



Graphics Library Application Programming Interface

Table of Contents

- qq_AnnCloseFile - close annotation file with save option.....
- qq_AnnGetBasePage - Return name of file for base page image.....
- qq_AnnGetHandle - return Annotation reference from IDX.....
- qq_AnnMergeFile - merge annotations into a FAX file.....
- qq_AnnOpenFile - Open a file as an annotation.....
- qq_AnnOpenSharedFile - open a copy of an annotation file.....
- qq_AnnSaveImage - save the current annotation tags.....
- qq_CloseFile - close file, and clean up memory.....
- qq_ConvertTo - create output file.....
- qq_Crop - write out a clipped FAX file.....
- qq_DisableStatusDlg - disable status reporting for current Idx.....
- qq_FastLoadBitmap - return an unscaled bitmap image (1 bit only).....
- qq_FaxDumpBitmap - dump the contents of a bitmap one line at a time.....
- qq_FaxOpenFile - open the output file.....
- qq_FaxSetFileResolution - set the resolution of the output file.....
- qq_FolderAddFile - add a file to the end of a folder.....
- qq_FolderGetFile - get the n'th name in the folder.....
- qq_GetBitsPerPel - return number of bits in a pel (1, 8).....
- qq_GetCompressionOptions - Retrieve last compression option value.....
- qq_GetFileType - Return file type for an image handle.....
- qq_GetHzDPI - return horizontal Dots Per Inch.....
- qq_GetImgHeight - get scaled height of image.....
- qq_GetImgWidth - get scaled width of image.....
- qq_GetLastError - Return code number for last annotation error.....
- qq_GetMaxScaling - gets the maximum allowable bitmap size.....
- qq_GetNumberOfPages - Return number of pages in a file.....
- qq_GetPageResolution - get default fax resolution and page size.....
- qq_GetPrintDevice - return name of Active Driver (or empty string).....
- qq_GetRotation - get current rotation.....
- qq_GetScale - get current scale.....
- qq_GetStretch - get current stretch value.....
- qq_GetTifBitOrder - get default TIF bit and byte order.....
- qq_GetUsageCount - Check if it is OK to quit application.....
- qq_GetVersion ab - Check if it is OK to quit application.....
- qq_GetVertDPI - return vertical Dots Per Inch.....
- qq_LoadBitmap - return a bitmap image.....
- qq_LoadPalette - return a palette value.....
- qq_OpenFile - Initialize instance variables, open file for read.....
- qq_PasteToFile - Paste clipboard contents into a file.....
- qq_SetBackground - Set Black on White or White on Black.....
- qq_SetDevMode - set device mode for printer.....
- qq_SetMaxScaling - sets the maximum allowable bitmap size.....
- qq_SetMirror - Set horizontal mirror option.....

qq_SetPage - Select page to view.....
 qq_SetPageResolution - set default fax resolution and page size
 qq_SetRotation - set current rotation.....
 qq_SetScalingOptions age to view
 qq_SetTifBitOrder - set default bit and byte ordering
 qq_UpdateBitmap - update the current bitmap.....
 qq_ViewAnnDeleteAnnotation - Delete a specified annotation
 qq_ViewAnnDeleteSelected - Delete selected annotations.....
 qq_ViewAnnGetArowObj - Get the xxx specific properties.....
 qq_ViewAnnGetBmpObj - Get the xxx specific properties.....
 qq_ViewAnnGetCircObj - Get the xxx specific properties.....
 qq_ViewAnnGetCurrentProcess - Get the xxx specific properties.....
 qq_ViewAnnGetFirstBase - Get the xxx specific properties
 qq_ViewAnnGetFirstSelectedBase - Get the xxx specific properties
 qq_ViewAnnGetHiliteObj - Get the xxx specific properties.....
 qq_ViewAnnGetIndexBase - Get the xxx specific properties
 qq_ViewAnnGetLastBase - Get the xxx specific properties
 qq_ViewAnnGetLastSelectedBase - Get the xxx specific properties
 qq_ViewAnnGetNextBase - Get the xxx specific properties.....
 qq_ViewAnnGetNextSelectedBase - Get the xxx specific properties.....
 qq_ViewAnnGetPostObj - Get the xxx specific properties.....
 qq_ViewAnnGetPostStringProperty - Get the xxx specific properties.....
 qq_ViewAnnGetPrevBase - Get the xxx specific properties.....
 qq_ViewAnnGetPrevSelectedBase - Get the xxx specific properties
 qq_ViewAnnGetRectObj - Get the xxx specific properties.....
 qq_ViewAnnPaste - Paste clipboard contents into this annotation
 qq_ViewAnnSetArowObj - Get the xxx specific properties
 qq_ViewAnnSetBmpObj - Get the xxx specific properties.....
 qq_ViewAnnSetCircObj - Get the xxx specific properties.....
 qq_ViewAnnSetHiliteObj - Get the xxx specific properties.....
 qq_ViewAnnSetPostObj - Get the xxx specific properties
 qq_ViewAnnSetRectObj - Get the xxx specific properties.....
 qq_ViewAnnTranslateRect - Translate from real to image coordinates
 qq_ViewAnythingSelected - Determine if anything is selected
 qq_ViewClose - cleans up the 'View' object.....
 qq_ViewCopyToClipboard - copies current selection box to the clipboard
 qq_ViewDoBackground - handles background updates
 qq_ViewGetArowDefaultProperties - Get the xxx specific properties
 qq_ViewGetCreatedAnnProperties - Get the base properties
 qq_ViewGetHiliteDefaultProperties - Get the xxx specific properties.....
 qq_ViewGetIdx - returns the assigned idx value.....
 qq_ViewGetMode - gets current view mode
 qq_ViewGetPosition - returns the x/y cx,cy size of display window/scroll bars
 qq_ViewGetPostItDefaultProperties - Get the postIt specific properties.....
 qq_ViewGetPostItDefaultStringProperty - Get the xxx specific properties.....
 qq_ViewGetRectDefaultProperties - Get the xxx specific properties.....
 qq_ViewGetScrollPos - get the X/Y offset of the display bitmap
 qq_ViewGetWindow - returns the specified window handle (display/hScroll/VScroll)
 qq_ViewHandleEvent - handles the event message.....
 qq_ViewOpen - builds a 'View' object to support image display
 qq_ViewOpenEx - builds a 'View' object to support image display
 qq_ViewReload - force image to be reloaded (new idx values)
 qq_ViewRepaint - force image to be repainted
 qq_ViewSetArowDefaultProperties - Get the xxx specific properties
 qq_ViewSetBackground - sets background mode for Vdx.....
 qq_ViewSetCreatedAnnProperties - Set the base properties
 qq_ViewSetCursor - sets the cursor resource for the specified mode.....

qq_ViewSetHiliteDefaultProperties - Set the xxx specific properties
qq_ViewSetIdx - assigns an image to a view for display
qq_ViewSetMode - sets current view mode.....
qq_ViewSetPosition - sets size, Z-order, and position of display window.....
qq_ViewSetPostItDefaultProperties - Set the xxx specific properties
qq_ViewSetRectDefaultProperties - Set the xxx specific properties
qq_ViewSetScrollBars - sets scroll bar visibility flags.....
qq_ViewSetScrollPos - sets the X/Y offset of the display bitmap
qq_ViewSetVisibility - sets visibility flag for image

qq_AnnCloseFile - close annotation file with save option

```
/*$*****  
qq_AnnCloseFile () - close annotation file with save option
```

Parameters:

idx: Image handle
save: if TRUE, save annotations before closing the file

Returns:

0 on success
-1 if errors either in saving or in closing annotation.

Description:

If Save is TRUE, saves annotation file back into original file with no change in embedded / extracted objects. If saving the annotation file is not successful, exits with error code. If there is no file name associated with the annotation, it will use the standard file dialog to request a file name from the user.

Then, closes the image table entry associated with the handle passed in. Also, checks the remaining entries in the image table; if there are no other entries for this annotation, it closes down the annotation.

```
*****$*/  
INT _export FAR PASCAL  
qq_AnnCloseFile(  
    INT idx,          // image handle  
    INT save          // 1: save annotations before closing 0: don't save  
) {  
}
```

qq_AnnGetBasePage - Return name of file for base page image

```
/*$*****  
qq_AnnGetBasePage () - Return name of file for base page image
```

Parameters:

- idx : Image handle
- buf : Pointer to area where file name is to be stored
- len : Maximum length of buffer
- BasePageNum: Pointer to storage for page number in base page image file

Returns:

- 0 for success; non-0 on error
- on success, buf filled in to maximum length

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_AnnGetBasePage(  
    INT idx,  
    LPSTR buf,  
    INT len,  
    FPINT BasePageNum)
```

```
{  
}
```

qq_AnnGetHandle - return Annotation reference from IDX

```
/*$*****  
qq_AnnGetHandle () - return Annotation reference from IDX
```

Parameters:

idx : Image handle

Returns:

Annotation handle
-1 if error

```
*****$*/  
HANNOTATIONSET _export FAR PASCAL  
qq_AnnGetHandle(  
    INT idx)  
{  
}
```

qq_AnnMergeFile - merge annotations into a FAX file

```
/*$*****  
qq_AnnMergeFile () - merge annotations into a FAX file
```

Description:

Merges all annotations and base images into a single TIFF file.

The output will be a multi-page TIFF or other output file type, with the name passed in. If the name of the output file is invalid, the function will fail.

topCrop and botCrop are the amount of the image which you wish to have removed, e.g. to make space for the FAX source ID at the top, and a page number at the bottom. If these values are negative, the bitmap is enlarged on the top or bottom by the specified number of lines. The number of FAX image lines indicated by topWhite and botWhite are also whited out, to make space for this information. topWhite and botWhite cannot be negative.

Allowable file types are:

PS_TIF_GAMA: reverse bits Group 3
PS_TIF_NORMAL: normal bits Group 3
PS_PACKBITS: packed bits
PS_MMR_NORMAL: Group 4 image, bits reversed
PS_MMR_REVERSE: Group 4 image, bits reversed
PS_MR_NORMAL: Group 3 image - 2 dimensional
PS_MR_REVERSE: Group 3 image - 2 dimensional
PS_DCX: Multi-page PCX file (DCX file type)
PS_PCX: Single page PCX file
PS_EMPTY: No file is written

NOTE:

Merging requires that at least one annotation index be unused.

Returns:

0 on success
1 if user cancelled
-1 if errors either in saving or in closing annotation.
3 if unable to open or create output file

