iff.library

COLLABORATORS			
	TITLE : iff.library		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 6, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

iff.library

1.1 Iff.Library Documentation

```
TABLE OF CONTENTS

IFFL_CloseIFF()         IFFL_CompressBlock()    IFFL_DecodePic()
IFFL_DecompressBlock()  IFFL_FindChunk()        IFFL_GetBMHD()
IFFL_GetColorTab()      IFFL_GetViewModes()     IFFL_IFFError()
IFFL_ModifyFrame()      IFFL_OpenIFF()          IFFL_PopChunk()
IFFL_PushChunk()        IFFL_SaveBitMap()       IFFL_SaveClip()
IFFL_WriteChunkBytes()  NewOpenIFF()            OpenIFF()
```

1.2 iffl_closeiff()

```
    NAME
  IFFL_CloseIFF -- Close an IFF file and deallocate buffers

    SYNOPSIS
  IFFL_CloseIFF( iff )
           A1

  void IFFL_CloseIFF( IFFL_HANDLE )

    FUNCTION
  Returns the memory previously allocated by IFFL_OpenIFF().

    INPUTS
  iff - IFF file handle, from IFFL_OpenIFF()

    RESULTS
  For now, always results TRUE (this may change in the future).

    SEE ALSO
       IFFOpenIFF()
```

1.3 iffl_compressblock()

```
  NAME
IFFL_CompressBlock -- Compress a memory block

  SYNOPSIS
result = IFFL_CompressBlock( source, destination, size, mode )
             A0      A1     D0    D1

ULONG IFFL_CompressBlock( APTR, APTR, ULONG, ULONG )

  FUNCTION
Compress the memory block using the appropriate compression mode.
If the compressed data would become longer than the uncompressed,
an error is returned.

  INPUTS
source      - Pointer to data to compress
destination - Target address for compression
size        - Number of data bytes to compress
mode        - Compression mode. Currently, the following modes
        are supported:

        IFFL_COMPR_NONE   - Vanilla copy
        IFFL_COMPR_BYTERUN1 - CmpByteRun1 (ILBM BODY data)
        IFFL_COMPR_FIBDELTA - Fibonacci Delta (8SVX BODY data)

  RESULTS
Length of compressed data or 0 if an error occurred.
IFFL_IFFError() returns IFFL_ERROR_BADCOMPRESSION if you ask for
an unsupported compression mode.

 BUGS
In IFFL_COMPR_BYTERUN1, if the compressed data would become longer,
the buffer will be overwritten, and no error is returned. So be
sure to supply a destination buffer which is big enough!

  SEE ALSO
IFFL_DecompressBlock()
```

1.4 iffl_decodepic()

```
  NAME
IFFL_DecodePic -- decode the BODY of an ILBM file into a BitMap

  SYNOPSIS
success = IFFL_DecodePic( iff, bitmap )
 D0       A1   A0

BOOL IFFL_DecodePic( IFFL_HANDLE, struct BitMap * )

  FUNCTION
Decodes and decompresses a picture into the user supplied BitMap.
If the picture is higher than your BitMap, it will be truncated.
If your BitMap is larger than the picture, the picture will be
drawn into the top left corner of your BitMap.
```

```
If the picture has less planes than the BitMap, the unused planes
are NOT touched by this routine, so you may wish to clear them
before calling IFFL_DecodePic(). If the picture has more planes
than your BitMap, the surplus planes of the picture are ignored.

  INPUTS
iff    - IFF file handle, from IFFL_OpenIFF()
bitmap - Pointer to a properly initialized BitMap structure:
   bm_Planes[] must point to valid BitPlanes,
   bm_Depth contains the number of planes.
   bm_Width and bm_Height must be set according to the
   size of your bit planes.

  RESULTS
Non-zero if successful, zero if error. Call IFFL_IFFError() to
know the reason of the failure

  BUGS
If the picture is wider than your BitMap, one line of innocent
memory will be overwritten at the end of each bitplane. You can
avoid this by allocating a few bytes more for each plane.
Normally, you allocate your BitMap after inspecting the BMHD
chunk, so this should not be a problem.

  NOTE
This routine needs at least 650 bytes of stack space

  SEE ALSO
```

