

ilbm

Michael Zucchi

Copyright © 1994 Michael Zucchi, All right reserved

COLLABORATORS

	<i>TITLE :</i> ilbm		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Michael Zucchi	August 30, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ilbm	1
1.1	ilbm.guide	1
1.2	Module overview	1
1.3	The guy who wrote it	2
1.4	ilbm_new	3
1.5	ilbm_dispose	3
1.6	ilbm_pictureinfo	4
1.7	ilbm_loadpicture	5
1.8	ilbm_freebitmap	6
1.9	Information about the examples	6

Chapter 1

ilbm

1.1 ilbm.guide

IFF ILBM picture loading module for AmigaE2.5+

© 1993 Michael Zucchi
All Rights Reserved

This document describes the `ilbm.m` module, which includes commands for loading and displaying (if possible) IFF `ilbm` pictures. The functions provided are intended to be as easy to use as possible, while providing a lot of flexibility.

The following sections are available:

`OverView` some of the ideas behind the module

Module functions

`ilbm_New()` to open a picture
`ilbm_Dispose()` to cleanup

`ilbm_PictureInfo()` to get picture size/palette etc
`ilbm_LoadPicture()` loads data into a bitmap/screen

`ilbm_FreeBitMap()` to free a bitmap allocated by `LoadPicture`

Examples

NOTE: This module requires Workbench 2.0 (V36) or higher! Please make sure that this version of the system libraries is present before using these functions.

NOTE!!! Due to a small oversight, only COMPRESSED `ilbm`'s currently work. This will be fixed soon ... i hope? (uncompressed `ilbm`'s are uncommon anyway)

1.2 Module overview

Not much to say really - this module is just for loading/saving IFF ILBM's!

Designed to be used to easily load ilbm's for picture screens, or into bitmaps for later blitting, or anywhere else where an ilbm would be useful.

One thing - this module will work on V36 systems, however, on V39+ systems, new graphics.library functions are used wherever possible. e.g. LoadRGB32() for full 24-bit palettes on AGA+ machines.

future plans

As it stands, the module is ideal for loading pictures for displaying. Another idea that may be implemented is a 'chunky' mode loading function (e.g. ILBML_CHUNKY) whereby the data is converted to byte-per-pixel format before being output to a byte array.

A save function would also be useful - i havent implemented it yet because of lack of time, and also to keep the module small.

Actually ...

This sort of thing should be handled much better using datatypes. Unfortunately, they're a bit of a pain to use at the moment - and very inefficient. I see this module being primarily used for loading ILBM's for graphics for games/applications, rather than for writing picture viewers.

1.3 The guy who wrote it

The iff unpacker i wrote a long time ago for zgif, its reasonably fast, but it doesn't go all out for speed!

Presently, i study 'from time to time' (:-) in order to obtain a Computer Systems Engineering degree from the Univerity Of South Australia. (1994=final year ↔)

I'm also currently 'Zed' of FRONTIER in my anti-os hours.

I can be contacted in the following ways:

Internet email:

9107047w@lux.levels.unisa.edu.au
till the end of '94 at least - reliable

'Real Mode' (tm) mail:

Michael Zucchi
PO BOX 824
Waikerie
South Australia 5330
slow, but very reliable - till mum sells the house :)

Michael Zucchi
110 Dunrobin Rd
Warradale
South Australia 5046
to my door - till i move (?)

1.4 ilbm_new

ilbm.m/ilbm_New

ilbm.m/ilbm_New

SYNTAX

```
ilbmhandle := ilbm_New( name:PTR TO CHAR,  
                      flags:LONG );
```

PURPOSE

Creates a (private!) ilbm handle structure, and fills in several fields. The file specified by 'name' is opened, and the IFF ILBM chunks BMHD, CAMG, and CMAP are parsed.

INPUTS

name A NULL terminated string, specifying the name of the picture. This MUST be present!

flags A mask of options, current flags are:

ILBMF_COLOURS4

create a LoadRGB4() compatible version of the palette, and store a pointer to it in the picture info block as 'pal4' (see ilbm_PictureInfo())

ILBMF_COLOURS32

create a LoadRGB32() compatible version of the palette, and store a pointer to it in the picture info block as 'pal32' (see ilbm_PictureInfo())

OUTPUTS

ilbmhandle A !!PRIVATE!! handle used with the other ilbm_xxx functions. If for some reason something didn't work, it will be 0.

NOTES

SEE ALSO

ilbm_Dispose(), ilbm_LoadPicture(), ilbm_PictureInfo()

1.5 ilbm_dispose

ilbm.m/ilbm_Dispose

ilbm.m/ilbm_Dispose

SYNTAX

```
ilbm_Dispose( iffhandle:LONG );
```

PURPOSE

Closes the original file, free's the iffparse.library stuff, and closes some libraries. Use this to free unneeded data once the picture has been loaded.

INPUTS

iffhandle An iffhandle obtained from ilbm_New(), or 0.

OUTPUTS

NOTES

It is safe to pass `iffhandle:=0` to this function.

SEE ALSO

`ilbm_New()`

1.6 ilbm_pictureinfo

`ilbm.m/ilbm_PictureInfo`

`ilbm.m/ilbm_PictureInfo`

SYNTAX

```
pictureinfo := ilbm_PictureInfo( iffhandle:LONG )
```

PURPOSE

Returns a pointer to a `pictureinfo` object which contains information about the picture being loaded.