```
*****$ */
```

```
INT _export FAR PASCAL
```

```
qq_AnnMergeFile(  
    INT idx,          // Image handle  
    LPSTR szOutputFileName, // File name for merged output  
    INT fileType,    // Output file type  
    INT startPage,  // First page to write (0-based)  
    INT endPage,    // Last page to write  
    INT lowRes,     // if TRUE, low-resolution output  
    INT breakPages, // if TRUE, one file per page  
    INT topCrop,    // number of lines off top  
    INT topWhite,   // number of top lines to white out  
    INT botWhite,   // number of bottom lines to white out  
    INT botCrop     // number of lines to eat off bottom  
)  
{  
}
```


qq_AnnOpenFile - Open a file as an annotation

```
/*$*****  
qq_AnnOpenFile () - Open a file as an annotation
```

Parameters:

hWnd: Handle of parent window

fileName: Pointer to the name of the file to load the annotations from

OpenMode: Mode in which to attempt to open the file:

0: We expect to write; open in read/write and fail if already open

1: We do not expect to write to the source file; make a copy if the file is already locked

NoPrompt: Whether we should ask to open a similarly named annotation file // KWD Nov 23

True (non-0) if we want to simply open the passed file name

False (0) to ask the user if he wants to open the annotation file instead

Returns:

Image handle if annotations loaded successfully

-1 if errors

Call qq_GetLastError for most recent error code

Description:

Passed a file name, loads the file as an annotation object; if the file is an annotation file, it loads normally, else the file is loaded as the base page image for the annotations, and no actual annotations are attached.

Information as necessary to uniquely identify the annotation inside the file is retained invisibly to the caller. The annotations loaded from the file, if any, are inserted into an annotation object which we create.

Parent window is required in order to set up the annotation; it is the window in which the annotation will be drawn. It can be the main window of the application, or another window; it can also be changed after initialization.

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_AnnOpenFile(  
  HWND hWnd,      // handle of parent window (NULL otherwise)
```

```
  LPSTR fileName, // name of annotation file or FAX file
```

```
  INT OpenMode,   // 0: read/write 1: read only
```

```
  INT NoPrompt)  // 1: open file as-is 0: prompt for annotation file
```

```
{  
}
```

qq_AnnOpenSharedFile - open a copy of an annotation file

```
/*$*****  
qq_AnnOpenSharedFile () - open a copy of an annotation file
```

Parameters:

hWnd: Handle of parent window
idx: Image handle returned by the original qq_AnnOpenFile call
page: Page number to load base page image of

Returns:

Image handle if annotations loaded successfully
-1 if errors
Call qq_GetLastError for most recent error code

Description:

Given an already-opened annotation, this routine sets up a second image handle, conceptually for either a thumbnail view or a second page, of the same annotation.

Parent window is required in order to set up the annotation; it is the window in which the annotation will be drawn. It can be the main window of the application, or another window; it can also be changed after initialization.

Note: This does require an additional image handle, of which there are a limited supply.

```
*****$*/
```

```
INT _export FAR PASCAL  
qq_AnnOpenSharedFile(  
    HWND hWnd,  
    INT idx,  
    INT page)  
{  
}
```

qq_AnnSaveImage - save the current annotation tags

```
/*$*****  
qq_AnnSaveImage () - save the current annotation tags
```

Parameters:

- idx: Image handle for the image to be saved
- fileName: File name which the file is to be saved under.
Note: If annotation was opened with OpenMode == 1, this must be a different file name than was used then.
- Extract: Controls whether embedded images are extracted:
 - 0: neither extract nor embed any images
 - 0x0001: extract all embedded objects to individual files
 - 0x0002: embed all objects from external files
 - 0x0010: extract embedded base page images to individual files
 - 0x0020: embed all base page images from external files

Returns:

- 0 on success
- 1 on failure

Description:

Passed an image handle and a file name, this routine attempts to save the annotations into the file. Embedded objects in the file may be moved up in the file so that they occupy minimum space, and all annotation tags are added behind that.

```
*****$*/  
INT _export FAR PASCAL  
qq_AnnSaveImage(  
    INT idx,  
    LPSTR fileName,  
    INT Extract)  
{  
}
```

qq_CloseFile - close file, and clean up memory

```
/*$*****  
qq_CloseFile () - close file, and clean up memory
```

returns

-1 for error, 0 otherwise

Note:

If the file is an annotation file, this routine calls qq_AnnCloseFile()
to close the annotation as well.

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_CloseFile (
```

```
    INT idx      // Image handle to close
```

```
)  
{  
}
```

qq_ConvertTo - create output file

```
/*$*****  
qq_ConvertTo () - create output file
```

Description:

Use this routine to write the specified output file format from the currently opened file. For all file types other than 'FAX' or 'DCX', the default is to use the current scaling value, and output a separate file per page. For FAX and DCX images, each page is scaled up to appropriately fill the page (203x198 dpi), and pages are written out multiple pages per file.

To Create a Tiff image (MH, MR, or MRR) at a resolution other than standard fax dpi, set the proper page scaling, and use the 'TIF' file types, not the 'FAX' file types. Alternatively, use the FAX API functions (provided as a separate DLL) to output fax lines one at a time.

This function supports conversion FROM the following file types: DCX, TIFF (all flavours except for LZW), PCX, BMP, GIFF, and ASCII text.

Single Page file formats:

Multiple pages are numbered files (eg OUTPUT.T01, OUTPUT.T02).

Images are written out at the same scaling as the input file.

```
110 // PCX 1,2,4,8 bit, 4 bits is planes  
* 111 // PCX 1,2,4,8 bit, 4 bit is packed  
* 140 // GIFF 1,2,4,8 bit  
* 150 // BMP 1,2,4,8 bit  
151 // BMP force to 4 bit  
152 // BMP force to 8 bit  
  
129 // TIFF 1bit Black and White uncompressed  
229 // TIFF 1bit Black and White uncompressed  
130 // TIFF 1bit Black and White Pack Bits  
230 // TIFF 1bit Black and White Pack Bits Low Resolution  
131 // TIFF shades of grey  
231 // TIFF shades of grey Low Resolution  
* 132 // TIFF 1,2,4,8 bit  
232 // TIFF 1,2,4,8 bit Low Resolution  
  
* 133 // TIFF MH group 3  
233 // TIFF MH group 3 Low Resolution  
137 // TIFF MR group 3 2D  
237 // TIFF MR group 3 2D Low Resolution  
135 // TIFF MMR group 4  
235 // TIFF MMR group 4 low resolution
```

FAX Multi-Page file formats:

Files contain multiple pages per file

Images are scaled to fit a standard fax page (1728 bits wide, 203*196 dpi)

--> To change default page size and resolution, use qq_SetPageResolution ()

--> To change byte or bit ordering, use qq_SetTifBitOrder ()

--> To force separate pages, add 1024 (0x0400, VI_FLAG_SINGLE_PAGES)

```
* 120 // DCX intel format  
* 134 // FAX MH group 3
```

```

234 // FAX MH group 3 low resolution
* 136 // FAX MMR group 4
236 // FAX MMR group 4 low resolution
* 138 // FAX MR group 3 2D
238 // FAX MR group 3 2d low resolution

```

TIF Assembly: Append TIF pages to output file 'AS IS'

- copy contents of TIF file over to output file
- If output file already exists, then append, otherwise, create it.
- update page pointers (each header points to the next page in the file)
- update 'page x of y' value
- keep current resolution information
 - * 400 // Append TIF files

Notes:

- * - recommended values for each supported file type.
- MH - Modified Huffman (standard Group 3 fax compression)
- MR - Modified Read (2 dimensional Group 3 fax encoding)
- MMR- Modified Modified Read (2 dimensional Group 4 fax encoding)

To suppress Conversion dialog box

When doing the conversion, a status dialog box pops up. To hide this dialog box, call qq_DisableStatusDlg() before calling qq_ConvertTo().

This routine WILL NOT write out an annotation file; passed the handle of an annotation file, it will write out only the base image. To write out an annotation file, use the routine qq_AnnSaveImage or qq_AnnCloseFile. To write a merged bitmap, with all annotations merged into the output graphics file, use the routine qq_AnnMergeFile.

Example:

```

// to create a TIFF MH Group 3 file from filename.tif
// Output file name is outfile.tif
INT idx; // file handle
INT pages;
INT scale = 100; // current scaling factor
INT stretch = 100; // stretch height by 'x' percent
INT options = 05; // FAX_ENHANCE_BLACK (white, normal, grey, black)
INT fileType = 134; // FAX MH Group 3
INT quiet = FALSE; // if TRUE, then don't display status dialog

idx = qq_OpenFile ("filename.tif"); // returns -1 for error
pages = qq_GetNumberOfPages (idx);
qq_SetScalingOptions (idx, scale, stretch);
qq_SetCompressionOptions (idx, options);
if (quiet)
    qq_DisableStatusDlg (idx, TRUE); // don't display status

// optionally set default (affects how FAX files are generated)
qq_SetPageResolution (idx, FAX_DEFAULT_WIDTH, FAX_DEFAULT_HEIGHT,
    FAX_X_RESOLUTION, FAX_FINE); // (optional) set custom FAX defaults
qq_SetTifBitOrder (idx, FAX_REVERSE_BITS, FAX_INTEL); // optional

qq_ConvertTo (idx, "outfile.tif", fileType, 0, pages-1);
qq_CloseFile (idx);

```

Returns:

0 for Success, -1 otherwise

*****\$*/

INT _export FAR PASCAL

qq_ConvertTo (

INT idx, // open image handle (scaling set)

LPSTR outName, // name of output file

INT outType, // output file type (if high bit set - break pages)

INT startPage, // starting page (0 based)

INT endPage // ending page (inclusive)

)
{
}

qq_Crop - write out a clipped FAX file

```
/*$*****  
qq_Crop () - write out a clipped FAX file
```

Description:

Create a specified FAX output file (multiple pages / single pages).
Requires that IMGFAPI.DLL be present (uses the IMGFAPI.DLL)

This utility allows the end user to crop fax images, removing the Caller Id, and other header or footer information.

supported clipping values:

topClip - if positive, eat top lines, if negative, add top lines
topWhite - convert top lines to 'White'
botWhite - convert bottom lines to White
botClip - clip remaining bottom lines

Supported File Formats:

PS_TIF_GAMA	0	// reverse bit order
PS_TIF_NORMAL	1	// normal bit order
PS_DCX	2	
PS_PCX	3	
PS_MR_REVERSE	4	// Group 3 2D Modified Read (reverse)
PS_MR_NORMAL	5	// Group 3 2D Modified Read (normal)
PS_MMR_REVERSE	6	// Group 4 Modified Mofified Read (reverse)
PS_MMR_NORMAL	7	// Group 4 Modified Modified Read (normal)
PS_PACKBITS	8	// Normal Tif Packbits

Algorithm:

