, **imquantize** e **iminfo** não funcionam pois não existe um linha de comando 32 bits nesse ambiente.

## Variáveis e Tipos

Para manipular as imagens é necessário a declaração de variáveis tais como:

```
int Width; // Largura da imagem.
int Height; // Altura da imagem

int Type; // Tipo de imagem no arquivo (0 = RGB, 1 = MAP).

unsigned char *Red, *Green, *Blue; // A imagem RGB.

unsigned char *Map; // A imagem Map.
long * Colors; // A tabela de cores para a imagem Map.
int PalSize; // Numero de cores presentes na tabela.
```

## Inicialização

A função **imImageInfo** retorna informação sobre a imagem contida no arquivo. Com esta informação pode-se reservar memória para a imagem. Que sempre é feito da maneira a seguir:

```
Map ou Red ou Green ou Blue = (unsigned char *)malloc(Width *
Height);
```

e para a paleta de cores:

```
Colors = (long *)malloc(PalSize * sizeof(long));
```

## Leitura e Escrita

A imagem pode ser lida usando-se as funções **imLoadRGB** ou **imLoadMap**. A função **imLoadMap** só pode ser executada para arquivos que contenham uma imagem com 8 bits por pixel ou menos. A função **imLoadRGB** pode ser usada em qualquer situação, RGB ou MAP. Convém lembrar que certos formatos possuem tipos de imagens que não se enquadram nos dos tipos definidos, RGB e MAP.

A imagem pode ser salva usando-se as funções **imSaveMap** ou **imSaveRGB**.

Verifique no arquivo IM.H os códigos de erro retornados por essas funções.

A especificação do formato de arquivo e compressão são feitas através de uma única variável. Para tanto a informação sobre o formato vem no primeiro byte (0x00FF) e a informação sobre a compressão vem no segundo byte (0xFF00).

## Cores

As cores na paleta são codificadas, para poderem ser manipuladas deve-se usar as funções **imEncodeColor** e **imDecodeColor** para transformar valores RGB em uma cor no formato da IM. A codificação de cor usada é a mesma usada na biblioteca CD.

Quando a tela possui apenas 256 cores ou menos e a imagem é RGB, pode-se utilizar a função **imRGB2Map**, que converte de maneira ótima, uma imagem RGB em uma imagem de 256 cores ou menos.

## Uso com a Biblioteca CD - Canvas Draw

A biblioteca foi preparada para ter um ajuste perfeito com o CD.

A função **cdGetImageRGB** captura uma imagem do canvas ativo. As funções **cdPutImageRGB** e **cdPutImageMap** coloca uma imagem RGB ou indexada no canvas ativo respectivamente, estas funções permitem reduzir ou ampliar a imagem ao colocá-la no canvas.

Em dispositivos com 256 cores é recomendado o uso da função **cdPalette** antes de desenhar a imagem para melhorar sua qualidade.

## Implementando um Novo Formato

A implementação de um novo formato requer um esforço menor do que o normal. A IM supõe que todo formato de image possui a estrutura **Header+ColorMap+Data** ou **Header+Data+ColorMap**. Para imagens RGB o ColorMap não existe. O Header contém todos os dados sobre a imagem e o arquivo. E Data contém a imagem organizada por linhas, comprimida ou não.

Usando apenas algumas funções para cada formato, todas as funções da IM são implementadas. As funções necessárias de cada formato são Open/ColorMap/Line/LineRGB para Read e Write mais a função Close, totalizando 9 normalmente pequenas funções.

São fornecidos dois arquivos: [imxxx.c](#) e [imxxx.h](#) para serem usados como modelo para a implementação de um novo formato. Esses arquivos acessam o arquivo de definição de um arquivo de imagem **imFile** e de um formato de arquivo de imagem **imFormat**. Veja no arquivo IMIO.H para maiores detalhes.

A aplicação que for usar esse formato deve incluir o arquivo IMXXX.H, e antes de chamar qualquer função da IM, chamar a fun **imAddFormatXXX**, que garantirá a inclusão do formato na lista interna de formatos da IM. Para usar os formatos da distribuição da IM essa inicialização não é necessária, pois é feita pela própria IM. Pode-se então usar a macro IM\_XXX para salvar o arquivo no formato desejado ou saber se um determinado arquivo está naquele formato.

## Acessando Arquivos Virtuais

IntelxMotorola e permitem sua configuração para acessar um área de memória em vez de um arquivo, basta usar a função **binFileSetMemoryFunctionTable** onde o nome do arquivo será na realidade o buffer de leitura/escrita. Para voltar a acessar arquivos normalmente deve-se usar a função **binFileSetStreamFunctionTable**.

Pode-se ainda criar um acesso completamente personalizado criando-se um driver de acesso com funções do tipo Open/Close/Read/Write/SeekTo/etc... Permitindo por exemplo que uma imagem seja lida diretamente de um campo longo em um banco de dados.

Para maiores detalhes veja o arquivo IMIO.H.



---

## Funções

---

### Cor

**long imEncodeColor(unsigned char red, unsigned char green, unsigned char blue);**

Converte uma cor no formato RGB para o formato interno da IM. (É idêntica a função equivalente da biblioteca CD.)

**void imDecodeColor(unsigned char\* red, unsigned char\* green, unsigned char\* blue, long color);**

Converte uma cor no formato interno da IM para o formato RGB. (É idêntica a função equivalente da biblioteca CD.)

---

### Informação

**int imFileFormat(char \*filename, int\* format);**

Informa o formato de um determinado arquivo. Retorna `IM_ERR_NONE`, `IM_ERR_OPEN` ou `IM_ERR_READ`. `format` recebe uma combinação do identificador do formato com o identificador de compressão. Para extrair o identificador do formato faça (`format & 0x00FF`), para extrair o identificador de compressão faça (`format & 0xFF00`).

**int imImageInfo(char \*filename, int \*width, int \*height, int \*type, int \*pal\_size);**

Retorna informações sobre a imagem em determinado arquivo. Retorna `IM_ERR_NONE`, `IM_ERR_OPEN`, `IM_ERR_READ`, `IM_ERR_FORMAT`, `IM_ERR_TYPE` ou `IM_ERR_COMP`.

---

### I/O

**int imLoadRGB(char \*filename, unsigned char \*red, unsigned char \*green, unsigned char \*blue);**

Lê uma imagem RGB de um determinado arquivo. É necessário **imImageInfo** para poder reservar memória para os ponteiros **red**, **green** e **blue** antes de ler a imagem. Retorna `IM_ERR_NONE`, `IM_ERR_OPEN`, `IM_ERR_READ`, `IM_ERR_FORMAT`, `IM_ERR_TYPE` ou `IM_ERR_COMP`.

**int imSaveRGB(int width, int height, int format, unsigned char \*red, unsigned char \*green, unsigned char \*blue, char \*filename);**

Salva uma imagem RGB em determinado arquivo e formato. Retorna `IM_ERR_NONE`, `IM_ERR_OPEN`, `IM_ERR_WRITE`, `IM_ERR_FORMAT`, `IM_ERR_TYPE` ou `IM_ERR_COMP`. Para especificar o formato realize um "ou" binário entre o identificador do formato e o identificador da compressão. Ex: (`IM_BMP | IM_COMPRESSED`)

**int imLoadMap(char \*filename, unsigned char \*map, long \*colors);**

Lê uma imagem indexada de um determinado arquivo. É necessário usar a função **imImageInfo** para poder reservar memória para os ponteiros **map** e **color**. Retorna

IM\_ERR\_NONE, IM\_ERR\_OPEN, IM\_ERR\_READ, IM\_ERR\_FORMAT, IM\_ERR\_TYPE ou IM\_ERR\_COMP.

**int imSaveMap(int width, int height, int format, unsigned char \*map, int pal\_size, long \*colors, char \* filename);**

Salva uma imagem indexada em determinado arquivo e formato. Retorna IM\_ERR\_NONE, IM\_ERR\_OPEN, IM\_ERR\_WRITE, IM\_ERR\_FORMAT, IM\_ERR\_TYPE ou IM\_ERR\_COMP. Para especificar o formato realize um "ou" binário entre o identificador do formato e o identificador da compressão. Ex: (IM\_BMP | IM\_COMPRESSED)

---

## Conversão

**void imRGB2Map(int width, int height, unsigned char \*red, unsigned char \*green, unsigned char \*blue, unsigned char \*map, int pal\_size, long \*colors);**

Converte uma imagem RGB em uma imagem indexada. A imagem resultante é do mesmo tamanho da imagem original e sua tabela de cores pode conter até **pal\_size** cores, mas sempre menor que 256. É necessário reservar memória para os ponteiros **map** e **colors** antes de converter a imagem.

**void imMap2RGB(int width, int height, unsigned char \*map, int pal\_size, long \*colors, unsigned char \*red, unsigned char \*green, unsigned char \*blue);**

Converte uma imagem indexada em uma imagem RGB. A imagem resultante é do mesmo tamanho da imagem original. É necessário r ponteiros **red**, **green** e **blue** antes de converter a imagem.

**void imRGB2Gray(int width, int height, unsigned char \*red, unsigned char \*green, unsigned char \*blue, unsigned char \*map, long \*grays);**

Converte uma imagem RGB em uma imagem indexada com 256 tons de cinza. A imagem resultante é do mesmo tamanho da imagem original e sua tabela de cores contem 256 valores. É necessário reservar memória para os ponteiros **map** e **grays** antes de converter a imagem.

**void imMap2Gray(int width, int height, unsigned char \*color\_map, int pal\_size, long \*colors, unsigned char \*gray\_map, long \*grays);**

Converte uma imagem indexada em uma imagem indexada com 256 tons de cinza. A imagem resultante é do mesmo tamanho da imagem original e sua tabela de cores contem 256 valores. É necessário reservar memória para os ponteiros **gray\_map** e **grays** antes de converter a imagem.

---

## Tamanho

**void imResize(int src\_width, int src\_height, unsigned char \*src\_map, int dst\_width, int dst\_height, unsigned char \*dst\_map);**

Realiza uma mudança no tamanho da imagem usando interpolação bilinear. Para usar em imagens RGB chame a função uma vez para cada componente. É necessário reservar memória para o ponteiro **dst\_map** antes de chamar a função.



## Formatos

### **BMP (Windows 3.x ou OS/2 1.x Device Independent Bitmaps)**

O formato suporta imagens com 1, 4, 8, 16, 24 ou 32 bits per pixel. As imagens de até 8 bits por pixel podem estar comprimidas com algoritmo semelhante ao padrão RLE (exceto 1 bpp) e possuem uma palette RGB associada.

As imagens com mais de 8 bits per pixel não podem ser comprimidas e representam a cor RGB diretamente. Em 16 bits per pixel as combinações mais usuais para os bits -5-5 ou 5-6-5, para respectivamente R-G-B. Em 32 bits geralmente a cor é representada com 24 bits mais um byte extra para o canal alfa.

O formato é acessado na sua completude, incluindo imagens comprimidas. Mas imagens com 4 bpp comprimidas não são suportadas. O formato pode possuir informação de resolução definida em pixels por centimetro.

### **PCX (ZSoft PC-PaintBrush Picture)**

O formato suporta imagens com 1, 4, 8 ou 24 bits per pixel. Todos os bpp disponíveis podem ser comprimidos com algoritmo semelhante ao padrão RLE. Recomenda-se o uso do formato sempre comprimido para melhor compatibilidade com aplicações existentes, pois algumas delas julgam que o PCX sempre estará comprimido e tentam invariavelmente descomprimi-lo, mesmo que não o esteja gerando erros diversos.

O formato é acessado na sua completude. O formato pode possuir informação de resolução definida em pixels por polegada.

### **GIF (Compuserve Graphics Interchange Format)**

O formato suporta imagens com 1, 4 ou 8 bits per pixel. Não podem ser salvas imagens RGB comprimidas ou não, o formato não suporta.