INPUTS

`iffhandle` A VALID `iffhandle` obtained from `ilbm_New()`.

OUTPUTS

`pictureinfo` A pointer to an object of type `'pictureinfo'`.
The fields will be set-up as following:

`bmhd` pointer to the `BitMapHeader` from the IFF file
`modeid` the `modeid`, as obtained from the `CAMG` chunk - or 0.
 This may also be set by the application before calling `ilbm_LoadPicture()`
`colours` number of colours represented in the picture. An IFF-24 picture will have 16,777,216 stored here!
`palraw` If the number of colours (above) is 256 or less, and a `CMAP` chunk was present, `palraw` is a pointer to the raw 24-bit palette read from the IFF file. The colours are stores as groups of 3 bytes - Red/Green/Blue
`pal4` If `ILBMF_COLOURS4` was specified when the `iffhandle` was created, and there was a `CMAP` present, AND there was enough memory, `pal4` is a pointer to a `LoadRGB4()` compatible array of colours - `'colours'` of them.
`pal32` If `ILBMF_COLOURS32` was specified when the `iffhandle` was created, and there was a `CMAP` present, AND there was enough memory, `pal32` is a pointer to a `LoadRGB32()` compatible array of colours - `'colours'` of them.

NOTES

The `modeid` field is the only one writeable! All others are read-only.

If memory is tight, the `pal4` and `pal32` fields may still be zero, even if they were requested originally. It would be a good idea always to check these fields before use.

SEE ALSO

`ilbm_New()`, `ilbm_Dispose()`, `ilbm_LoadPicture()`

1.7 ilbm_loadpicture

ilbm.m/ilbm_LoadPicture

ilbm.m/ilbm_LoadPicture

SYNTAX

```
status := ilbm_LoadPicture ( iffhandle:LONG,
                             taglist:LONG )
```

PURPOSE

Loads the picture into the specified environment.

INPUTS

`iffhandle` An iffhandle obtained using `ilbm_New()`, or 0 in which case an error will be returned

`tags` A tag-list specifying the loading options. Currently defined tags are:

`ILBML_BITMAP` `tag.data` points to an existing bitmap in which to load the picture data. The bitmap needs to be big enough ...

`ILBML_SCREEN` `tag.data` points to an existing screen in which to load the picture/palette.

`ILBML_CHUNKY` `tag.data` specifies a byte array to store a chunky-pixel version of the picture
NOT IMPLEMENTED

`ILBML_GETBITMAP` This specifies that `ilbm_LoadPicture()` will allocate its own bitmap. In this case, `tag.data` is a pointer to a variable, which will hold the obtained bitmap.
`ilbm_FreeBitMap()` MUST be used to free this bitmap.

`ILBML_GETSCREEN` Specifies that `ilbm_LoadPicture()` will open the screen for you. `tag.data` points to a variable that will hold the screen pointer once obtained. If the screen could not open, zero is stored that variable. The screen must be closed by a `CloseScreen()` call - this can be after `ilbm_Dispose()` is called.

`ILBML_GETCHUNKY` guess! NOT IMPLEMENTED

`ILBML_SCREENTAGS` If `ILBML_GETSCREEN` was used, then this tag can be used to specify additional tags to be used when opening the screen. The following tags must NOT be used: `SA_WIDTH`, `SA_HEIGHT`, `SA_DEPTH`, `SA_DISPLAYID`.

`ILBML_NOCOLOUR` If `SA_SCREEN/SA_GETSCREEN` have been specified, then using this `BOOL` tag will prevent `ilbm_LoadPicture()` from setting the palette for the screen. Only specify if it is to be true.

OUTPUTS

status =0 if all went OK, or negative for errors (see ilbmdefs.m)

NOTES

Remember, if one of the 'GET' tags is used, it is up to the application to free whatever was got.

SEE ALSO

ilbm_New(), ilbm_Dispose(), ilbm_PictureInfo(), ilbm_FreeBitMap()

1.8 ilbm_freebitmap

ilbm.m/ilbm_FreeBitMap

ilbm.m/ilbm_FreeBitMap

SYNTAX

```
ilbm_FreeBitMap( bitmap )
```

PURPOSE

Free's a bitmap returned by ilbm_LoadPicture(), via the ILBML_GETBITMAP tag.

INPUTS

bitmap a VALID bitmap, as returned by the ILBML_GETBITMAP tag.

OUTPUTS

NOTES

If V39 is present, this function just calls FreeBitMap() - otherwise, it uses its own custom routines.

SEE ALSO

ilbm_LoadPicture()

1.9 Information about the examples

This section describes the source-form examples so far provided.

showpic

A simple picture-viewer. It demonstrates an easy way to load and display a picture in an Amiga Screen. The use of the asl.library's file requester is also demonstrated.

usage:

```
showpic
```

picwindow

Another simple picture-viewer. This one displays the picture on the workbench screen, in a suitably sized window. It demonstrates loading into bitmaps, obtaining information about the picture

before it is displayed, and blitting into workbench windows.

usage:

picwindow