1.5 iffl_decompressblock()

```
  NAME
IFFL_DecompressBlock -- Decompress a memory block

  SYNOPSIS
result = IFFL_DecompressBlock( source, destination, size, mode )
             A0      A1     D0    D1

ULONG IFFL_DecompressBlock( APTR, APTR, ULONG, ULONG )

  FUNCTION
Decompress the memory block using the appropriate Decompression mode.
If the Decompressed data would become longer than the unDecompressed,
an error is returned.

  INPUTS
source      - Pointer to data to decompress
destination - Target address for decompression
size      - Number of _DECOMPRESSED_ data bytes
mode      - Compression mode. Currently, the following modes
        are supported:

        IFFL_COMPR_NONE   - Vanilla copy
        IFFL_COMPR_BYTERUN1 - CmpByteRun1 (ILBM BODY data)
        IFFL_COMPR_FIBDELTA - Fibonacci Delta (8SVX BODY data)
```

```
  RESULTS
Length of uncompressed data or 0 if an error occurred.
IFFL_IFFError() returns IFFL_ERROR_BADCOMPRESSION if you ask for
an unsupported compression mode.

  SEE ALSO
IFFL_CompressBlock()
```

1.6 iffl_findchunk()

```
 NAME
IFFL_FindChunk -- find an IFF-chunk

 SYNOPSIS
chunk = IFFL_FindChunk( iff, chunkname )
D0      A1   D0

APTR IFFL_FindChunk( IFFL_HANDLE, ULONG )

 FUNCTION
Find a specific chunk in an IFF file

 INPUTS
iff   - IFF file handle, from IFFL_OpenIFF()
chunkname - 4 characters packed ASCII ('BODY', 'VHDR' ...)
      if chunkname is 0, FindChunk() returns a pointer to the
      first byte after the end of the current FORM. This can
      be used by ANIM readers for jumping from FORM to FORM.

 RESULTS
Pointer to the beginning of the chunk (that means to the chunk
name itself, followed by the chunk size); zero if chunk not found

 BUGS
none known

 SEE ALSO
IFFL_GetBMHD() IFFL_GetColorTab()
```

1.7 iffl_getbmhd()

```
  NAME
IFFL_GetBMHD -- find a BitMapHeader of an IFF-file

  SYNOPSIS
header = IFFL_GetBMHD( iff )
D0           A1

struct BitMapHeader *IFFL_GetBMHD( IFFL_HANDLE )

  FUNCTION
```

```
Returns a pointer to a BMHD (BitMapHeader) structure as defined
in iff.h and iff.i

  INPUTS
iff - IFF file handle, from IFFL_OpenIFF()

  RESULTS
Pointer to the BitMapHeader, or NULL if no BMHD chunk found

  SEE ALSO
IFFL_FindChunk()
IFFL_GetColorTab()
```

1.8 iffl_getcolortab()

```
  NAME
IFFL_GetColorTab -- find a CMAP and convert it to a ColorTable

  SYNOPSIS
count = IFFL_GetColorTab( iff, colortable )
D0        A1   A0

LONG IFFFL_GetColorTab( IFF_HANDLE, UWORD * )

  FUNCTION
Searches the CMAP chunk of an IFF file and converts it, if it's
there, to an Amiga color table structure. This colortable can
directly be used as a parameter for the LoadRGB4() function.

  INPUTS
iff    - IFF file handle, from IFFL_OpenIFF()
colortable - Pointer to a block of memory which must be large
       enough to hold the colortable (2 bytes per color).
       Must be WORD aligned.

  RESULT
Number of colors actually found, or zero if the file has no
CMAP chunk

  SEE ALSO
IFFL_FindChunk()
IFFL_GetBMHD()
```