```
Set input to B&W, 100% scaling  
Open output page file  
For each page in image:  
  Load bitmap  
  dump bitmap (specify crop lines, and white space)  
  Release bitmap  
  if more pages {  
    if breakPages  
      Close output file  
      Open new output file  
    else  
      Append page  
  }  
}  
Close output file
```

Returns:

0 for Success, -1 otherwise

```
*****$*/  
INT _export FAR PASCAL  
qq_Crop (  
  LPSTR inputFileName, // Name of input file  
  LPSTR outFileName, // Name of output file  
  INT fileType, // output file type PS_DCX, PS_GAMA, etc
```

```
INT startPage, // startPage 0 based offset of start page
INT endPage, // 0 based, inclusive. Can be any large number
INT lowRes, // 0:false 1:true. Force low res output
INT breakPages, // 0:false 1:true. Force one page per file, files
// are numbered
INT topCrop, // Number of lines to crop. If negative, add lines
// to top
INT topWhite, // Number of lines to overwrite with White
INT botWhite, // Number of ending lines to overwrite with white
INT botCrop // Number of lines to crop at end. If negative, pad
) // end.
{
}
```

qq_DisableStatusDlg - disable status reporting for current Idx

```
/*$*****  
qq_DisableStatusDlg () - disable status reporting for current Idx
```

Parameters:

idx : Image handle
value : TRUE to disable status dialog, FALSE to enable

Description:

Disables status dialogs for current IDX handle. Useful if the conversion process is being done in background.

Although the status dialog is disabled, the conversion process still checks for (and processes) messages in a properly behaved fashion.

Note: You must call this function only after opening the image.

Returns:

previous status value (or -1 for failure)

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_DisableStatusDlg (
```

```
    INT idx,          // active image handle
```

```
    INT value        // TRUE for disable, FALSE to re-enable
```

```
)
```

```
{
```

```
}
```

qq_FastLoadBitmap - return an unscaled bitmap image (1 bit only)

```
/*$*****  
qq_FastLoadBitmap () - return an unscaled bitmap image (1 bit only)
```

Parameters:

idx: Image handle for file

Description:

This routine was written to optimize loading an unscaled bitmap image. The currently defined page is loaded at up to twice the speed as a normal image.

This routine does not handle any file type other than TIFF MH, MR, and MMR. If the file type is not handled, qq_LoadBitmap is called (and the time to load doubles).

You are responsible for deleting the bitmap after you are finished with it. To delete a bitmap, call the windows API:

```
DeleteObject (hBitmap)
```

Memory requirements:

- a raw bit buffer the size of the page (500,000 bytes)
- an offscreen bitmap (a product of CreateBitmap())

Example:

```
nIdx = qq_OpenFile (lpzFilename);  
hBitmap = qq_FastLoadBitmap (nIdx, 0); // loads at 100% scaling  
qq_CloseFile(nIdx);
```

... use bitmap ...

```
DeleteObject (hBitmap);
```

returns

0 for error, handle to bitmap otherwise

```
*/
```

```
HBITMAP _export FAR PASCAL
```

```
qq_FastLoadBitmap (
```

```
INT idx // currently active file
```

```
)
```

```
{
```

```
HBITMAP hBitmap = (HBITMAP) 0;
```

```
VI_VIEW_INFO viewInfo;
```

```
if (idx < 0 || !img_buf || !img_buf[idx].used)  
return (0);
```

```
if (qq_GetBitsPerPel(idx) >= 2)  
return (0);
```

```
qqIncUsage (); // increment usage count
```

```
// if MaxBitmapSize set, returns less than 100%
```

```

qq_SetScalingOptions (idx, 100, 100);
qqSetMapInfo (idx);

ViGetViewInfo (img_buf[idx].imgRef, &viewInfo);
hBitmap = qqFastLoad (idx, &viewInfo); // load an unscaled bitmap

// if fast load fails, then at least do as best we can
if (!hBitmap)
    hBitmap = qq_LoadBitmap (idx, 0);

qqDecUsage (); // decrement usage count
return (hBitmap);
}

```

```

/*$
qq_BuildBitmap - build a pre-emptive display block

```

Parameters:

idx : Image handle
hPal: Palette to be attached to bitmap

Description:

In order to do quick re-calcs (for rotation, or scaling) we always keep the last image cached in memory. As an image may take up a lot of memory, we do not want to cache all images. If an old cache is active, then we want to disable it first before setting up the new cache.

To support Pre-emptive display logic, implement the following:

Calling app opens the image
Calling app requests a Display bitmap (current scaling)
- the bitmap is created blank (no contents)
- calling app displays blank bitmap

Calling app idle loop:
- requests the next band to be added to the bitmap
(added left to right, or top to bottom depending on rotation)
- does an 'invalidate rect' for the updated region of the bitmap
(causing a 'WM_PAINT' message to be generated)
- calling app processes WM_PAINT message

VB code:

support an internal queue to keep track of opened hctl's
WM_NCCREATE adds to queue
WM_NCDESTROY removes from queue
if timer tick available call qq_BuildBitmap () instead of
qq_LoadBitmap()
implement a timer tick to call qq_UpdateBitmap for each hctl in
list (do one at a time, in order from most recent to oldest)

Returns:

Updated Build_Bitmap structure
0 for error, handle to bitmap otherwise

```

*****$*/
HBITMAP _export FAR PASCAL
qq_BuildBitmap (

```

```
INT idx,          // currently active file
HPALETTE hPal    // Logical palette (to be realized to bitmap)
)
{
}
```

qq_FaxDumpBitmap - dump the contents of a bitmap one line at a time

```
/*$*****  
qq_FaxDumpBitmap () - dump the contents of a bitmap one line at a time
```

Dumps the passed bitmap to the file (opened by ps_OpenWrite) in the specified format.

Returns:

0 for success, -1 otherwise

NOTES regarding the status window:

Periodically, this routine will send a message to the window whose handle is passed as hWnd. The message has the values
message = MY_WMPRINTPCT (#defined as WM_USER + 1000 in imgconvt.h)
wParam = page number passed into this routine
lParam = % completion of this page
If the passed parameter is NULL (0), no messages are sent.

In the current incarnation of this library, it is not possible to set a control in this window which will terminate the merge process.

```
*****$*/
```

INT

```
_export FAR PASCAL
```

```
qq_FaxDumpBitmap (
```

```
    INT fh,           // output file handle (FAXAPI)  
    HBITMAP hBitmap, // handle to bitmap  
    INT curPage,     // current page # (required for status dialog)  
    INT lowRes,      // if True, chop out every second line in file  
    INT topCrop,     // number of lines off top  
    INT topWhite,    // number of top lines to white out  
    INT botWhite,    // number of bottom lines to white out  
    INT botCrop,     // number of lines to eat off bottom  
    HWND hWnd       // status dialog window to send % messages to
```

```
)  
{  
}
```

qq_FaxOpenFile - open the output file

```
/*$*****  
qq_FaxOpenFile () - open the output file
```

Returns:

0 for success, -1 otherwise

```
*****$*/
```

INT

_export FAR PASCAL

qq_FaxOpenFile (

LPSTR outName, // name of output file

LPINT fh, // returned file handle

INT fileType, // PS_GAMA and other types

INT overwrite, // force overwrite if file exists

INT pad // round off page to multiple of x lines

)

{

}

qq_FaxSetFileResolution - set the resolution of the output file

```
/*$*****  
qq_FaxSetFileResolution() - set the resolution of the output file
```

MUST BE CALLED BEFORE qq_FaxOpenFile(). If this function is not called, qq_FaxOpenFile defaults to high resolution. Note that this setting is sticky; if you set low resolution, all files will be opened in low resolution until you explicitly set high resolution.

Returns:

0 for success, 1 otherwise

```
*****$*/  
INT  
_export FAR PASCAL  
qq_FaxSetFileResolution(  
    INT LowRes)          // if True, low resolution output  
{  
}
```

qq_FolderAddFile - add a file to the end of a folder.

```
/*$*****  
qq_FolderAddFile () - add a file to the end of a folder.
```

Description:

The folder name must end with a .MFX extension.

This function opens the current file, and appends the name to the end.
If folder does not exist, create it.

Also see:

qq_FolderGetFile ()

Returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_FolderAddFile (
```

```
LPSTR folderName, // name of folder
```

```
LPSTR fName, // name of file to add
```

```
INT flags, // mode flags: 0x1: force page 0x2: force lines
```

```
INT forcePage, // force the display to a certain page
```

```
INT forceLen // force page len to be limited to a max # of lines
```

```
)
```

```
{
```

```
}
```

qq_FolderGetFile - get the n'th name in the folder

```
/*$*****  
qq_FolderGetFile () - get the n'th name in the folder
```

Description:

Pick out the n'th file from the folder. The file name is returned in the specified buffer.

The folder name must end with a .MFX extension.

Also see:

qq_FolderAddFile ()

Returns:

The following error codes are returned:

- 1 general error (folder file does not exist, or not valid)
- 0 success
- 1 index is out of range
- 2 specified file at index position does not exist
- 3 specified file at index position is not recognized as a valid file

Name of file, and additional default information.

```
*****$*/  
INT _export FAR PASCAL  
qq_FolderGetFile (  
    LPSTR folderName, // name of folder  
    INT index, // n'th file to retrieve. 0 based.  
    LPSTR fName, // returned buffer for retrieved file name  
    FPINT flags, // returned flags  
    FPINT forcePage, // if (flags & 0x1), force page QQ_FOLDER_FORCE_PAGE  
    FPINT forceLines // if (flags & 0x2), force lines QQ_FOLDER_FORCE_LINES  
)  
{  
}
```

qq_GetBitsPerPel - return number of bits in a pel (1, 8)

```
/*$*****  
qq_GetBitsPerPel () - return number of bits in a pel (1, 8)
```

Parameters

idx : Image handle

returns

number of bits per pel (for current map setting), 1 or 8
-1 if idx is invalid

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetBitsPerPel (
```

```
INT idx
```

```
)  
{  
}
```

qq_GetCompressionOptions - Retrieve last compression option value

```
/*$*****  
qq_GetCompressionOptions () - Retrieve last compression option value
```

Parameters:

idx: Image handle

Returns:

Compression option last set by SetCompressionOptions.

