Implementing the iff.library

By Michael Bodin

Editor's Note: Welcome to the first installment of our monthly column on Amiga programming, which will cover C, BASIC, Modula-2, and ARexx. Each month, a different guest columnist will share advice and provide techniques for his language of choice.

WHAT'S A PROGRAMMER to do? IFF (Interchange File Format) is a wonderful file standard, but with no standard way of implementing it.

To the rescue comes Chris Weber from Zurich, Switzerland and his iff.library, the first attempt at a standard for reading and writing IFF files. The library provides a set of very useful tools that allow you to develop IFF compatible programs without re-inventing the wheel each time you write IFF support code. Although these routines are geared mainly for graphic IFF files, they are flexible and can be used for Anim (animations), 8SVX (sound), or SMUS (music) files. The functions available are:

- ifffile = OpenIFF(filename)—Allocates any available memory for an IFF file and then loads it in.

- ifffile = NewOpenIFF(filename, memattr)—Allocates specific memory for the IFF file and then loads it in.

- CloseIFF(ifffile)—Closes an IFF file and deallocates the memory created by OpenIFF( ).

- header = GetBMHD(ifffile)—Finds the BitMapHeader chunk within an ILBM image file.

- count = GetColorTab(ifffile, colortable)—Finds the CMAP chunk and converts it to a ColorTable.

- result = DecodePic(ifffile,bitmap)—Decodes the BODY of an ILBM file into a bitmap structure.

- result = SaveBitMap(filename,bitmap, colortable,flags)—Saves the contents of the BitMap as an IFF-ILBM file.

- result = SaveClip(filename,bitmap,color table,flags,xoff,yoff,width,height)—Saves a part of the overall BitMap as an IFF-ILBM file.

- viewmodes = GetViewModes(ifffile)—Gets the ViewModes word.

- chunk = FindChunk(ifffile,chunkname)—Finds the first occurrence of the specified chunk id. (ILBM, 8SVX, etc.)

- error = IffError( )—Returns a more detailed error code after a library function has failed.

Before you can start coding, copy the iff.library to the libs: directory of your Workbench disk, and copy the iff.h include file to a backup of your include-files disk. Iff.h must be called in your source code to allow you to use the library definitions. If you use Lattice C, you will find a pragma file that comes with the library's zoo file. Place these pragma definitions into your source code along with the include statements for the iff.h header file. Be aware that SaveClip( ) has eight parameters and Lattice's pragma definitions can handle only six. If you try to use SaveClip( ) you will get a linker error. If you are not using Lattice, you must assemble the library stub routines included in the zoo file, then link their object code along with other standard library modules. You may use the SaveClip( ) routine only if you assemble the library stub routines.

Where to find iff.library

Network	Area	Location	Title
CompuServe	AmigaTech Forum	LIB #1	IFFLIB.ZOO
PeopleLink	Amiga Zone	Sec. #11	IFF_LIB_LAT.ARC
GEnie	Amiga Starship	Sec. #13	SHOWIFF2.ZOO
Lattice BBS	Amiga	Amiga Files	IFLIB161.ZOO

IFF IN ACTION

Listing 1, QuikShow.c, is a compact IFF file viewer that demonstrates the power of these functions. It was written with Lattice C 5.02. To run it from the CLI, type QuikShow <filename>. The program opens an IFF picture file (filename), displays it, and exits.

Let's look closer at QuikShow. Notice that the IFFBase is declared as a pointer to a Library struct. This is the one declaration you must make that is not in the iff.h header file. The program starts at main( ) by checking the input. If the user has typed a question mark or used improper syntax, the program prompts the user with the proper syntax for the command, then exits.

If a filename is given, the program opens the Graphics, Intuition, and IFF libraries. The program then calls the OpenIFF( ) function from the IFF library. This function opens the file named on the command line, checks if its file type is IFF, allocates memory for the file, and loads it in. Note that the memory used might be chip or fast RAM. When using the New- ►

OpenIFF( ) function, you can specify the exact type of RAM.

Once the file has been successfully opened, the program checks the file for a BitMapHeader chunk. If QuikShow finds the BMHD chunk, it sets up the screen structure according to the header definition. The screen's width, height, and depth are all set from the incoming picture file. Next, the GetViewModes( ) function gets the ViewMode words (HIRES, HAM, etc.) and enters them into the screen structure. Finally, the program opens the viewing screen.