1.9 iffl_getviewmodes()

```
  NAME
IFFL_GetViewModes() -- Get Amiga-specific ViewModes

  SYNOPSIS
viewmodes = IFFL_GetViewModes( iff )
D0             A1
```

```
ULONG IFFL_GetViewModes( IFFL_HANDLE )

  FUNCTION
Searches the IFF file for a 'CAMG' chunk which holds the view modes
information. If there is no CAMG chunk, the view modes are calcu-
lated using the information in the BitMapHeader structure.
You can directly put the low WORD of the result of a call to
IFFL_ GetViewModes() into the ns_ViewModes field of a NewScreen
structure, or you can use the whole ULONG for the SA_DisplayID tag
under OS 2.x.

  INPUTS
iff - IFF file handle, from OpenIFF()

  RESULT
viewmodes - ULONG containing the view modes (HAM, LACE, HIRES ...)
All forbidden bits are masked out, as suggested by CBM.
Under Kickstart V1.3, only the lower WORD is used.

  BUGS
If the IFF file has no CAMG chunk and 6 bitplanes, the HAM bit
will be set. This is not always correct since the picture could
be in the Extra Halfbrite mode. You can load such Halfbrite
pictures into DPaint III and save them again, DPaint will generate
the correct CAMG chunk.

  SEE ALSO
<graphics/displayinfo.h>
```

1.10 iffl_ifferror()

```
  NAME
IFFL_IFFError -- Get detailed error descrpition after an error

  SYNOPSIS
error = IFFL_IFFError()
D0

LONG IFFL_IFFError( VOID )

  FUNCTION
If one of the iff.library functions returns zero, you can call
IFFL_IFFError() to know the reason for the failure. An error
code is returned, please refer to the files 'iff.h' or 'iff.i'
for the complete list of errors.

  INPUTS
none

  RESULT
Error-number generated by the latest function call, or zero if
no error.

  BUGS
If you don't close the IFF library at the end of your program
```

```
(using CloseLibrary()) the error node will not be freed. The
same task will then not be able to re-open the iff.library.
(This is not a bug, it's a feature ;-))

  SEE ALSO
<iff.h>
```

1.11 iffl_modifyframe()

```
  NAME
IFFL_ModifyFrame -- Modify an anim frame using a DLTA chunk

  SYNOPSIS
success = IFFL_ModifyFrame( modifyform, bitmap )
D0          A1    A0

BOOL IFFL_ModifyFrame( VOID *, struct BitMap * )

  FUNCTION
Uses the DLTA chunk of the supplied FORM to modify the planes-data
of the bitmap. Usually, playback of ANIMs will require two buffers,
and double-buffering between them. So the data in the bitmap must
be two frames back, and the DLTA chunk is used to modify the hidden
frame to the next frame to be shown.

  INPUTS
modifyform - pointer to the FORM containing the actual DLTA chunk
bitmap     - Pointer to a properly initialized BitMap structure,
       the planes must contain the image which was displayed
       to frames back (using double-buffering)

  RESULT
Non-zero if OK, 0 if error; call IFFL_IFFError() to know the reason
of the failure

  RESTRICTIONS
Currently, only compression type 5 (Byte Vertical Delta Mode) is
implemented. If you have animations which use modes 1 to 4, try
loading them with DPaint III and saving them again.
Sculpt-Animate ('J' type ANIM, Movie format) support will be
added soon.
I will implement some more compression types upon request.

  NOTE
This routine needs at least 820 bytes of stack.
The size of the bitmap is not checked by this routine, the planes
must have at least the size described in the BMHD of the anim
file.

  SEE ALSO
IFFL_IFFError()
```

1.12 iffl_openiff()

```
 NAME
IFFL_OpenIFF -- Open an IFF file for reading or writing

 SYNOPSIS
iff = IFFL_OpenIFF( filename, mode )
D0        A0        D0

IFFL_HANDLE IFFL_OpenIFF( char *, ULONG )

 FUNCTION
If mode == IFFL_MODE_READ:
  This function opens a file on a disk and looks whether it's an IFF
  file or not. If it is an IFF file, memory is allocated and the file
  is read into memory.

 New for V22:
 If xpkmaster.library is installed in your system, IFFL_OpenIFF()
 will be able to read and decompress compressed IFF files, if they
 use one of the xpk standard compression schemes.

If mode == IFFL_MODE_WRITE:
  Initializes an IFF file handle for writing. You may create chunks
  with IFFL_PushChunk() and IFFL_PopChunk(), and you can write data
  using the IFFL_WriteChunkBytes() routine.

 INPUTS
filename - Pointer to a null-terminated string
mode   - IFFL_MODE_READ:  Open file for reading
     IFFL_MODE_WRITE: Open file for writing

 RESULT
iff - IFF handle. Making assumptions about the internal structure
      of this handle is unwise, and may break in the future.
      If this function fails, NULL will be returned, and you may
      call IFFL_IFFError() to know the reason of the failure.

 SEE ALSO
IFFL_CloseIFF()
IFFL_PushChunk()
IFFL_PopChunk()
IFFL_WriteChunkBytes()
IFFL_IFFError()
 BUGS
None known
```

1.13 iffl_popchunk()

```
  NAME
IFFL_PopChunk -- Pop top context node off context stack.

  SYNOPSIS
success = IFFL_PopChunk( iff )
```

```
D0       A0

BOOL IFFL_PopChunk( IFFL_HANDLE )

  FUNCTION
Pops top context chunk and updates the file accordingly.
The function is normally called only for writing files and signals
the end of a chunk.

  INPUTS
iff - IFF handle

  RESULTS
Non-zero  if successful or 0 if not successful (call IFFL_IFFError()
to get an IFFL_ERROR_... error code.

  SEE ALSO
IFFL_PushChunk()
IFFL_WriteChunkBytes()
```

1.14 iffl_pushchunk()

```
  NAME
IFFL_PushChunk -- Push a new context node on the context stack.

  SYNOPSIS
success = IFFL_PushChunk( iff, type, id )
D0        A0   D0    D1

BOOL IFFL_PushChunk( IFFL_HANDLE, ULONG, ULONG )

  FUNCTION
Pushes a new context node on the context stack by reading it from the
stream if this is a read file, or by creating it from the passed
parameters if this is a write file.  Normally this function is only
called in write mode, where the type and id codes specify the new
chunk to create.  If this is a leaf chunk, i.e. a local chunk inside
a FORM or PROP chunk, then the type argument is ignored.

  INPUTS
iff  - IFF handle
type - chunk type specifier (ex. ILBM) (ignored for read mode or
       leaf chunks).
id   - chunk id specifier (ex. CMAP) (ignored for read mode).

  RESULTS
Non-zero if successful or 0 if not successful (call IFFL_IFFError()
to get an IFFL_ERROR_... error code.

  NOTE
     Currently, the level of nested FORMs is restricted to 7.

  SEE ALSO
IFFL_PopChunk()
```

```
IFFL_WriteChunkBytes()
```

1.15 iffl_savebitmap()

```
  NAME
IFFL_SaveBitMap -- save the planes of a BitMap as an IFF-file

  SYNOPSIS
result = IFFL_SaveBitMap( filename, bitmap, colortable, flags )
D0        A0      A1      A2    D0

BOOL IFFL_SaveBitMap(char *, struct BitMap *, UWORD *, ULONG )

  FUNCTION
Save the planes of a BitMap as an IFF-file, optionally with a
colortable. The IFF file will contain the following chunks:

FORM  -  The IFF header, with the type ILBM
BMHD  -  The BitMap Header structre
CMAP  -  The color map, this chunk is omitted if colortable is zero
CAMG  -  The Amiga ViewModes word, contains the special ViewModes
   information (HAM, LACE, HIRES ...)
BODY  -  The (crunched or uncompressed) picture

 INPUTS
filename    - Name of the IFF file to create
bitmap      - Pointer to the BitMap holding your picture
colortable  - Pointer to an Amiga ColorTable structure or zero
        (if colortable = 0, no CMAP chunk will be generated).
flags     - Bit 0: 1 = Use the "CmpByteRun1" compression algorythm,
         0 = Save the file uncompressed
        Bit 7: 1 = This is a HAM (Hold and modify) picture
         0 = This is a normal or Extra-Halfbrite picture

 RESULT
Non-zero if successful, 0 if an error occurred. You can then call
IFFL_IFFError() to know more about the reason of the failure.

 NOTE
Up to V19 this routine needs at least 650 bytes of stack space

 SEE ALSO
IFFL_SaveClip()
```