If a value other than the following pre-defined options was set, then that value will be returned.

```
FAX_ENHANCE_WHITE: 0  
FAX_NORMAL        1  
FAX_ENHANCE_HZ:   0x2  
FAX_ENHANCE_VT:   0x3  
FAX_GRAY_SCALE:   0x4 // switch to gray scale  
FAX_ENHANCE_BLACK: 0x5
```

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetCompressionOptions (
```

```
    INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetFileType - Return file type for an image handle

```
/*$*****  
qq_GetFileType () - Return file type for an image handle
```

Parameters:

idx: Image handle returned from qq_OpenFile or qq_AnnOpenFile

Returns:

UNSUPPORTED	0	
PCX	110	PCX 1,2,4,8 bit, 4 bits is planes
PCX_P	111	PCX 1,2,4,8 bit, 4 bit is packed
DCX	120	DCX 1bit B&W only
TIFUBW	129	TIFF uncompressed B&W
TIFBW	130	TIFF 1bit B&W only
TIFCL	132	TIFF 1,2,4,8 bit
TIF3	133	TIFF 1bit B&W only FAX group 3
TIF4	135	TIFF 1bit B&W only FAX group 4
TIF2D	137	TIFF 1bit B&W only FAX MR group 3 2D
GIF	140	GIFF 1,2,4,8 bit
BMP	150	BMP 1,2,4,8 bit
FXR	190	Winfax 3.0/2.0 high res format
ANO	300	Imager annotation file

-1 for error

Description:

Returns file type for open files only.

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetFileType (
```

```
    INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetHzDPI - return horizontal Dots Per Inch

```
/*$*****  
qq_GetHzDPI () - return horizontal Dots Per Inch
```

Parameters:

idx : Image handle

returns

-1 for error, 0 if unavailable, otherwise Dots per Inch.

typical values:

Fax Resolution: 204 DPI

Scanner 300 DPI

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetHzDPI (
```

```
INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetImgHeight - get scaled height of image

```
/*$*****  
qq_GetImgHeight () - get scaled height of image
```

Parameters:

idx : Image handle

returns

number of pixels high, -1 for error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetImgHeight (
```

```
INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetImgWidth - get scaled width of image

```
/*$*****  
qq_GetImgWidth () - get scaled width of image
```

Parameters:

idx : Image handle

returns

number of pixels across, -1 for error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetImgWidth (
```

```
INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetLastError - Return code number for last annotation error

```
/*$*****  
qq_GetLastError () - Return code number for last annotation error
```

Parameters: None

Returns:
Last error code that occurred in the program

Side effects:
Zeroes out the error code in local storage

```
*****$*/  
INT _export FAR PASCAL  
qq_GetLastError(  
void)  
{  
}
```

qq_GetMaxScaling - gets the maximum allowable bitmap size

```
/*$*****  
qq_GetMaxScaling () - gets the maximum allowable bitmap size
```

Description:

See qq_SetMaxScaling

Returns:

-1 for failure, 0 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetMaxScaling (
```

```
    INT idx,          // image handle
```

```
    LPDWORD maxSize, // 0:unbounded, max size otherwise
```

```
    LPINT disableCache // 0:FALSE 1:TRUE
```

```
)  
{  
}
```

qq_GetNumberOfPages - Return number of pages in a file

```
/*$*****  
qq_GetNumberOfPages () - Return number of pages in a file
```

Parameters:

idx: Image handle

Returns:

number of pages
-1 for error

```
*****$*/
```

```
INT _export FAR PASCAL  
qq_GetNumberOfPages (  
INT idx  
)  
{  
}
```

qq_GetPageResolution - get default fax resolution and page size

```
/*$*****  
qq_GetPageResolution () - get default fax resolution and page size
```

Description:

Gets the default page size and resolution of currently opened image. These are the values which will be used when creating a FAX file using qq_ConvertTo().

Also see:

qq_SetPageResolution ()

Returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetPageResolution (  
    INT idx,          // image handle  
    LPINT faxWidth,   // number of columns in a 'FINE resolution' fax page  
    LPINT faxHeight,  // number of lines in a 'FINE resolution' fax page  
    LPINT faxHzRes,   // horizontal resolution (FINE mode)  
    LPINT faxVtRes    // vertical resolution (FINE mode)  
)  
{  
}
```

qq_GetPrintDevice - return name of Active Driver (or empty string)

```
/*$*****  
qq_GetPrintDevice () - return name of Active Driver (or empty string)
```

returns:

length of string

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetPrintDevice (
```

```
LPSTR deviceName, // return name of device
```

```
INT maxLen // maximum length of string
```

```
)
```

```
{
```

```
}
```

qq_GetRotation - get current rotation

```
/*$*****  
qq_GetRotation () - get current rotation
```

Parameters:

idx: Image handle

Returns:

current rotation, as:

- 0: Normal
- 1: 90 degrees (rotate right)
- 2: 180 degrees (upside down)
- 3: 270 degrees (rotate left)
- 1: error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetRotation (
```

```
    INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetScale - get current scale

```
/*$*****  
qq_GetScale () - get current scale
```

Parameters:

idx: Image handle

returns

scale value

100 if image is not initialized

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetScale (
```

```
INT idx
```

```
)
```

```
{
```

```
}
```

qq_GetStretch - get current stretch value

```
/*$*****  
qq_GetStretch () - get current stretch value
```

Parameters:

idx : Image handle

Description:

If stretch set to 100%, then height and width are scaled proportionately.
Otherwise, height is multiplied by the stretch value.

To double the image height, set stretch to 200. To halve the image
height, set stretch to 50.

Returns

stretch value

```
*****$*/  
INT _export FAR PASCAL  
qq_GetStretch (  
    INT idx  
)  
{  
}
```

qq_GetTifBitOrder - get default TIF bit and byte order

```
/*$*****  
qq_GetTifBitOrder () - get default TIF bit and byte order
```

Description:

Gets the default bit and byte ordering for a FAX file type.
(affects only VI_WRT_FAX*, VI_WRT_TIF* file format types).

Defaults are:

bit ordering

- 1: normal bit ordering FAX_NORMAL_BITS
- 2: reverse bit ordering FAX_REVERSE_BITS

byte ordering

- 1: Intel FAX_INTEL
- 2: Motorola FAX_MOTOROLA

Also see:

qq_SetTifBitOrder ()

Returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_GetTifBitOrder (  
    INT idx,          // image handle  
    LPINT bitOrder,  // 0:default 1:normal 2:reversed  
    LPINT byteOrder  // 0:default 1:Intel 2:Motorola  
)  
{  
}
```

qq_GetUsageCount - Check if it is OK to quit application.

```
/*$*****  
qq_GetUsageCount () - Check if it is OK to quit application.
```

Description:

Use this routine to check if it is OK to quit the main application. The possibility exists that one or more routines are still executing (print/write/view).

If the foreground wants to do a quit operation, but the background isn't finished, then the foreground must not process the QUIT command until qq_CheckInUseFlag returns 0.

Windows attempts to do multi-tasking by giving control to the Windows Kernel as often as it can. This occurs during printing, file-io, and during any Dialog related activity.

The QQ engine may be in the middle of an operation when windows gains control, and fires off a message to the owner application indicating the the user has requested that the application quit. QQ functions easily interrupted include qq_ViewReload, qq_ViewRepaint, qq_ConvertTo, qq_OpenFile, etc.

A usage count is kept for each active process attached to the engine, based on the value returned from GetCurrentTask, or GetCurrentProcess. If the usage count is 0, then it is safe to quit. Otherwise, the foreground application must keep processing events until the usage count falls to 0.

Returns:

Number of uncompleted QQ functions currently executing in task.

```
*****$*/  
INT _export FAR PASCAL  
qq_GetUsageCount (  
)  
{  
}
```

qq_GetVersion **ab - Check if it is OK to quit application.**

```
/*$*****  
qq_GetVersion ()
```

Parameters:
None

Returns:
Image Conversion Library version number (high byte / low byte)
*****\$*/

```
INT _export FAR PASCAL  
qq_GetVersion (  
)  
{  
}
```

qq_GetVertDPI - return vertical Dots Per Inch

```
/*$*****  
qq_GetVertDPI () - return vertical Dots Per Inch
```

Paameters:

idx : Image handle

returns

-1 for error, 0 if unavailable, otherwise Dots per Inch.

typical values:

Normal FAX Resolution: 98 DPI

Fine FAX Resolution: 196 DPI

Scanner: 300 DPI

```
*****$*/
```

INT _export FAR PASCAL

qq_GetVertDPI (

INT idx

```
)  
{  
}
```

qq_LoadBitmap - return a bitmap image

```
/*$*****  
qq_LoadBitmap () - return a bitmap image
```

Parameters:

idx: Image handle for image to load bitmap from
hPal: Palette to be loaded with the bitmap

Description:

This loads a bitmap at the current image scaling.

You are responsible for deleting the bitmap after you are finished with it. To delete a bitmap, call the windows API DeleteObject (hBitmap)

Example:

```
nIdx = qq_OpenFile (lpzFilename);  
qq_SetScalingOptions (nIdx, nScale, nStretch);  
hBitmap = qq_LoadBitmap (nIdx, 0);  
qq_CloseFile(nIdx);
```

... use bitmap ...

```
DeleteObject (hBitmap);
```

returns

0 for error, handle to bitmap otherwise

```
*****$*/  
HBITMAP _export FAR PASCAL  
qq_LoadBitmap (  
    INT idx,          // currently active file  
    HPALETTE hPal     // Logical palette (to be realized to bitmap)  
)  
{  
}
```

qq_LoadPalette - return a palette value

```
/*$*****  
qq_LoadPalette () - return a palette value  
  
returns  
0 for error, handle to HPALETTE otherwise  
*****$*/  
HPALETTE _export FAR PASCAL  
qq_LoadPalette (  
    INT idx,           // currently active image handle  
    HDC hAppDC        // Sample Device Context (or NULL)  
)  
{  
}
```

qq_OpenFile - Initialize instance variables, open file for read

```
/*$*****  
qq_OpenFile () - Initialize instance variables, open file for read
```

Parameters:

fname: far pointer to string value to read

Returns:

idx: Image handle; handle for instance data.
-1 for error

Description:

Ensures that there is storage for the instance data; allocates instance data space of the global heap if not. Initializes instance; opens file for read.

Notes:

To support Folders (OLE containers):
The file name can contain two additional parameters:
1) start offset within file
2) length of object within file

This information is formatted as follows:
filename;StartOffset;Len

There are no spaces allowed after the semi-colon.

This routine does not handle opening annotation files. To open an annotation file, use qq_AnnOpenFile().

```
*****$*/
```

```
INT _export FAR PASCAL  
qq_OpenFile (  
    LPSTR fname  
)  
{  
}
```

qq_PasteToFile - Paste clipboard contents into a file

```
/*$*****  
qq_PasteToFile() - Paste clipboard contents into a file
```

This routine will paste the contents of the clipboard (if it is a bitmap or a DIB) into a specified file as a bitmap.

Parameters:

szFile : File name
pRect : (OPTIONAL) Size of bitmap image

Returns:

FALSE on failure to create or save file, or non-bitmap, or non-DIB

Note: The bitmap is stored in the file unscaled. It is written as a .BMP file.

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_PasteToFile(  
    LPSTR szFile,  
    LPRECT pRect)
```

```
{  
}
```

qq_SetBackground - Set Black on White or White on Black

```
/*$*****  
qq_SetBackground () - Set Black on White or White on Black
```

Parameters:

idx: Image handle
reflect: Background option: False (0) is black text on white BG

Description:

Set Black on White (default), or reverse to White on Black
Compression options may also have to be changed to get optimum results

Default is FALSE (0)

Returns:

old value

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetBackground (
```

```
    INT idx,
```

```
    INT reflect      // Default: set to FALSE for black text on white bkg
```

```
)
```

```
{
```

```
}
```

qq_SetDevMode - set device mode for printer

```
/*$*****  
qq_SetDevMode () - set device mode for printer
```

Sets the hDevMode element of the PRINTDLG structure that we use to control printing for this task.

This function is used in applications that want to control how the print dialog box comes up (MFC applications in particular). If the print dialog box is opened by the parent application, the associated data is also purely owned by the parent application, and the library has no way of retrieving it; the library can open a second dialog box, but requiring the user to answer the same questions a second time is not a reasonable plan. If your application calls PrintDlg(), or if your application is an MFC application that uses the default OnPreparePrint or DoPreparePrinting method, you must pass a pointer to the device mode whose handle is returned by PrintDlg() in the PRINTDLG structure element hDevMode (or the value returned by the member function CPrintInfo::GetDevMode()) to the library using this function before you call qq_Print().

Typical call using MFC:

```
void CView::OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo )  
{  
    ...  
  
    LPDEVMODE lpdv = pInfo->m_pPD->GetDevMode();  
    DevSet = qq_SetDevMode(lpdv);  
    GlobalUnlock(lpdv);  
    if (!DevSet) {  
        // error handling  
    }  
    ...  
}
```

returns:

TRUE (non-0) on success
FALSE (0) on failure

```
*****$*/  
INT _export FAR PASCAL  
qq_SetDevMode(  
    LPDEVMODE lpDevMode)  
{  
}
```

qq_SetMaxScaling - sets the maximum allowable bitmap size

```
/*$*****  
qq_SetMaxScaling () - sets the maximum allowable bitmap size
```

Description:

This routine allows the foreground application to set the memory parameters for the image. Two parameters are supported:
maxSize - maximum number of bytes to allocate for image
cacheFlag - if OFF, then do not maintain an uncompressed version of image in memory.

The cache is kept for the last accessed image only, and is used to speed up display updates to the image. If disabled, then all scale/rotate updates are read directly from disk. If cache enabled, then all scale/rotate updates are read from memory. Typically a fax image requires 500K to store an uncompressed version of the image.

maxSize is checked to ensure that the value is within an acceptable range. The minimum size is 32K (anything smaller doesn't do anyone any favours).

Also see:

qq_GetMaxScaling (), qq_GetScale (), qq_SetScalingOptions ()

Source Sample:

```
CHAR fname = "C:\\garp.tif";  
INT idx;  
DWORD maxMemSize = 100000L; // maximum of 100K for bitmap  
INT disableCache = TRUE // Don't keep unscaled image in memory  
INT scale = 100; // set to 100%  
INT stretch = 100; // set to normal  
  
idx = qq_OpenFile (fname);  
if (idx >= 0) {  
    qq_SetMaxScaling (idx, maxMemSize, disableCache);  
    qq_SetScalingOptions (idx, scale, stretch)  
    if (scale > qq_GetScale (idx)) {  
        OutputDebugString ("Scale reduced to maxMemSize\n");  
    }  
}  
qq_CloseFile (idx);
```

Returns:

-1 for failure, 0 otherwise

```
*****$*/  
INT _export FAR PASCAL  
qq_SetMaxScaling (  
    INT idx, // image handle  
    DWORD maxSize, // 0:unbounded, max size otherwise  
    INT disableCache // 0:FALSE 1:TRUE  
)  
{  
}
```


qq_SetMirror - Set horizontal mirror option

```
/*$*****  
qq_SetMirror () - Set horizontal mirror option
```

Parameters:

idx: Image handle
hzMirror: Horizontal mirror option: TRUE (non-0) is left-right swapped.

Description:

Set horizontal mirror options.
Vertical mirroring is not directly supported. To mirror vertically, mirror horizontally, and rotate 180 degrees

Default is FALSE (0)

Returns:

old value

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetMirror (  
    INT idx,  
    INT hzMirror  
)  
{  
}
```

qq_SetPage - Select page to view

```
/*$*****  
qq_SetPage () - Select page to view
```

Parameters:

idx: Image handle
newPage: New page to view

Description:

Sets specified page. Results in the view information being updated.

Default is 0

Notes:

Page number is restricted to number of pages in the loaded file.

In some cases, errors in changing pages on annotation files will leave the image with no associated image file. If changing pages on an annotation file results in an error, we recommend immediately resetting to page 0, which should always result in a valid image, or else closing and re-opening the annotation file.

Returns:

Previously displayed page
-1 on error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetPage (  
    INT idx,  
    INT newPage  
)  
{  
}
```

qq_SetPageResolution - set default fax resolution and page size

```
/*$*****  
qq_SetPageResolution () - set default fax resolution and page size
```

Description:

Sets the default page size and resolution for the currently opened image. These values are used when creating a TIF or FAX file using qq_ConvertTo().

The default used by qq_ConvertTo is based on the currently opened page. If the currently opened image is a TIFF file, then the default resolution is set to the same values as are contained in the opened image. (If you open a 300x300 dpi image, then you will write out a 300x300 dpi image). Otherwise resolution is set to a standard fax resolution:

hz resolution: 203
vt resolution: 198

qq_ConvertTo assumes that pages within the image all have the same horizontal and vertical resolution, or a mixed case of standard and fine resolution images. If each page in the input file has a different resolution, then it is best for you to do a 'Set Page' for each page, and write out individual files, (limit the start and end page in qq_ConvertTo() to one page at a time).

If the output format is FAX, then all pages are scaled to fit the default height/width. (1728/2160). If the output is TIF, then all images are written at the current scaling, irrespective of there being different page resolutions in the image. To output a fax image at any size other than 1728/2160 (irrespective of output resolution, or input page dimensions) requires manually setting the default values.

Vertical and horizontal resolution are meant for reference only. These numbers do not affect how images are written or read, other than to indicate what the scanning resolution was.

If saving a file as a Low resolution FAX/TIF file, the vertical resolution value is automatically halved.

When viewing a file, if the vertical resolution is approximately half of the horizontal resolution, then the image height is doubled.

Note:

The defaults are re-set every time you switch pages. To ensure defaults are updated, make this call just before you make the call to qq_ConvertTo.

Also see:

qq_ConvertTo ()
qq_GetPageResolution ()

Returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetPageResolution (  
    INT idx,          // image handle  
    INT faxWidth,     // number of columns in a 'FINE resolution' fax page  
    INT faxHeight,    // number of lines in a 'FINE resolution' fax page  
    INT faxHzRes,     // horizontal resolution (FINE mode)
```

```
INT faxVtRes // vertical resolution (FINE mode)
)
{
}
```

qq_SetRotation - set current rotation

```
/*$*****  
qq_SetRotation () - set current rotation
```

Parameters:

idx: Image handle
rotation: New rotation value
0: Normal
1: 90 degrees (rotate right)
2: 180 degrees (upside down)
3: 270 degrees (rotate left)

returns

previous rotation, -1 for error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetRotation (
```

```
    INT idx,
```

```
    INT rotation
```

```
)
```

```
{
```

```
}
```

qq_SetScalingOptions age to view

```
/*$*****  
qq_SetScalingOptions ()
```

Description:

Set the horizontal and vertical scaling parameters for the image.

If qq_SetMaxScale () has been set, then scaling is adjusted not to exceed the maximum amount.

To set the Horizontal and Vertical Scaling

Modify the 'scale' parameter. Keep stretch set to 100.

This is the default mode, as most users want to scale up or down in both directions.

To set the Vertical Scaling only:

Modify the 'stretch' parameter. Set it to 200 to double the height. Set it to 50 to half the height.

To set the Horizontal Scaling only:

As you modify the 'scale' parameter, you must adjust the 'stretch' to correct. As you decrease the width, increase the stretch by the same percentage according to the following formula:

$$\text{new_stretch} = \text{old_stretch} * (\text{old_scale} / \text{new_scale})$$

New Stretch	Old_strech	old_scale	new_scale	comments
200	100	100	50	decrease width by 50%
111	100	100	90	decrease by 10
124	111	90	80	decrease another 10
142	124	80	70	decrease another 10
165	142	70	60	decrease another 10
200	165	60	50	decrease another 10
100	200	50	100	increase by 100%

See Also:

qq_SetMaxScaling(), qq_GetScale(), qq_GetStretch()

Returns:

0 for success, -1 otherwise

```
*/
```

```
INT _export FAR PASCAL
```

```
qq_SetScalingOptions (
```

```
    INT idx,      // Image Handle
```

```
    INT scale,    // Overall image scaling: scale for height and width (100 is normal)
```

```
    INT stretch // Vertical stretch: 100 for normal, 200 to double
```

```
                // height, 50 to set to half height
```

```
)
```

```
{
```

```
    DWORD val;
```

```

if (idx < 0 || !img_buf || !img_buf[idx].used)
    return (-1);

qq_ViewRemoveSelectionBox(-1);

val = (DWORD) scale * (DWORD) stretch / 100L;

img_buf[idx].stretch = stretch;
img_buf[idx].mapInfo.nHscale = scale;
img_buf[idx].mapInfo.nVscale = (INT) val;
qqSetMapInfo (idx);          // re-calcs bitmapSize

// force bitmap to minimal size if maxBitmapSize is set
if (img_buf[idx].maxBitmapSize) {
    while ((img_buf[idx].bitmapSize > img_buf[idx].maxBitmapSize) && (scale > 5)) {
        scale--;
        val = (DWORD) scale * (DWORD) stretch / 100L;
        img_buf[idx].mapInfo.nHscale = scale;
        img_buf[idx].mapInfo.nVscale = (INT) val;
        qqSetMapInfo (idx);          // re-calcs bitmapSize
    }
}
return (0);
}
}

```

```

/*$
qq_SetCompressionOptions () - Set enhancement used on image compression

```

Parameters:

idx : Image handle
enhanceFlag : Options to use when compressing an image

Description:

Sets enhancement technique used when compressing FAX image for display at a smaller scale. Values for enhance_flag, with the corresponding operations done in the Virtual Image engine, are as follows:

FAX_ENHANCE_WHITE: 0
VI_SC_OR

FAX_NORMAL 1
0 (no enhancements)

FAX_ENHANCE_HZ: 0x2
VI_SC_OR | VI_SC_PRE_REV | VI_SC_POST_REV | VI_SC_OR_HZ_ONLY

FAX_ENHANCE_VT: 0x3
VI_SC_OR | VI_SC_PRE_REV | VI_SC_POST_REV | VI_SC_OR_VT_ONLY

FAX_GRAY_SCALE: 0x4 // switch to gray scale
VI_SC_ADD|VI_SC_OR|VI_SC_PRE_REV|VI_SC_POST_REV;

FAX_ENHANCE_BLACK: 0x5
VI_SC_OR | VI_SC_PRE_REV | VI_SC_POST_REV;

```

VI_SC_OR      0x8000 // or bits when compressing
VI_SC_PRE_REV 0x4000 // reverse image before compressing
VI_SC_POST_REV 0x2000 // reverse image after compressing
VI_SC_ADD     0x1000 // add black bits when compressing
VI_SC_OR_VT_ONLY 0x0800 // only add black bits vertically
VI_SC_OR_HZ_ONLY 0x0400 // only add black bits horizontally

```

You may also OR together the VI_SC constants above to produce a custom enhancement technique and pass that value in via enhanceFlag.

Returns:

```

0 for success, -1 otherwise
*****$*/
INT _export FAR PASCAL
qq_SetCompressionOptions (
    INT idx,
    INT enhanceFlag
)
{
}

```

qq_SetTifBitOrder - set default bit and byte ordering

```
/*$*****  
qq_SetTifBitOrder () - set default bit and byte ordering
```

Description:

FAX tiff files can have either normal bit ordering, or reversed bit ordering; and Intel byte ordering, or Motorola byte ordering

Intel byte ordering: Ints: Low byte / High byte
 Longs: Low int / High int

Motorola byte ordering: Ints: High byte / Low byte
 Longs: High int / Low int

For normal bit ordering, bits are read

76543210.76543210. etc

For reversed bit ordering, bits are read

01234567.01234567

Also see:

qq_GetTifBitOrder ()

Returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_SetTifBitOrder (
```

```
  INT idx,            // image handle
```

```
  INT bitOrder,       // 0:default 1:normal 2:reversed
```

```
  INT byteOrder       // 0:default 1:Intell 2:Motorolla
```

```
)
```

```
{
```

```
}
```

qq_UpdateBitmap - update the current bitmap

```
/*$*****  
qq_UpdateBitmap () - update the current bitmap
```

Parameters:

idx: Image handle
hBitmap: Handle of bitmap to be used as source of update
hPal: Logical bitmap to be realized to bitmap, or NULL if none
pUpdateRect: far pointer to update area, or NULL to update entire rectangle

Description:

Update the specified bitmap as defined by the update rectangle.
Top: start row
Left: start col
width: right - left
height: bottom - top

returns:

0 for success, -1 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_UpdateBitmap (
```

```
    INT idx,          // currently active file  
    HBITMAP hBitmap, // display bitmap  
    HPALETTE hPal,   // Logical palette (to be realized to bitmap) or NULL  
    LPRECT pUpdateRect // Display area to be updated (or NULL for everything)
```

```
)  
{  
}
```

qq_ViewAnnDeleteAnnotation - Delete a specified annotation

```
/*$*****  
qq_ViewAnnDeleteAnnotation() - Delete a specified annotation
```

Given a view handle and an annotation handle, this routine will delete the annotation from the view..

Parameters:

vdx : View handle
ann : Annotation to delete

Returns:

0 on success;
-1 if not an annotation or not supported;
Annotation error code (non-0) if deletion failed

Note: Requires annotation support, and requires that the view originated from an annotation file.

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnDeleteAnnotation(  
    INT vdx,          /* View handle */  
    HANNOTATION ann) /* Annotation handle */  
{  
}
```

qq_ViewAnnDeleteSelected - Delete selected annotations

```
/*$*****  
qq_ViewAnnDeleteSelected() - Delete selected annotations
```

Given a view handle, this routine will delete all selected annotations from the view..

Parameters:
vdx : View handle

Returns:
0 on success (or no annotations to delete);
-1 if not an annotation or not supported;
Annotation error code (non-0) if deletion failed

Note: Requires annotation support, and requires that the view originated from an annotation file.

Note: A non-0 return code may indicate that not all selected annotations have been deleted. If there are five selected annotations, and deletion of the third one in the list fails, deletion of the fourth and fifth annotations in the list will not be attempted.

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnDeleteSelected(  
    INT vdx) /* View handle */  
{  
}
```

qq_ViewAnnGetArowObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetArowObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetArowObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_AROW pArow  
)  
{  
}
```

qq_ViewAnnGetBmpObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetBmpObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetBmpObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_BMP pBmp)  
{  
}
```

qq_ViewAnnGetCircObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetCircObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetCircObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_CIRC pCirc  
)  
{  
}
```

qq_ViewAnnGetCurrentProcess - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetCurrentProcess() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetCurrentProcess(  
INT vdx  
)  
{  
}
```

qq_ViewAnnGetFirstBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetFirstBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetFirstBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetFirstSelectedBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetFirstSelectedBase() - Get the xxx specific properties
```

To be filled in...

```
*****$/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetFirstSelectedBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetHiliteObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetHiliteObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetHiliteObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_RECT pRect  
)  
{  
}
```

qq_ViewAnnGetIndexBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetIndexBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetIndexBase(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetLastBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetLastBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetLastBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetLastSelectedBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetLastSelectedBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetLastSelectedBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetNextBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetNextBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetNextBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetNextSelectedBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetNextSelectedBase() - Get the xxx specific properties
```

To be filled in...

```
*****$/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetNextSelectedBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetPostObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetPostObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetPostObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_POST pPost  
)  
{  
}
```

qq_ViewAnnGetPostStringProperty - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetPostStringProperty() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetPostStringProperty(  
INT vdx,  
HANNOTATION hAnn,  
LPSTR lpString,  
INT maxLen  
)  
{  
}
```

qq_ViewAnnGetPrevBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetPrevBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetPrevBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetPrevSelectedBase - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetPrevSelectedBase() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnGetPrevSelectedBase(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewAnnGetRectObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnGetRectObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnGetRectObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_RECT pRect  
)  
{  
}
```

qq_ViewAnnPaste - Paste clipboard contents into this annotation

```
/*$*****  
qq_ViewAnnPaste() - Paste clipboard contents into this annotation
```

Given a view handle, this routine will paste the contents of the clipboard (if it is a bitmap or a DIB) into the current annotation set as a bitmap annotation, or (if it is available as text) as a Post-It annotation. Future development: if the clipboard contents are an annotation, it will paste the annotation into the current annotation set.

Parameters:
vdx : View handle

Returns:
Handle of new annotation on success
0 on failure or annotation not supported;

Note: Requires annotation support, and requires that the view originated from an annotation file.

Note: Annotation requires that bitmaps come from a file. A DIB or bitmap item in the clipboard is first saved as a temporary file; when the annotation is saved, these bitmaps are forced to be embedded in the saved file, and the temporary files are deleted when the annotation is closed. NOTE THE SIDE EFFECTS: Specifying Save linked is overridden for these annotations only; and later specifying Save Linked (or Extract) will result in bitmap files with obvious temporary-file names being extracted and stored in the annotation file's directory.

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewAnnPaste(  
    INT vdx)          /* View handle */  
{  
}
```

qq_ViewAnnSetArowObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetArowObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetArowObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_AROW lpArow  
)  
{  
}
```

qq_ViewAnnSetBmpObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetBmpObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetBmpObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_BMP lpBmp  
)  
{  
}
```

qq_ViewAnnSetCircObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetCircObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetCircObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_CIRC lpCirc  
)  
{  
}
```

qq_ViewAnnSetHiliteObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetHiliteObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetHiliteObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewAnnSetPostObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetPostObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetPostObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_POST lpPost,  
LPSTR lpString  
)  
{  
}
```

qq_ViewAnnSetRectObj - Get the xxx specific properties

```
/*$*****  
qq_ViewAnnSetRectObj() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnSetRectObj(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewAnnTranslateRect- Translate from real to image coordinates

```
/*$*****  
qq_ViewAnnTranslateRect() - Translate from real to image coordinates
```

Given a view handle and a rectangle, this routine will translate the rectangle from real to view co-ordinates or vice versa.

Parameters:

vdx : View handle
RectIn : Rectangle to translate
RectOut: Translated rectangle
ToView : if TRUE, translate from REAL to VIEW co-ordinates
if FALSE, translate from VIEW to REAL co-ordinates

Returns:

0 on success; -1 if not an annotation or not supported.

Note: Requires annotation support, and requires that the view originated from an annotation file.

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnnTranslateRect(  
    INT vdx,  
    RECT RectIn,  
    RECT FAR *RectOut,  
    INT ToView)  
{  
}
```

qq_ViewAnythingSelected- Determine if anything is selected

```
/*$*****  
qq_ViewAnythingSelected () - Determine if anything is selected
```

Description:

Get the dimensions of the selection box. Pass a VDX handle of -1 to get the first selection box available.

Returns:

-1 if error or nothing selected
vdx handle otherwise (fills in rect dimensions if not NULL)

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewAnythingSelected(  
    INT vdx,          // requested vdx, or -1 to check them all  
    LPRECT lpRect  
)  
{  
}
```

qq_ViewClose - cleans up the 'View' object

```
/*$*****  
qq_ViewClose () - cleans up the 'View' object
```

Description:

Release the current View object.
Delete the main window
Delete the Bitmap and Palette
Delete the cursor resources
Delete the Hz and Vt scroll bars

Returns:

Success (0) or Failure (-1)

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewClose (  
    INT vdx          // index to view object  
)  
{  
}
```

qq_ViewCopyToClipboard - copies current selection box to the clipboard

```
/*$*****  
qq_ViewCopyToClipboard () - copies current selection box to the clipboard
```

Description:

Copies the contents either at current scaling (and rotation), or copies at 100% scaling

If the image hasn't been currently loaded, then do so.