Todas as imagens são sempre comprimidas com o algoritmo LZW. Assim, o formato suporta a escrita de imagens não comprimidas, é obrigatória a compressão.

O formato pode conter mais de uma imagem, apenas a primeira imagem é lida, as outras são ignoradas. O formato pode possuir informação de resolução definida em pixels por polegada.

### **TIF (TIFF, Tagged Image File Format)**

Devido a vasta possibilidades de imagens neste formato convém lembrar que a biblioteca se resumirá a ler imagens com 1, 4, 8, e 24 bpp, seguindo o Baseline TIFF. As imagens podem estar comprimidas nos padrões CITT3, Huffman, LZW e PackBits. A IM irá salvar imagens comprimidas usando o formato LZW. O formato pode possuir informação de resolução definida em pixels por centimetro ou em pixels por polegada.

Consideramos o TIFF como o formato mais completo e mais bem suportado pelas aplicações para armazenamento de imagens.

Devido à complexidade da compressão JPEG usamos uma biblioteca de domínio público desenvolvida pelas pessoas do próprio grupo JPEG, ver em ver em <ftp://ftp.uu.net/graphics/jpeg>.

### **LED (IUP/LED, Linguagem de Especificação de Diálogos)**

Esse não é propriamente um formato de imagem, mas conter imagens e por sua difusão no TeCGraf foi necessário inclui-lo na IM. O formato suporta apenas

O formato contém muitas outras informações além da imagem, por ser uma arquivo  
 -se por forçar o usuário a criar um arquivo separado para  
 que a IM possa le-lo. Neste novo arquivo remova todos os comentários e coloque a  
 declaração da imagem no início do mesmo e com as letras LED como as três  
 primeiras letras do arquivo.

Por exemplo:

```
LEDImage = IMAGE[
0= "0 0 0",
1= "192 192 192",
2= "0 0 128",
3= "255 255 255"]
(20,19
,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 etc...
```

A IM irá salvar um arquivo exatamente com essa características para ser  
 posteriormente incluído em outro arquivo LED.

## TGA (True Vision Targa)

O formato suporta imagens RGB, RGB indexadas por palette e tons de cinza. As  
 imagens pode ser comprimidas no padrão RLE.

O formato é acessado na sua completude, mas são acessadas apenas imagens com 8,  
 16, 24 e 32 bits per pixel.

---

## Quadro Comparativo

|            | 8 BPP                 |                 |                   |                 | 24 BPP                |                 |                   |              |
|------------|-----------------------|-----------------|-------------------|-----------------|-----------------------|-----------------|-------------------|--------------|
|            | <i>Not Compressed</i> |                 | <i>Compressed</i> |                 | <i>Not Compressed</i> |                 | <i>Compressed</i> |              |
|            | <b>Read</b>           | <b>Write</b>    | <b>Read</b>       | <b>Write</b>    | <b>Read</b>           | <b>Write</b>    | <b>Read</b>       | <b>Write</b> |
| <b>PCX</b> | Ok                    | Ok <sup>N</sup> | Ok                | Ok              | Ok                    | Ok <sup>N</sup> | Ok                | Ok           |
| <b>BMP</b> | Ok                    | Ok              | Ok                | Ok              | Ok                    | Ok              | ---               | ---          |
| <b>TIF</b> | Ok                    | Ok              | Ok                | Ok              | Ok                    | Ok              | Ok                | Ok           |
| <b>SGI</b> | Ok                    | Ok <sup>G</sup> | Ok                | ???             | Ok                    | Ok              | Ok                | ???          |
| <b>RAS</b> | Ok                    | Ok              | ***               | ???             | Ok                    | Ok              | ***               | ???          |
| <b>GIF</b> | ---                   | ---             | Ok                | Ok              | ---                   | ---             | ---               | ---          |
| <b>JPG</b> | ---                   | ---             | Ok                | Ok <sup>G</sup> | ---                   | ---             | Ok                | Ok           |
| <b>TGA</b> | Ok                    | Ok              | Ok                | Ok              | Ok                    | Ok              | Ok                | Ok           |
| <b>LED</b> | Ok                    | Ok              | ---               | ---             | ---                   | ---             | ---               | ---          |

