IffParser

COLLABORATORS			
	TITLE : IffParser		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 14, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 IffParser **1**

1.1 IFFParser Guide . . . 1
1.2 SAS/C and StormC Notes . . . 2
1.3 IFFParser C-Only Functions . . . 3
1.4 Introduction . . . 3
1.5 Author's Info . . . 3
1.6 Amiga Foundation Classes Module: IFFParser / Error Table . . . 4
1.7 IFFParser / History . . . 4
1.8 Amiga Foundation Classes Module: IFFParser / iffparser() . . . 5
1.9 Amiga Foundation Classes Module: IFFParser / close() . . . 5
1.10 Amiga Foundation Classes Module: IFFParser / save(filename) . . . 6
1.11 Amiga Foundation Classes Module: IFFParser / createchunk(type, id, size) . . . 6
1.12 Amiga Foundation Classes Module: IFFParser / writechunk(datas, datalen) . . . 7
1.13 Amiga Foundation Classes Module: IFFParser / closechunk() . . . 7
1.14 Amiga Foundation Classes Module: IFFParser / load(filename) . . . 7
1.15 Amiga Foundation Classes Module: IFFParser / setscan(type, id) . . . 8
1.16 Amiga Foundation Classes Module: IFFParser / first(type, id) . . . 8
1.17 Amiga Foundation Classes Module: IFFParser / addr() . . . 9
1.18 Amiga Foundation Classes Module: IFFParser / size() . . . 9
1.19 Amiga Foundation Classes Module: IFFParser / succ() . . . 9
1.20 Amiga Foundation Classes Module: IFFParser / scan() . . . 10
1.21 Amiga Foundation Classes Module: IFFParser / getheader(string, filename) . . . 10
1.22 Amiga Foundation Classes Module: IFFParser / exit(type, id) . . . 10
1.23 Amiga Foundation Classes Module: IFFParser / version() . . . 11
1.24 Amiga Foundation Classes Module: IFFParser / error() . . . 11
1.25 Amiga Foundation Classes: IFFParser/empty() . . . 12

Chapter 1

IffParser

1.1 IFFParser Guide

```
                                ** IFFParser - Written By Fabio Rotondo **

                    * Part of the Amiga Foundation Classes *



                Introduction

                History

                Author's Info
                  Amiga Foundation Classes


                SAS/C and StormC Notes

                C-Only Functions
                   Requires:  resourceTracker

Class Base: $00080000

COMMANDS                                BRIEF DESCRIPTION
------------------------------------------------------------------------------ -

                iffparser(resTracker=NIL)
                   Inits the iffparser OBJECT.

                addr()
                   Returns current data position in mem.

                close()
                   Closes currently IFF session.

                closechunk()
                   Closes last created chunk.

                createchunk(type, id, size)
                   Inits a new chunk to write.
```

```
empty()
   Checks if the class is empty.

error()
   Returns last error raised.

exit(type, id)
   Defines when scan() must stop.

first(type, id)
   Prepares data examination.

getheader(string, filename)
   Returns header of a file.

load(filename)
   Inits an IFF Load session.

save(filename)
   Inits an IFF SAVE session.

scan()
   Begins scanning an IFF file.

setscan(type, id)
   Defines type of chunk to scan for.

succ()
   Gets next memory data location.

size()
   Returns current data size.

writechunk(datas, datalen)
   Writes some datas to a chunk.

version()
   Returns IFFParser class version and revision.



Error Table
```

1.2 SAS/C and StormC Notes

```
SAS/C and StormC Notes

We provide ".o" files for both SAS/C and StormC compilers.


NOTE:
  The StormC compiler we are using is a _demo_ version, so we cannot create
  optimized version of the class.
```

We have asked to Haage & Partner a FREE version of StormC, but we receive no answer at all. We are still waiting.
We are not going to buy it because we usually write code using AmigaE and furthermore, we have already bought SAS/C compiler.

StormC supports exceptions, while SAS/C doesn't. So we have been forced to find a "work around" for SAS/C compiler.

We have introduced the error() method which will return the last error encountered during program execution.

So keep in mind this: when an error occurs, StormC version will raise an exception, while SAS/C version will return an error code or NULL and you will have to check error() status.

1.3 IFFParser C-Only Functions

C-Only Functions

We have created this function:

```
       NAME: id(idname)

   SYNOPSIS: ULONG id(STRPTR idname)

DESCRIPTION: This function is a short cut to allow you to easily create "chunks" ↩
    name.
             As you may now, chunks name are ULONG values. Usually you have to  ↩
                use
             the MAKE_ID() macro, but the id() function is faster.

      INPUT: idname       - a string, for example "ILBM", "BODY"....

    RETURNS: the corrisponding ULONG value.
```

We hope you'll find it useful

1.4 Introduction

IFFParser is an Object to easily read/write IFF files.

It is not so powerful as IFFParse.library: it misses some features, but I have written it to match all my needs, and it does. I hope you will enjoy using. Look at the examples for working programs.