If background flag is set, then wait for background to complete before cutting.

If '100%' scaling, and image size is different, then build a separate bitmap at 100%, then cut.

Assumptions: Assumes there is a valid Selection Region.

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewCopyToClipboard (  
    INT vdx,  
    INT mode          // 0: current scaling 1:100% scaling  
)  
{  
}
```

qq_ViewDoBackground - handles background updates

```
/*$*****  
qq_ViewDoBackground() - handles background updates
```

Description:

Handles background updates for all background views.

If Background Updates is TRUE, then the image is painted to the screen in bands. As the next band is loaded, it is painted to the screen. Depending on the orientation of the image, it is either painted right to left, or top to bottom.

Background images are painted in order. The topmost image is the last image to be re-loaded (lastVdx)

To support background painting, you must call the 'ViewDoBackgroundTick' routine when idle in your main event loop:

This function will only handle background updates for the currently active process. Each process has its own list of active opened handles.

Example:

```
// setup for background display updates  
ret = qq_ViewSetBackground (vdx, 1);  
  
while (GetMessage (&msg, 0, 0, 0)) {  
    TranslateMessage(&msg);  
    DispatchMessage(&msg);  
  
    // handle background updates  
    while (!peekMessage (&msg, 0, 0,0, PM_NOREMOVE)) {  
        if (!qq_ViewDoBackground ())  
            break;          // nothing left to do  
    }  
}
```

See Also

qq_BuildBitmap(), qq_UpdateBitmap().

Returns:

True if more work to do

False otherwise

```
*****$*/
```

```
INT _export FAR PASCAL  
qq_ViewDoBackground ()  
{  
}
```

qq_ViewGetArowDefaultProperties - Get the xxx specific properties

```
/*$*****  
qq_ViewGetArowDefaultProperties() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetArowDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_AROW lpArow  
)  
{  
}
```

qq_ViewGetCreatedAnnProperties - Get the base properties

```
/*$*****  
qq_ViewGetCreatedAnnProperties() - Get the base properties
```

To be filled in...

```
*****$*/  
HANNOTATION _export FAR PASCAL  
qq_ViewGetCreatedAnnProperties(  
INT vdx,  
PPROPERTY_LIST_BASE pProp  
)  
{  
}
```

qq_ViewGetHiliteDefaultProperties - Get the xxx specific properties

```
/*$*****  
qq_ViewGetHiliteDefaultProperties() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetHiliteDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewGetIdx - returns the assigned idx value

```
/*$*****  
qq_ViewGetIdx () - returns the assigned idx value
```

Description:

Returns the current image handle, or -1 for none. The current image handle is used to do repaints, updates, and reloads.

returns:

Image handle, or -1 if none or for failure

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewGetIdx (
```

```
INT vdx
```

```
)  
{  
}
```

qq_ViewGetMode - gets current view mode

```
/*$*****  
qq_ViewGetMode () - gets current view mode
```

Returns:

```
    -1 // error (invalid vdx)  
MODE_NONE      0 // mouse has no effect  
MODE_SELECT    1 // mouse draws selection box  
MODE_MOVE_WINDOW 2 // mouse moves window  
MODE_MOVE_IMAGE 3 // mouse moves image (hand cursor)  
MODE_MAGNIFY    4 // mouse draws magnification box  
MODE_ANN_SELECT 5 // mouse selects annotations  
MODE_ANN_CIRC   6 // mouse creates a circle  
MODE_ANN_HILITE 7 // mouse creates a rectangle  
MODE_ANN_BMP    8 // mouse creates a bitmap  
MODE_ANN_POST   9 // mouse creates a Post-It note  
MODE_ANN_CLINE 10 // mouse creates a curved line  
MODE_ANN_SLINE 11 // mouse creates a straight line/arrow  
MODE_ANN_COVERPAGE 12 // mouse creates a coverpge template
```

Description:

Returns the current view mode which was set using qq_ViewSetMode().

See Also

qq_ViewSetMode ()

```
*****$*/
```

```
INT _export FAR PASCAL  
qq_ViewGetMode (  
INT vdx  
)  
{  
}
```

qq_ViewGetPosition - returns the x/y cx,cy size of display window/scroll bars

```
/*$*****  
qq_ViewGetPosition() - returns the x/y cx,cy size of display window/scroll bars
```

Returns the bounding box which surrounds the Display Window/Cursor control

returns:

```
-1 for failure, 0 otherwise  
*****$/  
INT _export FAR PASCAL  
qq_ViewGetPosition (  
INT vdx,  
FPINT x, // left hand edge  
FPINT y, // top edge  
FPINT cx, // box width  
FPINT cy // box height  
)  
{  
}
```

qq_ViewGetPostItDefaultProperties - Get the postIt specific properties

```
/*$*****  
qq_ViewGetPostItDefaultProperties() - Get the postIt specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetPostItDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_POST lpPost  
)  
{  
}
```

qq_ViewGetPostItDefaultStringProperty - Get the xxx specific properties

```
/*$*****  
qq_ViewGetPostItDefaultStringProperty( - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetPostItDefaultStringProperty(  
INT vdx,  
LPSTR lpString,  
INT maxLen  
)  
{  
}
```

qq_ViewGetRectDefaultProperties - Get the xxx specific properties

```
/*$*****  
qq_ViewGetRectDefaultProperties() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetRectDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewGetScrollPos - get the X/Y offset of the display bitmap

```
/*$*****  
qq_ViewGetScrollPos () - get the X/Y offset of the display bitmap
```

Description:

Gets the x and y offset of the display bitmap within the current window.

The X and Y values are positive increments if moving the bitmap to the left, or up in the display window.

See Also

qq_ViewSetScrollPos ()

Returns:

-1 for failure, 0 otherwise

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewGetScrollPos (  
INT vdx,          // index to view object  
FPINT Xpos,      // x offset (0 for default, positive to move left)  
FPINT Ypos       // y offset (0 for default, positive to move up)  
)  
{  
}
```

qq_ViewGetWindow - returns the specified window handle (display/hScroll/VScroll)

```
/*$*****  
qq_ViewGetWindow () - returns the specified window handle (display/hScroll/VScroll)
```

Notes:

Supported Window types include:

```
VIEW_DISP_WINDOW 0 // main display window
```

returns:

Handle to Window, or 0 for none

```
*****$*/
```

```
HWND _export FAR PASCAL
```

```
qq_ViewGetWindow (
```

```
INT vdx,
```

```
INT nWindowType // 0:main 1:hz scroll bar 2:vt scroll bar
```

```
)
```

```
{
```

```
}
```

qq_ViewHandleEvent - handles the event message

```
/*$*****  
qq_ViewHandleEvent() - handles the event message
```

Description:

Handles events for all active Vdx windows. Called internally from the window functions.

You can optionally call this function externally if you are processing 'Peek' messages on your own. Otherwise, it is best to let TranslateMessage() and DispatchMessage() do its thing, and let the window handlers do their thing.

Returns:

TRUE if event handled
FALSE if event not handled/recognized

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewHandleEvent(  
    MSG FAR * lpMsg  
)  
{  
}
```

qq_ViewOpen - builds a 'View' object to support image display

```
/*$*****  
qq_ViewOpen () - builds a 'View' object to support image display
```

Description:

Sets up the View object (but doesn't paint it)
This includes building the main view window, the vertical and horizontal scroll bars, loading the default cursor handles, and positioning the display box coordinates.

If the View Window class does not exist, then create it.

If a parent window is defined, the following messages are sent from the child View window:

Wparam contains the VDX handle

Lparam contains additional details

```
QQ_WM_CHAR          WM_USER + 2  // virtual Key code  
QQ_WM_KEYDOWN       WM_USER + 3  // virtual key code  
QQ_WM_KILLFOCUS     WM_USER + 4  // window getting focus  
QQ_WM_SETFOCUS      WM_USER + 5  // window loosing focus  
QQ_WM_MOUSEMOVE     WM_USER + 6  // xPos=Loword yPos=Hiword  
QQ_WM_LBUTTONDOWN   WM_USER + 7  // xPos=Loword yPos=Hiword  
QQ_WM_LBUTTONUP     WM_USER + 8  // xPos=Loword yPos=Hiword  
QQ_WM_RBUTTONDOWN   WM_USER + 9  // xPos=Loword yPos=Hiword  
QQ_WM_RBUTTONUP     WM_USER + 10 // xPos=Loword yPos=Hiword  
QQ_WM_ANN_GETCURSEL WM_USER + 11 // number of ann's selected
```

See also:

qq_OpenFile () - opens a file for viewing
qq_ViewSetIdx () - assigns a valid idx filename to the view window
qq_ViewReload () - forces the display bitmap to be re-loaded
qq_ViewRepaint () - forces the current window to be repainted
qq_ViewClose () - closes down the view object

Equivalent Function:

To control background color, use qq_ViewOpenEx ()

Returns:

Vdx - index handle to view object
-1 for error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewOpen (  
    HWND hParent,    // handle to parent window (or NULL)  
    HANDLE hInst,    // handle to application instance  
    INT x,           // left side  
    INT y,           // top side  
    INT cx,          // width  
    INT cy,          // height  
    DWORD style      // window style ie:WS_BORDER  
)  
{
```

}

qq_ViewOpenEx - builds a 'View' object to support image display

```
/*$*****  
qq_ViewOpenEx () - builds a 'View' object to support image display
```

Description:

see qq_ViewOpen () (equivalent function)

Additional parameters are as follows:

Background Window Color:

parameter used by GetStockObject()

LTGRAY_BRUSH is optimal

Unused

Reserved for future use. Must be 0

Returns:

Vdx - index handle to view object

-1 for error

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewOpenEx (
```

```
    HWND hParent,    // handle to parent window (or NULL)
```

```
    HANDLE hInst,    // handle to application instance
```

```
    INT x,           // left side
```

```
    INT y,           // top side
```

```
    INT cx,          // width
```

```
    INT cy,          // height
```

```
    DWORD style,     // window style ie:WS_BORDER
```

```
    INT32 fnObject,  // Window Background: BLACK_BRUSH/WHITE_BRUSH/GRAY_BRUSH, etc
```

```
    DWORD futureUse  // must be zero
```

```
)  
{  
}
```

qq_ViewReload - force image to be reloaded (new idx values)

```
/*$*****  
qq_ViewReload ()    - force image to be reloaded (new idx values)
```

Parameters:

 vdx : View handle

Description:

The current Bitmap and Palette are released, and a new Bitmap and Palette are generated.

If background mode is enabled, then the new bitmap is created but is empty.

This call invalidates the Window rectangle, eventually resulting in a qq_ViewRepaint () call from within the qq_ViewHandleEvent() routine.

returns:

 -1 for failure, 0 otherwise

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewReload (  
  INT vdx  
)  
{  
}
```

qq_ViewRepaint - force image to be repainted

```
/*$*****  
qq_ViewRepaint () - force image to be repainted
```

Description:

Repaints the display screen, re-sets the scroll bars. Normally called from qq_ViewHandleEvent().

This call is useful if you want to force an immediate display update. Otherwise, you can do an 'invalidate hWnd', and wait for the event loop to handle the update.

returns:

-1 for failure, 0 otherwise

```
*****$/  
INT _export FAR PASCAL  
qq_ViewRepaint (  
    INT vdx,  
    LPRECT updateRect // local Display Window Coordinates (or NULL)  
)  
{  
}
```

qq_ViewSetArowDefaultProperties - Get the xxx specific properties

```
/*$*****  
qq_ViewSetArowDefaultProperties() - Get the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetArowDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_AROW lpArow  
)  
{  
}
```

qq_ViewSetBackground - sets background mode for Vdx

```
/*$*****  
qq_ViewSetBackground() - sets background mode for Vdx
```

Set the background mode for the currently open Vdx object

If set to 1:background, then the screen isn't repainted until the frontend calls qq_ViewDoBackground. Otherwise, the image is loaded immediately after any change to settings.

See Also

qq_ViewDoBackground () - handles background updates

returns:

- 1 if error
- 0 if off
- 1 if on

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetBackground(  
  INT vdx,  
  INT bMode          // 0: normal 1: background
```

```
)  
{  
}
```

qq_ViewSetCreatedAnnProperties - Set the base properties

```
/*$*****  
qq_ViewSetCreatedAnnProperties() - Set the base properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetCreatedAnnProperties(  
INT vdx,  
HANNOTATION hAnn,  
PPROPERTY_LIST_BASE lpBase, // base properties (must not be NULL)  
LPVOID lpProp, // object-specific properties  
LPSTR lpText // text string or NULL if not a PostIt  
)  
{  
}
```

qq_ViewSetCursor - sets the cursor resource for the specified mode

```
/*$*****  
qq_ViewSetCursor () - sets the cursor resource for the specified mode
```

Description:

Default cursor resorces are stored in IMGCONVT.DLL; use this function to over-ride the defaults

Each View object has its own cursor resource (one for each mode). Use this to over-ride the standard cursor.

ToDo:

Who deletes the cursor object??

returns:

-1 for error, 0 otherwise

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetCursor (  
    INT Vdx,  
    HCURSOR hCursor,  
    INT mode  
)  
{  
}
```

qq_ViewSetHiliteDefaultProperties - Set the xxx specific properties

```
/*$*****  
qq_ViewSetHiliteDefaultProperties() - Set the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetHiliteDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewSetIdx - assigns an image to a view for display

```
/*$*****  
qq_ViewSetIdx () - assigns an image to a view for display
```

Description:

Assign the currently opened Idx handle to the View object.

Scaling is set to the requested default

```
LOAD_CURRENT_DEFAULT 0 // keep at current scaling  
LOAD_FIT_TO_WIDTH    1 // scale image to fit width of display  
LOAD_FIT_TO_HEIGHT   2 // scale image to fit height of display  
LOAD_FIT_TO_PAGE     3 // scale image to fit within page
```

The bitmap and palette are not loaded until qq_ViewReload is called. You need to do this at least once after you have set up the image, and every time thereafter when you make a change to one of the Image attributes (page number, scaling, rotation, enhancement, etc).

To clear display, set the Idx handle to -1, then call qq_ViewRepaint().

qq_ViewReload forces the image to be repainted, and the scroll bars to be updated.

See also:

qq_ViewReload () - forces the display bitmap to be re-loaded
qq_ViewRepaint () - forces the current window to be repainted

Returns:

-1 for failure, 0 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetIdx (  
    INT vdx,      // index to view object  
    INT idx,      // index to image object  
    INT scaleMode // LOAD_FIT_TO_WIDTH, etc  
)  
{  
}
```

qq_ViewSetMode - sets current view mode

```
/*$*****  
qq_ViewSetMode () - sets current view mode
```

Description:

Set the mode for the currently specified View Window. Sets the window cursor to be one of the selected modes. If the user then presses the left cursor button down, the mode is activated.

```
        -1 // error (invalid vdx)  
MODE_NONE      0 // mouse has no effect  
MODE_SELECT    1 // mouse draws selection box  
MODE_MOVE_WINDOW 2 // mouse moves window  
MODE_MOVE_IMAGE 3 // mouse moves image (hand cursor)  
MODE_MAGNIFY    4 // mouse draws magnification box  
MODE_ANN_SELECT 5 // mouse selects annotations  
MODE_ANN_CIRC   6 // mouse creates a circle  
MODE_ANN_HILITE 7 // mouse creates a rectangle  
MODE_ANN_BMP    8 // mouse creates a bitmap  
MODE_ANN_POST   9 // mouse creates a Post-It note  
MODE_ANN_CLINE  10 // mouse creates a curved line  
MODE_ANN_SLINE  11 // mouse creates a straight line/arrow  
MODE_ANN_COVERPAGE 12 // mouse creates a coverpge template
```

To avoid confusion, the default is to set the same mode for all other windows associated with the current process.

The following messages are sent to the parent application:

```
PostMessage (hParent, QQ_WM_LBUTTONDOWN, vdx, lParam);  
PostMessage (hParent, QQ_WM_LBUTTONDOWN, vdx, lParam);  
PostMessage (hParent, QQ_WM_MOUSEMOVE , vdx, lParam);  
PostMessage (hParent, QQ_WM_ANN_GETCURSEL, vdx, LastSelectionNumber);
```

where lParam is the parameter passed to the window message handler and contains the current mouse pointer.

If the mode requested is one of the Annotation functions, then additional information can be obtained by calling:

```
qq_ViewGetCreatedAnnProperties ()
```

See Also:

```
qq_ViewGetMode (), qq_ViewAnythingSelected ()
```

returns:

Old Mode, -1 for failure

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetMode (  
INT vdx, // View Handle  
INT mode // MODE_SELECT,MODE_MOVE_WINDOW,MODE_MOVE_IMAGE, etc.  
)  
{  
}
```

qq_ViewSetPosition - sets size, Z-order, and position of display window

```
/*$*****  
qq_ViewSetPosition () - sets size, Z-order, and position of display window
```

Description:

Forces the window to be re-sized/moved. Scroll bars are re-calculated, and displayed if set to visible. The border rect includes the display window plus size of scroll bars.

The image is repainted (but not reloaded)

See Also:

SetWindowPos () - Windows SDK

Parameters:

vdx: View handle

hwndInsertAfter: Window to insert current window after, or:

 HWND_BOTTOM

 HWND_TOP

 HWND_TOPMOST

 HWND_NOTOPMOST

x, y: Horizontal and vertical position of UL corner of new window

cx, cy: Width and height of display area

fuFlags: Windows standard positioning flags.

 SWP_DRAWFRAME

 SWP_HIDEWINDOW

 SWP_NOACTIVATE

 SWP_NOMOVE

 SWP_NOSIZE

 SWP_NOREDRAW

 SWP_NOZORDER

 SWP_SHOWWINDOW

returns:

Success (0) or Failure (-1)

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetPosition (
```

```
  INT vdx,          // handle to View Index
```

```
  HWND hwndInsertAfter, // Window to precede the positioned window in Z-order
```

```
  INT x,          // new position of left side of window
```

```
  INT y,          // new position of top of window
```

```
  INT cx,          // width of display area
```

```
  INT cy,          // height of display area
```

```
  INT fuFlags      // window sizing and positioning options
```

```
)
```

```
{
```

```
}
```

qq_ViewSetPostItDefaultProperties - Set the xxx specific properties

```
/*$*****  
qq_ViewSetPostItDefaultProperties() - Set the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetPostItDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_POST lpPost,  
LPSTR lpString  
)  
{  
}
```

qq_ViewSetRectDefaultProperties - Set the xxx specific properties

```
/*$*****  
qq_ViewSetRectDefaultProperties() - Set the xxx specific properties
```

To be filled in...

```
*****$*/  
INT _export FAR PASCAL  
qq_ViewSetRectDefaultProperties(  
INT vdx,  
PPROPERTY_LIST_BASE lpBase,  
PPROPERTY_RECT lpRect  
)  
{  
}
```

qq_ViewSetScrollBars - sets scroll bar visibility flags

```
/*$*****  
qq_ViewSetScrollBars () - sets scroll bar visibility flags
```

Description:

Sets the current visibility flag, and forces the image to be re-painted

Returns:

-1 for failure, 0 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetScrollBars (
```

```
INT vdx, // index to view object
```

```
INT bHzSBVisible, // set visibility for horizontal scroll bar
```

```
INT bVtSBVisible // set visibility for vertical scroll bar
```

```
)
```

```
{
```

```
}
```

qq_ViewSetScrollPos - sets the X/Y offset of the display bitmap

```
/*$*****  
qq_ViewSetScrollPos () - sets the X/Y offset of the display bitmap
```

Description:

Sets the x and y offset of the display bitmap within the current window.

The X and Y values are positive increments if moving the bitmap to the left, or up in the display window.

Returns:

-1 for failure, 0 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetScrollPos (
```

```
INT vdx, // index to view object
```

```
INT Xpos, // x offset (0 for default, positive to move left)
```

```
INT Ypos // y offset (0 for default, positive to move up)
```

```
)  
{  
}
```

qq_ViewSetVisibility - sets visibility flag for image

```
/*$*****  
qq_ViewSetVisibility () - sets visibility flag for image
```

Description:

Sets the current visibility flag and forces the image to be re-painted

Returns:

-1 for failure, 0 otherwise

```
*****$*/
```

```
INT _export FAR PASCAL
```

```
qq_ViewSetVisibility (  
    INT vdx,          // index to view object  
    INT bBorderVisible, // set visibility of border display  
    INT bWndVisible   // set visibility of image  
)  
{  
}
```