**Legenda:**

**Ok** - Implementado e funcionando de acordo.

**Ok<sup>G</sup>** - Somente como gray scale.

**Ok<sup>N</sup>** - Não recomendado.

**---** - Não aplicável.

**???** - Não implementado pois não tem documentação.

**\*\*\*** - Implementado mas sem exemplos para testes.

**Comentários**

Convém lembrar que imagens de até 8 bits per pixel são convertidas para 8 bpp. E imagens com mais de 8 bpp são convertidas para 24 bpp. A representação de cor é sempre RGB. Imagens tons de cinza são retornadas com uma palette de tons de cinza. A função **imSaveMap** sempre salva a imagem com 8 bpp e a função **imSaveRGB** sempre salva a imagem com 24 bpp. Se o formato só suporta imagens tons de cinza para 8 bpp então a imagem é automaticamente convertida para tons de cinza antes de ser salva.

A função **imLoadRGB** fará com que imagens de até 8 bpp sejam convertidas para 24 bpp. A função **imLoadMap** retorna erro se a imagem a ser lida for com mais de 8 bpp.

Existem formatos que não suportam imagens com mais de 24 bpp fazendo com que a **imSaveRGB** retorne erro.

A opção de compressão nos diversos formatos não obedece padrão algum entre si, cada formato implementa a sua própria ou variação de alguma conhecida. Se for especificado que a imagem deve ser comprimida e o formato não suporta especificado que a imagem não deve ser comprimida e o formato **imSaveMap** e **imSaveRGB** retornarão erro.

A compressão RLE é mais adequada para imagens com muitas áreas uniformes. Quando a imagem possui muita informação distinta o arquivo pode ser maior que o não comprimido.

## Identificadores

### File Format

**IM\_BMP** - DIB, Windows Device Independent Bitmap  
**IM\_PCX** - ZSoft Picture  
**IM\_RAS** - Sun Raster File  
**IM\_GIF** - Compuserve Graphics Interchange Format  
**IM\_SGI** - Silicon Graphics RGB, RGBA or BW  
**IM\_TIF** - TIFF, Tagged Image File Format  
**IM\_JPG** - JPEG, Join Photographic Experts Group  
**IM\_LED** - IUP/LED, Linguagem de Especificação de Diálogos  
**IM\_TGA** - TrueVision Targa

### Format Compression

**IM\_NONE** - Sem compressão. Se for especificado e o formato tem que ser comprimido fará com que **IM\_ERR\_COMP** seja retornado.  
**IM\_DEFAULT** - Compressão default para a imagem data no formato especificado. Se o formato não suporta compressão a imagem não é então comprimida.  
**IM\_COMPRESSED** - Durante a leitura indica um formato de compressão diferente do adotado pela IM para salvar a imagem. Durante a escrita força a compressão da imagem, se o formato não suporta compressão fará com que **IM\_ERR\_COMP** seja retornado.

### Image Types

**IM\_RGB** - 3 buffers separados (Red, Green and Blue)  
**IM\_MAP** - 1 buffer e uma tabela de cores

### Error Codes

**IM\_ERR\_NONE** - Nenhum erro.  
**IM\_ERR\_OPEN** - Erro durante a abertura do arquivo. O arquivo pode não existir ou você pode não ter direitos de acesso ao mesmo.  
**IM\_ERR\_READ** - Erro durante a leitura do arquivo.  
**IM\_ERR\_WRITE** - Erro durante a escrita do arquivo. Pode ter auido falta de espaço em disco.  
**IM\_ERR\_FORMAT** - Formato inválido. Formato de arquivo não conhecido.  
**IM\_ERR\_TYPE** - Tipo de imagem inválido. Alguns formatos podem salvar apenas imagens de um determinado tipo, ou a imagem é RGB e você tentou carregá-la como MAP.  
**IM\_ERR\_COMP** - Compressão não suportada.