1.16 iffl_saveclip()

```
  NAME
IFFL_SaveClip -- save a part of a BitMap as an IFF-file

  SYNOPSIS
result = IFFL_SaveClip
D0   ( filename, bitmap, coltab, flags, xoff, yoff, width, height )
```

```
     A0      A1      A2      D0     D1    D2  D3     D4

  FUNCTION
Save a part of a BitMap as an IFF file

  INPUTS
filename    - Name of the IFF file to create
bitmap      - Pointer to the BitMap holding your picture
colortable  - Pointer to an Amiga ColorTable structure or zero
        (if colortable = 0, no CMAP chunk will be generated).
flags     - Bit 0: 1 = Use the "CmpByteRun1" compression algorythm,
         0 = Save the file uncompressed
       Bit 7: 1 = This is a HAM (Hold and modify) picture
         0 = This is a normal or Extra-Halfbrite picture
xoff      - X offset of the clip to be saved (bytes from the left)
yoff      - Y offset of the clip to be saved (lines from the top)
width     - width in bytes of the rectangle
height      - height in lines of the rectangle

  RESULTS
Non-zero if successful, 0 if an error occurred. You can then call
IFFL_IFFError() to know more about the reason of the failure.

  NOTE
Up to V19 this routine needs at least 650 bytes of stack space

  BUGS
The width of the rectangle will be truncated to WORD boundaries,
because DPAINT wants it!

  SEE ALSO
IFFL_SaveBitMap()
```

1.17 iffl_writechunkbytes()

```
 NAME
IFFL_WriteChunkBytes -- Write data into the current chunk

 SYNOPSIS
success = IFFL_WriteChunkBytes( iff, buf, size )
D0        A0   A1   D0

LONG IFFL_WriteChunkBytes( IFFL_HANDLE, APTR, LONG )

 FUNCTION
Writes "size" bytes from the specified buffer into the current chunk.

 INPUTS
iff   - IFF file handle, from IFFL_OpenIFF().
buf   - pointer to buffer area with bytes to be written.
size  - number of bytes to write.

 RESULT
Non-NULL if the write was successful, or NULL if an error
occurred. Call IFFL_IFFError() to know what's going on.
```

```
 SEE ALSO
IFFL_PushChunk()
IFFL_PopChunk()
IFFL_IFFError()
```

1.18 newopeniff()

```
 NAME
NewOpenIFF -- allocate memory for an IFF-file and read it

 SYNOPSIS
ifffile = NewOpenIFF( filename, memattr )
D0          A0  D0

IFFFILE OpenIFF( char * )

 FUNCTION
THIS FUNCTION IS OBSOLETE. USE IFFL_OpenIFF() INSTEAD.

 INPUTS
filename - Pointer to a null-terminated string
memattr  - Memory requirements as used for Exec's AllocMem(),
     such as MEMF_CHIP, MEMF_PUBLIC ...
     (MEMF_CLEAR is not necessary)

 RESULT
ifffile - 'FileHandle', points to the beginning of the IFF file
("FORM...."), Zero if unsuccessful. Call IFFError() to get
the reason of the failure.

 SEE ALSO
IFFL_OpenIFF()
IFFL_CloseIFF()
IFFL_IFFError()

 BUGS
None known
```

1.19 openiff()

```
  NAME
OpenIFF -- allocate memory for an IFF-file and read it

  SYNOPSIS
ifffile = OpenIFF( filename )
D0       A0

IFFFILE OpenIFF( char * )

  FUNCTION
THIS FUNCTION IS OBSOLETE. USE IFFL_OpenIFF() INSTEAD.
```

```
 INPUTS
filename - Pointer to a null-terminated string

 RESULT
ifffile - 'FileHandle', points to the beginning of the IFF file
("FORM...."), 0 if unsuccessful. Call IFFError() to get the
reason of the failure.

 BUGS
None

 SEE ALSO
IFFL_OpenIFF()
IFFL_CloseIFF()
IFFL_IFFError()
```