With the screen ready, the program needs the rest of the picture file. GetColorTab( ) gets the color register definitions from the file and converts them to a ColorTable structure to be used with the LoadRGB4( ) graphics library function. This sets the screen's colors to those defined in the IFF picture file. QuikShow next calls the DecodePic( ) function. With this one command you decompress the file and then display it in the view screen's bitmap. The Delay( ) function inserts a pause before the program closes down the IFF file, the screen, and the libraries.

SOUND STATEMENTS

Although the example deals with graphics, loading a digitized sound file is equally quick. After you open the file using OpenIFF( ) or NewOpenIFF( ), use FindChunk( ) to locate the proper chunks for an 8SVX file, the VHDR (VoiceHeader) and BODY chunks. Then set your program's attributes according to the file parameters.

You can find Chris' freely distributable iff.library file on most of the major networks (see table). The current version of the file is 16.1 and dated 01-DEC-88. The library itself consumes a mere 2352 bytes, and its routines provide a much needed step towards a standard. The iff.library is well worth the few minutes you will spend downloading it. □

Mike Bodin is System Manager for Florida's State Division of Elections. Write to him c/o AmigaWorld, *Editorial Dept., 80 Elm St., Peterborough, NH 03458.*

Listing 1. QuikShow.c—an example program using the iff.library routines.

```
/* QuikShow.c - based on ShowIFF.c by Christian
*  A. Weber. Feel free to use or modify it. */

#include <exec/types.h>
#include <graphics/gfxbase.h>
#include <intuition/intuition.h>
#include "iff.h"

/* for Lattice Compiler */

#include "IFFpragm.h"

struct Library *IFFBase = NULL, *OpenLibrary();
struct GfxBase *GfxBase = NULL;
struct IntuitionBase *IntuitionBase = NULL ;
struct Screen *myscreen = NULL, *OpenScreen() ;
APTR ifffile = NULL ;

struct NewScreen ns =
{
  0,0,0,0,0,0,0, NULL, CUSTOMSCREEN|SCREENQUIET,
  NULL, "QuikShow", NULL, NULL
};

main(argc,argv)
int argc;
char **argv;
{
  long count = 0;
  UWORD colortable[128];
  struct BitMapHeader *bmhd;

  if((argc != 2) || !strcmp(argv[1],"?"))
  {
      printf("Format: %s filename\n",argv[0]);
      exit(20);
  }

  GfxBase = (struct GfxBase *)
              OpenLibrary("graphics.library",0L);

  IntuitionBase = (struct IntuitionBase *)
              OpenLibrary("intuition.library",0L);

  if(!(IFFBase = OpenLibrary(IFFNAME,IFFVERSION)))
    {
      printf("Copy the iff.library to your LIBS:");
      printf(" directory!\n");
      exit(10);
    }

  printf("Loading file %s ... ",argv[1]);

  if(!(ifffile=OpenIFF(argv[1])))
     Fail("Error opening file");
  if(!(bmhd=GetBMHD(ifffile)))
     Fail("BitMapHeader not found");

  ns.Width      = bmhd->w;
  ns.Height     = bmhd->h;
  ns.Depth      = bmhd->nPlanes;
  ns.ViewModes  = GetViewModes(ifffile);

  if(!(myscreen = OpenScreen(&ns)))
      Fail("Can't open screen!");

  count = GetColorTab(ifffile,colortable);

 /* Limited to 32 color registers */
  if(count>32L) count = 32L;

  LoadRGB4(&(myscreen->ViewPort),colortable,
     count);

  if(!DecodePic(ifffile,&myscreen->BitMap))
     Fail("Can't decode picture");

  Delay(250L);

  Fail("done"); /* Close the whole stuff */
}


Fail(text)
char *text;
{
  if(ifffile) CloseIFF(ifffile);
  if(myscreen) CloseScreen(myscreen);

  printf("%s\n",text);
  printf("IffError = %ld\n",IffError());

  if(IFFBase) CloseLibrary(IFFBase);
  /* MUST be closed when done! */
  CloseLibrary(IntuitionBase);
  CloseLibrary(GfxBase);
  exit(0);
}
```