1.5 Author's Info

Original By: Fabio Rotondo (fsoft@intercom.it)

E Version By: Fabio Rotondo

Address:

Fabio Rotondo
C.so Vercelli 9
28100 Novara
ITALY

e-mail: fsoft@intercom.it
Fabio.Rotondo@deagostini.it

Phone: (ITA) - (0)321 - 459676 (home)
(ITA) - (0)321 - 424272 (office)
(ITA) - (0)338 - 7336477 (GSM Phone)

Fax: (ITA) - (0)321 - 424560

Web: http://www.intercom.it/~fsoft (my home page)

http://www.intercom.it/~fsoft/ablast.html (Amiga Blast Home Page)

1.6 Amiga Foundation Classes Module: IFFParser / Error Table

VALUE	COSTANT NAME	DESCRIPTION
$0000	IFFERR_NOMEMORY	No Memory.
$0001	IFFERR_IFFLIB	Could not open iffparse.library.
$0002	IFFERR_ALLOC	Could not AllocIFF().
$0003	IFFERR_OPENSAVE	Could not create filename with Open(fname, MODE_NEWFILE).
$0004	IFFERR_OPENIFF	Could not OpenIFF().
$0005	IFFERR_INIT	IFFParser object not initialized with load() or save().
$0006	IFFERR_PUSH	Could not Push current chunk.
$0007	IFFERR_WRITECHUNKBYTES	WriteChunkBytes() failed.
$0008	IFFERR_POP	Could not Pop current chunk.
$0009	IFFERR_OPENFILE	Could not Open() file for reading.
$000A	IFFERR_COLLECTION	Could not do a CollectionChunk() call.
$000B	IFFERR_EXIT	Could not assign StopOnExit.

1.7 IFFParser / History

HISTORY

```
V1.00 - Initial Release

V1.10 - ENH:
               getheader()
               method optimized.

        FIX:
               succ()
               returns the right pointer.

V1.20 - ADD:
               error()
               method

V1.30 - ADD:
               empty()
               method

        ENH:
               Error Table
               with constant names
        ENH: Better docs.

V1.40 - ADD:  resourceTracker  support.
```

1.8 Amiga Foundation Classes Module: IFFParser / iffparser()

```
          NAME: iffparser(resTracker=NIL:PTR TO resourceTracker)

      SYNOPSIS: iffparser(resourceTracker * resTracker = NULL)

    DESCRIPTION: This method will initialize the iffparser object.

         INPUT: resTracker   - (Optional) Pointer TO a valid resourceTracker
                               class.

       RESULTS: NONE.

      SEE ALSO:  resourceTracker
```

1.9 Amiga Foundation Classes Module: IFFParser / close()

```
                          NAME: close()

      SYNOPSIS: void close(void)

    DESCRIPTION: This method will close curent IFF session.

         INPUT: NONE.

       RESULTS: NONE.
```

```
   SEE ALSO:
             load()

             save()
```

1.10 Amiga Foundation Classes Module: IFFParser / save(filename)

```
                        NAME: save(filename:PTR TO CHAR)

      SYNOPSIS: LONG save(STRPTR fname)

   DESCRIPTION: Use This method to create a new IFF file and to begin
                a IFF save session.

         INPUT: filename   - Name of the file to open / load.

       RESULTS: NONE.


      SEE ALSO:
             createchunk()

             writechunk()

             closechunk()

             close()
```

1.11 Amiga Foundation Classes Module: IFFParser / createchunk(type, id, size)

```
                        NAME: createchunk(type, id, size=IFFSIZE_UNKNOWN)

      SYNOPSIS: LONG createchunk(ULONG type, ULONG id, ULONG size =  ↩
         IFFSIZE_UNKNOWN)

   DESCRIPTION: This method will create a new chunk where to write in.

         INPUT: type       - (LONG) type of the chunk. (ex. "ILBM")
                id         - (LONG) id of the chunk. (ex. "FORM")

       RESULTS: NONE.


      SEE ALSO:
             closechunk()

             writechunk()

             save()
```

1.12 Amiga Foundation Classes Module: IFFParser / writechunk(datas, datalen)

```
                        NAME: writechunk(data, datalen)

   SYNOPSIS: LONG writechunk(STRPTR data, ULONG datalen)

DESCRIPTION: Use This method to write some data into a chunk.

      INPUT: data    - (PTR TO LONG) memory location of your datas.
             datalen - (LONG) length in bytes of your data.

    RESULTS: NONE.


   SEE ALSO:
             save()

             createchunk()

             closechunk()
```

1.13 Amiga Foundation Classes Module: IFFParser / closechunk()

```
                        NAME: closechunk()

   SYNOPSIS: LONG closechunk(void)

DESCRIPTION: This method will close current chunk.

      INPUT: NONE.

    RESULTS: NONE.


   SEE ALSO:
             createchunk()

             writechunk()

             save()
```

1.14 Amiga Foundation Classes Module: IFFParser / load(filename)

```
                        NAME: load(filename:PTR TO CHAR)

   SYNOPSIS: LONG load(STRPTR filename)

DESCRIPTION: Use This method to open an already existing IFF file and
             to begin a IFF load session.

      INPUT: filename   - Name of the file to open / load.
```

```
    RESULTS: NONE.


   SEE ALSO:
             setscan()

             scan()

             close()
```

1.15 Amiga Foundation Classes Module: IFFParser / setscan(type, id)

```
                        NAME: setscan(type, id)

      SYNOPSIS: LONG setscan(ULONG type, ULONG id)

   DESCRIPTION: Use This method to set chunk that will be loaded by
                scan().

         INPUT: type       - (LONG) type of chunk. (ex. "ILBM")
                id         - (LONG) id of chunk.   (ex. "FORM")

      RESULTS: NONE.


        NOTES: You can do more than a single setscan() before scan()ing the
               IFF file. This is the biggie ;)
               In this way you can load up in memory all you need in a single
               file access.

      SEE ALSO:
             scan()

             load()
```

1.16 Amiga Foundation Classes Module: IFFParser / first(type, id)

```
                        NAME: first(type, id)

      SYNOPSIS: APTR first(ULONG type, ULONG id)

   DESCRIPTION: Use This method to position IFFParser object to the FIRST
                memory data location of a specific kind.

         INPUT: type       - (LONG) type of the chunk. (ex. "ILBM")
                id         - (LONG) id of the chunk. (ex. "FORM")

      RESULTS: a PTR TO LONG memory location of the data (may be NIL)


      SEE ALSO:
             load()
```

```
              scan()

              setscan()

              succ()
```

1.17 Amiga Foundation Classes Module: IFFParser / addr()

```
                        NAME: addr()

    SYNOPSIS: APTR addr(void)

 DESCRIPTION: This method will return current memory data address.

       INPUT: NONE.

     RESULTS: a PTR TO LONG specifying memory data address (may be NIL)

    SEE ALSO:
              size()

              first()

              succ()
```

1.18 Amiga Foundation Classes Module: IFFParser / size()

```
                        NAME: size()

    SYNOPSIS: LONG size(void)

 DESCRIPTION: Use This method to get size of current memory data.

       INPUT: NONE.

     RESULTS: size   - (LONG) size of the memory data. (May be NIL)


    SEE ALSO:
              first()

              succ()
```

1.19 Amiga Foundation Classes Module: IFFParser / succ()

```
                        NAME: succ()

    SYNOPSIS: APTR succ(void)
```

```
DESCRIPTION: Use This method to position to the next memory data.

      INPUT: NONE.

    RESULTS: a PTR TO LONG to the new location of memory data. (May be NIL)


   SEE ALSO:
             load()

             first()
```

1.20 Amiga Foundation Classes Module: IFFParser / scan()

```
                       NAME: scan()

   SYNOPSIS: LONG scan(void)

DESCRIPTION: Use This method to begin scanning an IFF file.

      INPUT: NONE.

    RESULTS: NONE.


   SEE ALSO:
             setscan()

             load()
```

1.21 Amiga Foundation Classes Module: IFFParser / getheader(string, filename)

```
         NAME: getheader(string:PTR TO CHAR, filename:PTR TO CHAR)

   SYNOPSIS: LONG getheader(STRPTR s, STRPTR filename)

DESCRIPTION: Use This method to determinate the kind of a file.

      INPUT: string     - A _VALID_ Estring already initialized.
             filename   - Name of the file to examine.

    RESULTS: string     - your _VALID_ Estring will be filled with the
                          file header (ex. "ILBM")

       NOTE: This method returns a STRING not a LONG!


   SEE ALSO:
```

1.22 Amiga Foundation Classes Module: IFFParser / exit(type, id)

```
                        NAME: exit(type, id)

      SYNOPSIS: LONG exit(ULONG type, ULONG id)

   DESCRIPTION: Use This method to determinate WHEN scan() should stop.

         INPUT: type       - (LONG) type of the chunk. (ex. "ILBM")
                id         - (LONG) id of the chunk. (ex. "FORM")

       RESULTS: NONE.


      SEE ALSO:
                load()

                scan()

                setscan()
```

1.23 Amiga Foundation Classes Module: IFFParser / version()

```
           NAME: version()

      SYNOPSIS: LONG version(BOOL rev = FALSE)

   DESCRIPTION: This method returns version and revision of the class.

         INPUT: NONE.

       RESULTS: this method will return TWO values: version and revision.

 PORTING NOTES: The C++ class version behaves differently, since C++ cannot
                return two values at the same time.

                So, if you call just version(), you'll get the VERSION value,
                while calling version(TRUE), you'll get the REVISION value.
                This is just a quick and dirty workaround...

      SEE ALSO:
```

1.24 Amiga Foundation Classes Module: IFFParser / error()

```
           NAME: error()

      SYNOPSIS: ULONG error(void)

   DESCRIPTION: This method returns the last error raised.

         INPUT: NONE.

       RESULTS: this method will return the last error code raised.
```

SEE ALSO:

1.25 Amiga Foundation Classes: IFFParser/empty()

NAME: empty()

DESCRIPTION: This method will check if current class contains some data or not.

INPUT: NONE

RESULTS: TRUE - The class is empty.

FALSE - The class contains something.

SEE ALSO: