

**ini\_lib**

**COLLABORATORS**

	<i>TITLE :</i> ini_lib		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 10, 2022	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>ini_lib</b>	<b>1</b>
1.1	ini_lib.guide	1
1.2	ini.library/--background--	3
1.3	ini.library/iniAddContext	4
1.4	ini.library/iniAddContextItem	6
1.5	ini.library/iniAllocNameStr	7
1.6	ini.library/iniAllocPMem	8
1.7	ini.library/iniCheckComment	9
1.8	ini.library/iniClose	10
1.9	ini.library/iniCreateContext	11
1.10	ini.library/iniCreateContextItem	12
1.11	ini.library/iniDeleteContext	14
1.12	ini.library/iniDeleteContextItem	14
1.13	ini.library/iniFindContext	15
1.14	ini.library/iniFindItem	17
1.15	ini.library/iniFloatToStr	18
1.16	ini.library/iniFreeContext	19
1.17	ini.library/iniFreeContextItem	20
1.18	ini.library/iniFreeNameStr	21
1.19	ini.library/iniFreePMem	22
1.20	ini.library/iniGetArrayLine	23
1.21	ini.library/iniGetArrayPos	24
1.22	ini.library/iniGetByteA	25
1.23	ini.library/iniGetContextItem	27
1.24	ini.library/iniGetContextItemData	28
1.25	ini.library/iniGetContextItemDataA	29
1.26	ini.library/iniGetContextName	30
1.27	ini.library/iniGetFloat	31
1.28	ini.library/iniGetFloatA	32
1.29	ini.library/iniGetLong	34

---

---

1.30	ini.library/iniGetLongA . . . . .	36
1.31	ini.library/iniGetNumArrays . . . . .	38
1.32	ini.library/iniGetStr . . . . .	39
1.33	ini.library/iniGetStrA . . . . .	41
1.34	ini.library/iniGetWordA . . . . .	43
1.35	ini.library/iniInsertContext . . . . .	45
1.36	ini.library/iniInsertContextItem . . . . .	46
1.37	ini.library/iniIntToStr . . . . .	46
1.38	ini.library/iniOpenDefault . . . . .	48
1.39	ini.library/iniOpenFile . . . . .	49
1.40	ini.library/iniOpenFromFH . . . . .	50
1.41	ini.library/iniOpenMem . . . . .	51
1.42	ini.library/iniPutByteA . . . . .	52
1.43	ini.library/iniPutFloat . . . . .	54
1.44	ini.library/iniPutFloatA . . . . .	56
1.45	ini.library/iniPutLong . . . . .	58
1.46	ini.library/iniPutLongA . . . . .	60
1.47	ini.library/iniPutStr . . . . .	62
1.48	ini.library/iniPutStrA . . . . .	64
1.49	ini.library/iniPutWordA . . . . .	65
1.50	ini.library/iniReadByteA . . . . .	67
1.51	ini.library/iniReadFloat . . . . .	69
1.52	ini.library/iniReadFloatA . . . . .	70
1.53	ini.library/iniReadLong . . . . .	72
1.54	ini.library/iniReadLongA . . . . .	73
1.55	ini.library/iniReadStr . . . . .	75
1.56	ini.library/iniReadStrA . . . . .	77
1.57	ini.library/iniReadWordA . . . . .	78
1.58	ini.library/iniRemContext . . . . .	80
1.59	ini.library/iniRemContextItem . . . . .	81
1.60	ini.library/iniSaveFile . . . . .	82
1.61	ini.library/iniSaveToFH . . . . .	83
1.62	ini.library/iniSetNameStr . . . . .	84
1.63	ini.library/iniSetString . . . . .	85
1.64	ini.library/iniStrToFloat . . . . .	87
1.65	ini.library/iniStrToInt . . . . .	88
1.66	ini.library/iniWriteByteA . . . . .	89
1.67	ini.library/iniWriteFloat . . . . .	91
1.68	ini.library/iniWriteFloatA . . . . .	92

---

---

1.69	ini.library/iniWriteLong . . . . .	94
1.70	ini.library/iniWriteLongA . . . . .	96
1.71	ini.library/iniWriteStr . . . . .	98
1.72	ini.library/iniWriteStrA . . . . .	99
1.73	ini.library/iniWriteWordA . . . . .	101

---

# Chapter 1

## ini\_lib

### 1.1 ini\_lib.guide

#### TABLE OF CONTENTS

`ini.library/--background--`

`ini.library/iniAddContext`

`ini.library/iniAddContextItem`

`ini.library/iniAllocNameStr`

`ini.library/iniAllocPMem`

`ini.library/iniCheckComment`

`ini.library/iniClose`

`ini.library/iniCreateContext`

`ini.library/iniCreateContextItem`

`ini.library/iniDeleteContext`

`ini.library/iniDeleteContextItem`

`ini.library/iniFindContext`

`ini.library/iniFindItem`

`ini.library/iniFloatToStr`

`ini.library/iniFreeContext`

`ini.library/iniFreeContextItem`

`ini.library/iniFreeNameStr`

`ini.library/iniFreePMem`

---

ini.library/iniGetArrayLine  
ini.library/iniGetArrayPos  
ini.library/iniGetByteA  
ini.library/iniGetContextItem  
ini.library/iniGetContextItemData  
ini.library/iniGetContextItemDataA  
ini.library/iniGetContextName  
ini.library/iniGetFloat  
ini.library/iniGetFloatA  
ini.library/iniGetLong  
ini.library/iniGetLongA  
ini.library/iniGetNumArrays  
ini.library/iniGetStr  
ini.library/iniGetStrA  
ini.library/iniGetWordA  
ini.library/iniInsertContext  
ini.library/iniInsertContextItem  
ini.library/iniIntToStr  
ini.library/iniOpenDefault  
ini.library/iniOpenFile  
ini.library/iniOpenFromFH  
ini.library/iniOpenMem  
ini.library/iniPutByteA  
ini.library/iniPutFloat  
ini.library/iniPutFloatA  
ini.library/iniPutLong  
ini.library/iniPutLongA  
ini.library/iniPutStr

---

---

```
ini.library/iniPutStrA
ini.library/iniPutWordA
ini.library/iniReadByteA
ini.library/iniReadFloat
ini.library/iniReadFloatA
ini.library/iniReadLong
ini.library/iniReadLongA
ini.library/iniReadStr
ini.library/iniReadStrA
ini.library/iniReadWordA
ini.library/iniRemContext
ini.library/iniRemContextItem
ini.library/iniSaveFile
ini.library/iniSaveToFH
ini.library/iniSetNameStr
ini.library/iniSetString
ini.library/iniStrToFloat
ini.library/iniStrToInt
ini.library/iniWriteByteA
ini.library/iniWriteFloat
ini.library/iniWriteFloatA
ini.library/iniWriteLong
ini.library/iniWriteLongA
ini.library/iniWriteStr
ini.library/iniWriteStrA
ini.library/iniWriteWordA
```

## 1.2 ini.library/--background--

---



## PURPOSE

The 'ini.library' was created, because there was no easy way of handling configuration files. Each program had it's own format.

This library tries to help you in making a standard file format for configuration files. These files are in plain ASCII, so you can use your favourite text editor to edit them, and, in addition, of course, you can use the prefs section of your program.

## OVERVIEW

The initialization files (INI files) come from Microsoft Windows 3.x. They are built up as follows:

```
[Context1]
  Item1 = Long1
  Item2 = Long2
  Item3 = LongA1, LongA2, LongA3,
         LongA4, LongA5, LongA6,
  Item4 = Float1
  Item5 = String1
```

```
[Context2]
  Item1 = Float1
...
```

\* Easy accessible

The library makes it easy to read/write access those context/item fields. You can easily add/remove new items, contexts, etc.

\* Comments, etc. are all handled & preserved automatically!

Yes, if you update an INI context item, the comment after the line is preserved. That means, if you update the INI file in a program, you don't need to have any fears of lossing something.

\* Both high- and low-level access functions

You can choose between simply access and complex access. Low level functions are for complex access, high level functions are for quick and easy access. But the high level functions are a little slower.

## 1.3 ini.library/iniAddContext

NAME

iniAddContext -- adds a new freshly allocated context to an INI file structure

## SYNOPSIS

---

```
iniAddContext( iniFile, ContextStr );
              A0      A1
```

```
void iniAddContext( struct iniFile *, struct iniContext *);
```

#### FUNCTION

Adds a freshly created context to a specified INI file.  
To add items, you first need to add context items, see there.

#### INPUTS

iniFile - Pointer to INI structure where to add it  
ContextStr - Freshly context structure to add

#### EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

/* Check if "MyContext" already exists */
if (!( context = FindContext ( ini, "MyContext" )))
{
    /* If not, create it! */
    if (!( context = CreateContext ( "MyContext" )))
    {
        puts ( "Couldn't create my context !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContext ( ini, context );
}
```

#### NOTES

You need to call this function to make a \*NEW\* context available to the other functions.

#### BUGS

#### SEE ALSO

```
iniCreateContext
(),
iniFreeContext
(),
iniRemContext
(),
iniInsertContext
```

```
    ),  
    iniDeleteContext  
    ( ), <libraries/ini_lib.h>
```

## 1.4 ini.library/iniAddContextItem

### NAME

iniAddContextItem -- adds a new freshly allocated context item to  
an context structure

### SYNOPSIS

```
iniAddContextItem( ContextStr, ContextItemLine );  
                  A0          A1
```

```
void iniAddContextItem( struct iniContext *,  
                       struct iniContextItemLine *);
```

### FUNCTION

Adds a freshly created context item to a specified context.

### INPUTS

ContextStr - Pointer to context structure where to add it  
ContextItemLine - Freshly context item structure to add

### EXAMPLE

```
struct iniFile *ini;  
struct iniContext *context;  
struct iniContextItemLine *contextitem;  
  
/* Let's open an INI file */  
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );  
  
context = FindContext ( ini, "MyContext" );  
  
if (!( contextitem = FindContextItem ( context, "MyItem" )))  
{  
    /* If not, create it! */  
    if (!( contextitem = CreateContextItem ( "MyItem" )))  
    {  
        puts ( "Couldn't create my context item !" );  
  
        exit ( 20 );  
    }  
  
    /* Make context available for access */  
    AddContextItem ( context, contextitem );  
}
```

## NOTES

You need to call this function to make a \*NEW\* context item available to the other functions.

## BUGS

## SEE ALSO

```
iniCreateContextItem
(),
iniFreeContextItem
(),
iniRemContextItem
(),

iniInsertContextItem
(),
iniDeleteContextItem
(), <libraries/ini_lib.h>
```

## 1.5 ini.library/iniAllocNameStr

## NAME

iniAllocNameStr -- allocate a name string compatible with the library

## SYNOPSIS

```
namestring = iniAllocNameStr( string );
D0                                     A0

STRPTR iniAllocNameStr( STRPTR string );
```

## FUNCTION

Allocates a name string out of a standard C-String. This is required if you use own strings in the library handlers.

## INPUTS

string - The string to be allocated. Must be null terminated.

## RESULT

namestring - The initialized name structure. To deallocate, use

```
iniFreeNameStr
() on this result.
```

---

## NOTES

The namestring is a copy of string, but it's freed via  
iniFreePMem  
( )

## BUGS

## SEE ALSO

```
iniFreeNameStr  
( ),  
iniSetNameStr  
( ),  
iniSetString  
( )
```

## 1.6 ini.library/iniAllocPMem

## NAME

iniAllocPMem -- allocate MEMF\_PUBLIC|MEMF\_CLEAR memory like AllocMem()  
but use memory pools if running under OS3.0+

## SYNOPSIS

```
memoryBlock = iniAllocPMem( byteSize );  
D0 D0
```

```
APTR iniAllocPMem( ULONG );
```

## FUNCTION

Allocates memory always with MEMF\_PUBLIC and MEMF\_CLEAR flags set and uses, if possible, the OS3.0+ memory pools. I have decided to implement this function, because the INI library allocates often very small chunks of memory, which is handled more efficiency with memory pools.

Each memory pool has a size of 32768 bytes and a threshold of 4096 bytes.

## INPUTS

byteSize - number of bytes to allocate.

## RESULT

memoryBlock - the first byte of the allocated memory or NULL if the allocation failed.

---

## NOTES

Please note that you **MUST** deallocate memory allocated with `iniAllocPMem()` with `iniFreePMem()` or it **\*MAY\*** crash!

`iniAllocPMem()` is used always for internal purposes, so that you **\*MUST\*** allocate all structures used with this lib with this function.

## BUGS

## SEE ALSO

```
iniFreePMem
(), exec.library/AllocMem(), exec.library/AllocPooled()
```

## 1.7 ini.library/iniCheckComment

## NAME

`iniCheckComment` -- Checks if a context line belongs to a comment

## SYNOPSIS

```
success = iniCheckComment( ContextStr, ContextItemLine );
D0                                     A0         A1

BOOL iniCheckComment( struct iniContext *,
    struct iniContextItemLine *);
```

## FUNCTION

Checks if the given context item line is part of a multiline comment. This function is mainly for internal use to easy handle the parsing of lines.

## INPUTS

`ContextStr` - A pointer to context structure which contains the line to be examined.  
`ContextItemLine` - A pointer to the context item line which should be examined.

## RESULT

`success` - TRUE if the line is commented else NULL.

## NOTES

BUGS

SEE ALSO

<libraries/ini\_lib.h>

## 1.8 ini.library/iniClose

NAME

iniClose -- Deallocate a loaded INI file

SYNOPSIS

```
iniClose( iniFile );  
A0
```

```
void iniClose( struct iniFile *);
```

FUNCTION

This function is used to deallocate the memory required by the the loaded INI file. It deallocates everything in the given INI file structure.

INPUTS

iniFile - A pointer to the INI file structure to be freed

EXAMPLE

```
struct iniFile *ini;  
ULONG Width;  
  
/* Let's open an INI file to fiddle around with */  
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );  
  
/* Let's do some evaluating (read screen width) */  
Width = iniReadLong ( ini, "Screen", "Width", 640L, 0L );  
  
/* We have the info we need. So close it and free memory */  
iniClose ( ini );
```

NOTES

All memory blocks inside must be allocated using  
iniAllocPMem  
( ) or  
  
iniAllocNameStr  
( ) or it may crash!

BUGS

---

SEE ALSO

```
iniOpenFile
(),
iniOpenFromFH
(),
iniOpenMem
(), <libraries/ini_lib.h>
```

## 1.9 ini.library/iniCreateContext

NAME

iniCreateContext -- Creates a new context chunk to be used

SYNOPSIS

```
ContextStr = iniCreateContext( ContextName );
D0
```

A0

```
struct iniContext *iniCreateContext( STRPTR ContextName );
```

FUNCTION

Creates a new context structure. It must be added with

```
iniAddContext
()
```

to an INI file structure. The name given mustn't be an AllocNameStr() string. Because it's created automatically during the creation process.

INPUTS

ContextName - The line string of the context

EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

/* Check if "MyContext" already exists */
if (!( context = FindContext ( ini, "MyContext" )))
{
    /* If not, create it! */
    if (!( context = CreateContext ( "MyContext" )))
    {
        puts ( "Couldn't create my context !" );
    }
}
```



```
    exit ( 20 );
}

/* Make context available for access */
AddContext ( ini, context );
}
```

#### NOTES

ContextName is only the context name itself, not the full line, e.g. "Display", and not "[ Display ]".

#### BUGS

In ini.library v31 you had to give the full context line, i.e. "[ Display ]".

#### SEE ALSO

```
iniCreateContextItem
(),
iniFreeContext
(),
iniAddContext
(),

iniInsertContext
(), <libraries/ini_lib.h>
```

## 1.10 ini.library/iniCreateContextItem

#### NAME

iniCreateContextItem -- creates a new context item to be used

#### SYNOPSIS

```
ContextItemLine = iniCreateContextItem( CStr );
D0                                     A0
```

```
struct iniContextItemLine *iniCreateContextItem( STRPTR CStr );
```

#### FUNCTION

Creates a new context item line to be used. The string will be used as a context line name. Add the result structure to the context structure to make it available in the INI file structure.

#### INPUTS

CStr - A null terminated string which will be stored as a

---

```
iniAllocNameStr  
( ) string.
```

**RESULT**

ContextItemLine - The context item structure that can be added to the desired context.

**EXAMPLE**

```
struct iniFile *ini;  
struct iniContext *context;  
struct iniContextItemLine *contextitem;  
  
/* Let's open an INI file */  
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );  
  
context = FindContext ( ini, "MyContext" );  
  
if (!( contextitem = FindContextItem ( context, "MyItem" )))  
{  
    /* If not, create it! */  
    if (!( contextitem = CreateContextItem ( "MyItem" )))  
    {  
        puts ( "Couldn't create my context item !" );  
  
        exit ( 20 );  
    }  
  
    /* Make context available for access */  
    AddContextItem ( context, contextitem );  
}
```

**NOTES**

The context item is initialized with 0.

**BUGS**

In ini.library v31 the context item was not initialized.

**SEE ALSO**

```
iniCreateContext  
( ),  
iniFreeContextItem  
( ),  
iniAddContextItem  
( ),  
  
iniInsertContextItem  
( ), <libraries/ini_lib.h>
```

## 1.11 ini.library/iniDeleteContext

NAME

iniDeleteContext -- deletes a context from an INI file structure

SYNOPSIS

```
iniDeleteContext( ContextStr );  
                A0
```

```
void iniDeleteContext( struct iniContext *);
```

FUNCTION

Deletes a context of an INI file and all its associated lines. Memory will not be freed. To deallocate, use  
iniFreeContext  
( )

INPUTS

ContextStr - The context structure to be deleted

RESULT

NOTES

Please note that this function removes only the node from the INI file structure and doesn't deallocate any memory.

BUGS

SEE ALSO

```
        iniFreeContext  
        ( ),  
        iniRemContext  
        ( ),  
        iniInsertContext  
        ( ),  
<libraries/ini_lib.h>
```

## 1.12 ini.library/iniDeleteContextItem

NAME

iniDeleteContextItem -- deletes a context item line from an INI file.

---

## SYNOPSIS

```
iniDeleteContextItem( ContextItemLine );
                    A0
```

```
void iniDeleteContextItem( struct iniContextItemLine *);
```

## FUNCTION

Deletes a context item line from an INI context. It doesn't deallocate any memory, so it can be added otherwise.

## INPUTS

ContextItemLine - The pointer of the context item line to be deleted.

## RESULT

## NOTES

Please note that just the node is removed from the context structure and there are no deallocation processes. Use

```
iniFreeContextItem
()
```

for this purpose.

## BUGS

## SEE ALSO

```
iniFreeContextItem
(),
iniRemContextItem
(),
iniInsertContextItem
(),
<libraries/ini_lib.h>
```

## 1.13 ini.library/iniFindContext

## NAME

iniFindContext -- Search for a context in an INI file.

## SYNOPSIS

```
ContextStr = iniFindContext( iniFile, ContextName, Flags );
D0                    A0      A1      D0
```

```
struct iniContext *iniFindContext( struct iniFile *,
STRPTR, ULONG );
```

## FUNCTION

Searches a loaded INI file for the specified context.

## INPUTS

iniFile - Pointer to INI structure to search  
ContextName - Name of the context to be searched  
Flags - Search flags. They're currently defined as:  
    INIF\_ContextCase - Set this flag if the search of the context  
    name should be case sensitive.

## RESULT

ContextStr - The context structure if the context was found else  
NULL.

## EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

/* Check if "MyContext" already exists */
if (!( context = FindContext ( ini, "MyContext" )))
{
    /* If not, create it! */
    if (!( context = CreateContext ( "MyContext" )))
    {
        puts ( "Couldn't create my context !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContext ( ini, context );
}
```

## NOTES

## BUGS

## SEE ALSO

```
iniCreateContext
(),
iniFindItem
(), <libraries/ini_lib.h>
```

## 1.14 ini.library/iniFindItem

NAME

iniFindItem -- finds a context item in a specified context

SYNOPSIS

```
ContextItemLine = iniFindItem( ContextStr, ContextItemName, Flags );
D0                                     A0             A1             D0
```

```
struct iniContextItemLine *iniFindItem( struct iniContext *,
STRPTR, ULONG );
```

FUNCTION

Searches for a context item in the specified context

INPUTS

ContextStr - Context structure where to search in  
ContextItemName - Name of the context item to be searched  
Flags - Search flags. They're currently defined as:  
    INIF\_ContextItemCase - Set this flag if the search of the context  
    item name should be case sensitive.

RESULT

ContextItemLine - The context item structure if the context item  
was found else NULL.

EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}
```

```
    }
```

NOTES

BUGS

SEE ALSO

```
    iniCreateContextItem
    (),
    iniFindContext
    (), <libraries/ini_lib.h>
```

## 1.15 ini.library/iniFloatToStr

NAME

iniFloatToStr -- Converts a quick float value to a string.

SYNOPSIS

```
string = iniFloatToStr( Buffer, Float, FltFormat, IntLen, FracLen,
D0                A0      D0      D1      D2      D3
                    ZeroSep );
                    D4:8
```

```
STRPTR iniFloatToStr( STRPTR, ULONG, ULONG, ULONG, ULONG, UBYTE );
```

FUNCTION

This function is used to convert a quick float value to a standard ASCII string. A quick float value has in it's upper 16-bits the decimal value and in the lower 16-bits the fraction. That means, the highest possible accuracy is 1/65536.

INPUTS

Buffer - A pointer to at least a 128 byte buffer or NULL to create a new one.

Float - Quick float value to convert.

FltFormat - Format of the floating point value. Can be any of:

INI\_FLOAT\_FORMAT\_DEC - Use decimal with point separator

INI\_FLOAT\_UNSIGNED - Add this to indicate unsigned quick float

IntLen - Forced length of integer part or NULL for no force.

FracLen - Forced length of fractional part or NULL for no force.

ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

RESULT

string - Pointer to the string where the converted string is stored.

## EXAMPLE

```
STRPTR Buffer;

Buffer = iniFloatToStr ( 0x28000, INI_FLOAT_FORMAT_DEC, 3, 2, ' ');

/* Buffer will contain: " 2.50" */

puts ( Buffer );

iniFreeNameStr ( Buffer );
```

## NOTES

## BUGS

The buffer is not checked for overflow. That means, IntLen and FracLen should not be greater than 90% of your buffer when added together.

## SEE ALSO

```
iniStrToFloat
(),
iniStrToInt
(),
iniIntToStr
(), <libraries/ini_lib.h>
```

## 1.16 ini.library/iniFreeContext

## NAME

iniFreeContext -- Deletes if necessary and deallocates a context structure.

## SYNOPSIS

```
iniFreeContext( ContextStr );
    A0

void iniFreeContext( struct iniContext *);
```

## FUNCTION

This function removes first (see iniDeleteContext ()) , if necessary a context structure and then deallocates the memory used by it using

```
iniFreePMem
() and
```

---



```
iniFreeNameStr  
( ).
```

**INPUTS**

ContextStr - A pointer to a context structure to be deallocated.

**RESULT****NOTES**

The structure MUST be allocated with  
iniAllocPMem  
( ) and the strings  
must be allocated with  
iniAllocNameStr  
( )

**BUGS****SEE ALSO**

```
iniCreateContext  
( ),  
iniRemContext  
( ),  
iniDeleteContext  
( ),  
  
iniFreeContextItem  
( ), <libraries/ini_lib.h>
```

## 1.17 ini.library/iniFreeContextItem

**NAME**

iniFreeContextItem -- Deallocates a context item structure

**SYNOPSIS**

```
iniFreeContextItem( ContextItemLine );  
A0
```

```
void iniFreeContextItem( struct iniContextItemLine * );
```

**FUNCTION**

Removes (if necessary) a context item line and deallocates the memory required by it afterwards.

```
iniFreePMem
```

---

`()` and  
`iniFreeNameStr`  
`()` are  
used for deallocation.

#### INPUTS

`ContextItemLine` - Pointer to the context item line to be deallocated.

#### RESULT

#### NOTES

The structure MUST be allocated with  
`iniAllocPMem`  
`()` and the strings  
need to be allocated with  
`iniAllocNameStr`  
`()`.

#### BUGS

#### SEE ALSO

```
iniCreateContextItem
(),
iniRemContextItem
(),
iniDeleteContextItem
(),

iniFreeContext
(), <libraries/ini_lib.h>
```

## 1.18 ini.library/iniFreeNameStr

NAME  
`iniFreeNameStr` -- Deallocate an  
`iniAllocNameStr`  
`()` name structure.

#### SYNOPSIS

```
iniFreeNameStr( namestring );
A0

void iniFreeNameStr( STRPTR );
```

## FUNCTION

Deallocates a name structure, allocated previously by

```
iniAllocNameStr  
( )
```

## INPUTS

namestring - A pointer to a previously allocated name string. The size of deallocation is calculated automatically.

## RESULT

## NOTES

Please deallocate ONLY those things allocated with

```
iniAllocNameStr  
( )
```

with this function, since the allocation mechanism may change at any time!

## BUGS

## SEE ALSO

```
iniAllocNameStr  
( ),  
iniAllocPMem  
( ),  
iniFreePMem  
( )
```

## 1.19 ini.library/iniFreePMem

## NAME

iniFreePMem -- deallocates memory allocated by

```
iniAllocPMem  
( )
```

## SYNOPSIS

```
iniFreePMem( memoryBlock, byteSize)  
            A1                D0
```

```
void iniFreePMem( APTR, ULONG );
```

## FUNCTION

Deallocates any memory allocated by iniFreePMem(), it works just like

---

the `exec.library/FreeMem` function.

#### INPUTS

`memoryBlock` - The block that should be deallocated  
`byteSize` - Number of bytes to be deallocated. It is aligned automatically.

#### RESULT

#### NOTES

Please use this function to deallocate ONLY `iniAllocPMem` () blocks, if you don't do this, the system may crash within OS3.0+, since the function uses memory pools in that case.

#### BUGS

#### SEE ALSO

`iniAllocPMem`  
 ()

## 1.20 ini.library/iniGetArrayLine

#### NAME

`iniGetArrayLine` -- Returns the array line of the context item number

#### SYNOPSIS

```
ContextItemPos = iniGetArrayLine( ContextStr, ContextItemLine,
D0                               A0           A1
                                Number );
D0
```

```
struct iniContextItemLine *iniGetArrayLine( struct iniContext *,
struct iniContextItemLine *, ULONG );
```

#### FUNCTION

Gets the context item line structure of the given context item's nth array (nth is the Number given in D0)

#### INPUTS

`ContextStr` - Context structure where context item is  
`ContextItemLine` - Context item line where's array's first entry.

Number - Array number of which the line address should be returned.

#### RESULT

ContextItemLine - Context item line address of the number given.

#### NOTES

This function is mainly only for internal purposes. It allows array handling faster.

#### BUGS

#### SEE ALSO

```
iniGetNumArrays
(),
iniGetArrayPos
(), <libraries/ini_lib.h>
```

## 1.21 ini.library/iniGetArrayPos

#### NAME

iniGetArrayPos -- Returns the position of the context item number.

#### SYNOPSIS

```
ContextItemPos = iniGetArrayPos( ContextStr, ContextItemLine,
D0                               A0           A1
                               Number );
D0
```

```
struct iniContextItemLine *iniGetArrayPos( struct iniContext *,
struct iniContextItemLine *, ULONG );
```

#### FUNCTION

Gets the line position of the given context item's nth array (nth is the Number given in D0)

#### INPUTS

ContextStr - Context structure where context item is  
ContextItemLine - Context item line where's array's first entry.  
Number - Array number of which the line position should be returned.

#### RESULT

ContextItemLine - Context item line address of the number given.

## NOTES

This function is mainly only for internal purposes. It allows easy array access.

## BUGS

## SEE ALSO

```
iniGetNumArrays  
(),  
iniGetArrayLine  
(), <libraries/ini_lib.h>
```

## 1.22 ini.library/iniGetByteA

## NAME

iniGetByteA -- reads a context item array into a (U)BYTE array.

## SYNOPSIS

```
success = iniGetByteA( ContextStr, ContextItemLine, Array, Entries );  
D0                A0                A1                A2                D0
```

```
BOOL iniGetByteA( struct iniContext *, struct iniContextItemLine *,  
    BYTE *, ULONG );
```

## FUNCTION

Reads a context item array and stores the read bytes into a (U)BYTE table you specified.

## INPUTS

ContextStr - The context structure where the context line is in  
ContextItemLine - The context item line where the array is  
Array - An (U)BYTE array where to store the values  
Entries - Number of entries to read (further entries will be ignored)

## RESULT

success - TRUE if line could be evaluated else FALSE

## EXAMPLE

```
struct iniFile *ini;  
struct iniContext *context;  
struct iniContextItemLine *contextitem;  
BYTE MyArray[4] = {-2, -1, -0, 1};
```

```
/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniGetByteA ( context, contextitem, MyArray, sizeof (MyArray));

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 25, 50, 75, 100

   then
   MyArray[4] = {25, 50, 75, 100};
   Entries which can't be evaluated are left unchanged.
*/
```

#### NOTES

Make sure that the given array is big enough to hold all values or some memory area may be overwritten.

Fields which can't be evaluated are left unchanged.

#### BUGS

#### SEE ALSO

```
    iniGetWordA
    ( ),
    iniGetLongA
    ( ),
    iniReadByteA
    ( ),
    iniPutByteA
    ( ),
<libraries/ini_lib.h>
```

## 1.23 ini.library/iniGetContextItem

### NAME

iniGetContextItem -- Gets the name of the context item

### SYNOPSIS

```
Buffer = iniGetContextItem( ContextStr, ContextItemLine, Buffer );  
D0                A0                A1                A2
```

```
STRPTR iniGetContextItem( struct iniContext *,  
                          struct iniContextItemLine *, STRPTR );
```

### FUNCTION

Gets the context item name of the context item line given

### INPUTS

ContextStr - Context structure where context item lies  
ContextItemLine - Pointer to context item line structure  
Buffer - Optional buffer where to store name or NULL to create new

### RESULT

Buffer - STRPTR to newly created buffer if none was specified or NULL on error. If existing buffer was given, it will be returned instead.

### NOTES

This function is called by all functions which evaluate INI file context item data. It is, in fact, a low level function. The given buffer must have a minimum size of 128 bytes.

If no buffer is given, the new one must be freed with  
iniFreeNameStr  
( )

### BUGS

### SEE ALSO

```
iniGetContextName  
( ),  
iniGetContextItemData  
( ),  
  
iniGetContextItemDataA  
( ), <libraries/ini_lib.h>
```



## 1.24 ini.library/iniGetContextItemData

NAME

iniGetContextItemData -- Gets the data of the context item

SYNOPSIS

```
Buffer = iniGetContextItemData( ContextStr, ContextItemLine, Buffer );
D0                                A0                A1                A2
```

```
STRPTR iniGetContextItemData( struct iniContext *,
                               struct iniContextItemLine *, STRPTR );
```

FUNCTION

Gets the context item data of the context item line given

INPUTS

ContextStr - Context structure where context item data lies  
ContextItemLine - Pointer to context item line structure  
Buffer - Optional buffer where to store data or NULL to create new

RESULT

Buffer - STRPTR to newly created buffer if none was specified or NULL on error. If existing buffer was given, it will be returned instead.

NOTES

This function is called by all functions which evaluate INI file context item data. It is, in fact, a low level function. The given buffer must have a minimum size of 128 bytes.

If no buffer is given, the new one must be freed with  
iniFreeNameStr  
( )

BUGS

SEE ALSO

```
iniGetContextName  
( ),  
iniGetContextItem  
( ), iniContextItemDataA(),  
<libraries/ini_lib.h>
```

## 1.25 ini.library/iniGetContextItemDataA

NAME

iniGetContextItemDataA -- Gets the data of the context item array

SYNOPSIS

```
Buffer = iniGetContextItemDataA( ContextStr, ContextItemLine,  
D0                               A0           A1  
                                Buffer, Number );  
                                A2           D0
```

```
STRPTR iniGetContextItemDataA( struct iniContext *,  
struct iniContextItemLine *, STRPTR, ULONG );
```

FUNCTION

Gets the context item array data of the context item line given

INPUTS

ContextStr - Context structure where context item array lies  
ContextItemLine - Pointer to context item line structure  
Buffer - Optional buffer where to store data or NULL to create new  
Number - The entry which should be extracted out of the array. NULL  
is the first one

RESULT

Buffer - STRPTR to newly created buffer if none was specified or NULL  
on error. If existing buffer was given, it will be returned  
instead.

NOTES

This function is called by all functions which evaluate INI file  
context item array data. It is, in fact, a low level function. The  
given buffer must have a minimum size of 128 bytes.  
If no buffer is given, the new one must be freed with  
iniFreeNameStr  
( )

BUGS

SEE ALSO

```
iniGetContextName  
( ),  
iniGetContextItem  
( ),  
iniGetContextItemData  
( ),  
<libraries/ini_lib.h>
```

## 1.26 ini.library/iniGetContextName

### NAME

iniGetContextName - Gets the name of the context

### SYNOPSIS

```
Buffer = iniGetContextName( ContextLine, Buffer );  
D0                                A0                A1  
  
STRPTR iniGetContextName( STRPTR, STRPTR );
```

### FUNCTION

Gets the context name of the context line given

### INPUTS

ContextStr - Context structure where context line lies  
Buffer - Optional buffer where to store name or NULL to create new

### RESULT

Buffer - STRPTR to newly created buffer if none was specified or NULL on error. If existing buffer was given, it will be returned instead.

### NOTES

This function is called by all functions which evaluate INI file context names. It is, in fact, a low level function. The given buffer must have a minimum size of 128 bytes.

If no buffer is given, the new one must be freed with  
iniFreeNameStr  
( )

### BUGS

### SEE ALSO

```
iniGetContextItem  
( ),  
iniGetContextItemData  
( ),  
  
iniGetContextItemDataA  
( ), <libraries/ini_lib.h>
```

## 1.27 ini.library/iniGetFloat

NAME

iniGetFloat -- Gets a quick floating point value

SYNOPSIS

```
QFloatValue = iniGetFloat( ContextStr, ContextItemLine, Default );
D0                A0                A1                D0
```

```
LONG iniGetFloat( struct iniContext *, struct iniContextItemLine *,
LONG );
```

FUNCTION

Reads a quick float value out of the given context item line

INPUTS

ContextStr - Context structure where quick float value lies in  
ContextItemLine - Context item line where to extract quick float  
Default - Default value to take if it can't be evaluated

RESULT

QFloatValue - The quick float value extracted out of the context item line

EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyFloat;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}
```

```

MyFloat = iniGetFloat ( context, contextitem, 0x28000 );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 3.0

   then MyFloat will contain 0x30000. If this context or context item
   is not available, MyFloat will contain 0x28000 (default) instead.
*/

```

#### NOTES

This function is called from  
     iniReadFloat  
     ().

Only the first four fractional digits are evaluated. However, the  
 5th digit is evaluated for rounding purposes.

#### BUGS

#### SEE ALSO

```

iniGetLong
(),
iniGetStr
(),
iniGetFloatA
(),
iniPutFloat
(),

iniReadFloat
(),
iniWriteFloat
(), <libraries/ini_lib.h>

```

## 1.28 ini.library/iniGetFloatA

#### NAME

iniGetFloatA -- Gets quick floating point value(s) out of an array

#### SYNOPSIS

```

success = iniGetFloatA( ContextStr, ContextItemLine, Array, Entries );
D0                          A0                  A1                  A2          D0

```

```

BOOL iniGetFloatA( struct iniContext *, struct iniContextItemLine *,
                  LONG *, ULONG );

```

---

## FUNCTION

Reads one or more quick float value(s) out of the given context item line

## INPUTS

ContextStr - Context structure where quick float values lie in  
ContextItemLine - Context item line where to extract quick floats  
Array - The array where to store the quick float values  
Entries - Number of array entries. If the array in the INI file is bigger, the remaining entries will be ignored.

## RESULT

success - TRUE if accessing was successful else NULL.

## EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyFloat[4] = {-0x10000, -0x8000, 0, 0x8000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniGetFloatA ( context, contextitem, MyFloat,
               (sizeof (MyFloat) / sizeof (LONG)) );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 1.0, 1.5, 2.0, 2.5

   then
   MyFloat[4] = {0x10000, 0x18000, 0x20000, 0x28000};
   Entries which can't be evaluated are left unchanged.
*/
```

## NOTES

This function is called from

```
iniReadFloatA  
( ).
```

Only the first four fractional digits are evaluated. However, the 5th digit is evaluated for rounding purposes.

Array fields which can't be evaluated (e.g. bad syntax) are left unchanged. So it's good to fill the array with default values first.

BUGS

SEE ALSO

```
iniGetLongA  
( ),  
iniGetStrA  
( ),  
iniGetFloat  
( ),  
iniPutFloatA  
( ),  
  
iniReadFloatA  
( ),  
iniWriteFloatA  
( ), <libraries/ini_lib.h>
```

## 1.29 ini.library/iniGetLong

NAME

iniGetLong -- Gets a long integer value

SYNOPSIS

```
LongValue = iniGetLong( ContextStr, ContextItemLine, Default );  
D0                A0                A1                D0
```

```
LONG iniGetLong( struct iniContext *, struct iniContextItemLine *,  
LONG );
```

FUNCTION

Reads a long integer value out of the given context item line

INPUTS

ContextStr - Context structure where long integer value lies in  
ContextItemLine - Context item line where to extract long integer  
Default - Default value to take if it can't be evaluated

---

## RESULT

LongValue - The long integer value extracted out of the context item line

## EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyLong;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

MyLong = iniGetLong ( context, contextitem, 12345678 );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = -256

   then MyLong will contain -256. If this context or context item
   is not available, MyLong will contain 12345678 (default) instead.
*/
```

## NOTES

This function is called from  
iniReadLong  
().

## BUGS

## SEE ALSO

```
iniGetFloat
(),
iniGetStr
(),
iniGetLongA
```



```

    (),
    iniPutLong
    (),

    iniReadLong
    (),
    iniWriteLong
    (), <libraries/ini_lib.h>

```

### 1.30 ini.library/iniGetLongA

#### NAME

iniGetLongA -- Gets long integer value(s) out of an array

#### SYNOPSIS

```

success = iniGetLongA( ContextStr, ContextItemLine, Array, Entries );
D0                A0                A1                A2                D0

```

```

BOOL iniGetLongA( struct iniContext *, struct iniContextItemLine *,
    LONG *, ULONG );

```

#### FUNCTION

Reads one or more long integer value(s) out of the given context item line

#### INPUTS

ContextStr - Context structure where long integer values lie in  
ContextItemLine - Context item line where to extract long integers  
Array - The array where to store the long integer values  
Entries - Number of array entries. If the array in the INI file is bigger, the remaining entries will be ignored.

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyArray[4] = {4096, 65536, 16777216, 2147483647};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))

```

```
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniGetLongA ( context, contextitem, MyArray,
              (sizeof (MyArray) / sizeof (LONG)) );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = -4096, -65536, -16777216, -2147483648

   then
   MyArray[4] = {-4096, -65536, -16777216, -2147483648};
   Entries which can't be evaluated are left unchanged.
*/
```

#### NOTES

This function is called from  
    iniReadLongA  
    ().

Array fields which can't be evaluated (e.g. bad syntax) are left unchanged. So it's good to fill the array with default values first.

#### BUGS

#### SEE ALSO

```
iniGetFloatA
(),
iniGetStrA
(),
iniGetLong
(),
iniPutLongA
(),

iniReadLongA
(),
iniWriteLongA
(), <libraries/ini_lib.h>
```

## 1.31 ini.library/iniGetNumArrays

NAME

iniGetNumArrays -- Gets the amount of array fields

### SYNOPSIS

```
Arrays = iniGetNumArrays( ContextStr, ContextItemLine );
D0                                A0                                A1

ULONG iniGetNumArrays( struct iniContext *,
                      struct iniContextItemLine *);
```

### FUNCTION

Returns the amount of array entries in the given context item array.

### INPUTS

ContextStr - Context structure where array lies in  
ContextItemLine - Context item line structure where array starts

### RESULT

Arrays - Number of arrays in the given context item line

### EXAMPLE

```
struct iniFile *ini;
WORD *Palette;
ULONG PaletteEntries;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "Screen" );

if (!( contextitem = FindContextItem ( context, "Palette" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "Palette" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

PaletteEntries = iniGetNumArrays ( context, contextitem )

Palette = (WORD *) AllocMem ( PaletteEntries * sizeof (WORD),
                             MEMF_CHIP|MEMF_PUBLIC|MEMF_CLEAR );

iniGetWordA ( context, contextitem, Palette, PaletteEntries );
```

## NOTES

This function usually is used for dynamic array fields.  
If an error occurs during evaluation, NULL is returned.

## BUGS

## SEE ALSO

```
iniGetLongA
(),
iniGetWordA
(),
iniGetByteA
(),
iniGetFloatA
(),

iniGetStrA
(), <libraries/ini_lib.h>
```

## 1.32 ini.library/iniGetStr

## NAME

iniGetStr -- Gets a string

## SYNOPSIS

```
String = iniGetStr( ContextStr, ContextItemLine, Default );
D0                A0                A1                A2
```

```
STRPTR iniGetStr( struct iniContext *, struct iniContextItemLine *,
STRPTR );
```

## FUNCTION

Reads a string out of the given context item line

## INPUTS

ContextStr - Context structure where string lies in  
ContextItemLine - Context item line where to extract string  
Default - Default value to take if it can't be evaluated

## RESULT

String - The string extracted out of the context item line

## EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
STRPTR MyStr;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

MyStr = iniGetStr ( context, contextitem, "MyString" );

puts ( MyStr );

iniFreeNameStr ( MyStr );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = Hello world!

   then MyStr will contain "Hello world!". If this context or context
   item is not available, MyStr will contain "MyString" (default)
   instead.
*/

```

#### NOTES

This function is called from  
     iniReadStr  
     ().

This function calls  
     iniGetContextItemData  
     () with no buffer, this  
 means that the string returned must be deallocated with

```

iniFreeNameStr
()

```

#### BUGS

#### SEE ALSO

---

```

iniGetContextItemData
(),
iniGetLongA
(),
iniGetFloatA
(), iniGetStr(),

iniPutStr
(),
iniReadStrA
(),
iniWriteStrA
(), <libraries/ini_lib.h>

```

### 1.33 ini.library/iniGetStrA

#### NAME

iniGetStrA -- Extracts strings out of an array

#### SYNOPSIS

```

success = iniGetStrA( ContextStr, ContextItemLine, Array, Entries );
D0                A0                A1                A2                D0

```

```

BOOL iniGetStrA( struct iniContext *, struct iniContextItemLine *,
STRPTR *, ULONG );

```

#### FUNCTION

Reads one or more strings out of the given context item line

#### INPUTS

ContextStr - Context structure where string values lie in  
ContextItemLine - Context item line where to extract strings  
Array - The array where to store the pointers to the strings  
Entries - Number of array entries. If the array in the INI file is  
bigger, the remaining entries will be ignored.

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
STRPTR MyStr[4] = {NULL, NULL, NULL, NULL};

MyStr[0] = iniAllocNameStr ( "String 1" );

```

```

MyStr[1] = iniAllocNameStr ( "String 2" );
MyStr[2] = iniAllocNameStr ( "String 3" );
MyStr[3] = iniAllocNameStr ( "String 4" );

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniGetStrA ( context, contextitem, MyStr,
            (sizeof (MyStr) / sizeof (STRPTR)) );

printf ( "%s, %s, %s, %s\n", MyStr[0], MyStr[1], MyStr[2], MyStr[3]);

iniFreeNameStr ( MyStr[0] );
iniFreeNameStr ( MyStr[1] );
iniFreeNameStr ( MyStr[2] );
iniFreeNameStr ( MyStr[3] );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = Hello 1, Hello 2, Hello 3, Hello 4

   then
   MyStr[4] = {"Hello 1", "Hello 2", "Hello 3", "Hello 4"};
   Entries which can't be evaluated are left unchanged.
*/

```

## NOTES

This function is called from  
     iniReadStrA  
     ().

This function calls  
     iniGetContextItemData  
     () with no buffer.

Array fields which can't be evaluated (e.g. bad syntax) are left unchanged. So it's good to fill the array with default strings first. All entries of the array must be deallocated with  
     iniFreeNameStr  
     ()

when the strings are no longer of use. This means that the default entries of the array must be

```
iniAllocNameStr  
() strings!
```

BUGS

SEE ALSO

```
iniGetLongA  
(),  
iniGetFloatA  
(),  
iniGetStr  
(),  
iniPutStrA  
(),  
  
iniReadStrA  
(),  
iniWriteStrA  
(), <libraries/ini_lib.h>
```

## 1.34 ini.library/iniGetWordA

NAME

iniGetWordA -- reads a context item array into a (U)WORD array.

SYNOPSIS

```
success = iniGetWordA( ContextStr, ContextItemLine, Array, Entries );  
D0                A0                A1                A2                D0
```

```
BOOL iniGetWordA( struct iniContext *, struct iniContextItemLine *,  
WORD *, ULONG );
```

FUNCTION

Reads a context item array and stores the read words into a (U)WORD table you specified.

INPUTS

ContextStr - The context structure where the context line is in  
ContextItemLine - The context item line where the array is  
Array - An (U)WORD array where to store the values  
Entries - Number of entries to read (further entries will be ignored)

RESULT

success - TRUE if line could be evaluated else FALSE

---



## EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
WORD MyArray[4] = {16, 64, 256, 4096};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniGetWordA ( context, contextitem, MyArray,
             (sizeof (MyArray) / sizeof (WORD)) );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 10000, 1000, 10, 1

   then
   MyArray[4] = {10000, 1000, 10, 1};
   Entries which can't be evaluated are left unchanged.
*/
```

## NOTES

Make sure that the given array is big enough to hold all values or some memory area may be overwritten.

Fields which can't be evaluated are left unchanged.

## BUGS

## SEE ALSO

```
iniGetByteA
(),
iniGetLongA
(),
iniReadWordA
(),
```

```
        iniPutWordA  
        (),  
<libraries/ini_lib.h>
```

## 1.35 ini.library/iniInsertContext

### NAME

iniInsertContext -- inserts a context in an INI file structure

### SYNOPSIS

```
iniInsertContext( iniFile, ContextStr, PredContext );  
                  A0          A1          A2  
  
void iniInsertContext( struct iniFile *, struct iniContext *,  
                      struct iniContext *);
```

### FUNCTION

Inserts a context in an INI file and all its associated lines.

### INPUTS

iniFile - INI file structure where to insert context structure  
ContextStr - The context structure to be inserted  
PredContext - Context structure of the structure where to insert it

### RESULT

### NOTES

### BUGS

### SEE ALSO

```
        iniFreeContext  
        (),  
        iniRemContext  
        (),  
        iniDeleteContext  
        (),  
<libraries/ini_lib.h>
```

## 1.36 ini.library/iniInsertContextItem

### NAME

iniInsertContextItem -- inserts a context item line in an INI file.

### SYNOPSIS

```
iniInsertContextItem( ContextStr, ContextItemLine, PredLine );
                    A0           A1           A2

void iniInsertContextItem( struct iniContext *,
                          struct iniContextItemLine *, struct iniContextItemLine *);
```

### FUNCTION

Inserts a context item line in an INI context.

### INPUTS

ContextStr - The pointer of the context structure where to insert item  
ContextItemLine - The pointer of the context item line to be inserted.  
PredLine - The context item line where to insert it.

### RESULT

### NOTES

### BUGS

### SEE ALSO

```
iniFreeContextItem
(),
iniRemContextItem
(),
iniDeleteContextItem
(),
<libraries/ini_lib.h>
```

## 1.37 ini.library/iniIntToStr

### NAME

iniIntToStr -- Converts an integer value to a string.

### SYNOPSIS

```
string = iniIntToStr( Buffer, Integer, Format, Len, ZeroSep );
D0           A0           D0           D1           D2           D3:8
```

```
STRPTR iniIntToStr( STRPTR, ULONG, ULONG, ULONG, UBYTE );
```

#### FUNCTION

This function is used to convert an integer value to a standard ASCII string.

#### INPUTS

Buffer - A pointer to a buffer or NULL to create a new one.  
The buffer must be large enough to hold all values.  
Integer - Integer value to convert.  
Format - Format of the outputted string. Can be any of:  
INI\_FORMAT\_DEC - Use decimal with no precedor  
INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
INI\_FORMAT\_BIN - Use binary with % precedor  
INI\_FORMAT\_OCT - Use octal with & precedor  
INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
INI\_FORMAT\_YN - Use N for zero, Y for all others  
INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
INI\_UNSIGNED - Add this to indicate unsigned integer  
Len - Forced length of outputted string or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

string - Pointer to the string where the converted string is stored.

#### EXAMPLE

```
STRPTR Buffer;  
  
Buffer = iniIntToStr ( 0x4000, INI_FORMAT_DEC_CHAR, 7, '0');  
  
/* Buffer will contain: "#0016384" */  
  
puts ( Buffer );  
  
iniFreeNameStr ( Buffer );
```

#### NOTES

#### BUGS

The buffer is not checked for overflow. That means, IntLen should be lesser than your buffer.

#### SEE ALSO

```
iniStrToInt  
( ),
```

```
iniStrToFloat
(),
iniFloatToStr
(), <libraries/ini_lib.h>
```

## 1.38 ini.library/iniOpenDefault

NAME

iniOpenDefault -- Opens INI file for read access

SYNOPSIS

```
iniFile = iniOpenDefault( address, name, len );
D0                A0                A1                D0
```

```
struct iniFile *iniOpenDefault( APTR, STRPTR name, ULONG );
```

FUNCTION

Opens an INI file for read access and creates a iniFile structure for it. If the file doesn't exist, a default file will be created.

INPUTS

address - Address where default INI file lies (in memory)  
name - File name of the INI file to be accessed  
len - Length of the default INI file

RESULT

iniFile - A valid INI file structure ready to be evaluated.

EXAMPLE

```
char DefaultINI[]~= "/* Default INI file settings */\n\
  [MyContext]\n\
  MyItem = 5\n";
struct iniFile *ini;
ULONG MyValue;

/* Now let's open the INI file and create, if necessary an default
   ini file */

ini = iniOpenDefault ( DefaultINI, "ENVARC:MyPrefs.INI",
                      sizeof (DefaultINI));

MyValue = iniReadLong ( ini, "MyContext", "MyItem", 5L, 0L );

printf ( "%ld\n", MyValue );

iniClose ( ini );
```

## NOTES

The default file will only be created, if the Open() fails with an ERROR\_OBJECT\_NOT\_FOUND (code 205) error.  
If the default file can't be created (disk full, etc.) the function will use the default file in memory.

## BUGS

## SEE ALSO

```
    iniOpenFile
    (),
    iniOpenMem
    (),
    iniClose
    (),
    iniSaveFile
    (),
<libraries/ini_lib.h>
```

## 1.39 ini.library/iniOpenFile

## NAME

iniOpenFile -- Prepares an INI file for~context access

## SYNOPSIS

```
iniFile = iniOpenFile( name, accessMode );
D0          D1    D2

struct iniFile *iniOpenFile( STRPTR name, LONG );
```

## FUNCTION

Opens an INI file for read access and prepares an iniFile structure for evaluation. After this the INI file contents can be evaluated

## INPUTS

name - Name of the INI file to be opened.  
accessMode - Read mode of file (see <libraries/dos.h> for details)

## RESULT

iniFile - An INI file structure which is ready for evaluation

## EXAMPLE

```
struct iniFile *ini;
```

---

```
ULONG MyValue;

ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

MyValue = iniReadLong ( ini, "MyContext", "MyItem", 5L, 0L );

printf ( "%ld\n", MyValue );

iniClose ( ini );
```

## NOTES

## BUGS

## SEE ALSO

```
iniOpenDefault
(),
iniOpenFromFH
(),
iniOpenMem
(),
iniClose
(),

iniSaveFile
(), <libraries/ini_lib.h>, <libraries/dos.h>
```

## 1.40 ini.library/iniOpenFromFH

## NAME

iniOpenFromFH -- initializes an INI file from an already open file

## SYNOPSIS

```
iniFile = iniOpenFromFH( fh, len );
D0                D1 D2

struct iniFile *iniOpenFromFH( BPTR, ULONG );
```

## FUNCTION

Reads the INI data from an already open file and initializes the iniFile structure.

## INPUTS

fh - BPTR to an file handle of the already opened file to be read.  
len - Length of file (or length of bytes to read at maximum)

---

## RESULT

iniFile - An initialized INI file structure ready for evaluation

## EXAMPLE

```
struct iniFile *ini;
BPTR fh;
ULONG MyValue;

fh = Open ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );
ini = iniOpenFromFH ( fh, -1 );

MyValue = iniReadLong ( ini, "MyContext", "MyItem", 5L, 0L );

printf ( "%ld\n", MyValue );

iniClose ( ini );
```

## NOTES

## BUGS

## SEE ALSO

```
iniOpenDefault
(),
iniOpenFile
(),
iniOpenMem
(),
iniClose
(),

iniSaveFile
(),
iniSaveToFH
(), <libraries/ini_lib.h>
```

## 1.41 ini.library/iniOpenMem

## NAME

iniOpenMem -- Initializes a INI file structure from an INI file already in memory.

## SYNOPSIS

```
iniFile = iniOpenMem( address, len );
```

---



```
D0                A0                D0

struct iniFile *iniOpenMem( APTR, ULONG );
```

**FUNCTION**

Initializes an INI file structure from an INI file already in memory.

**INPUTS**

address - Address where the INI file lies  
len - Length of INI file in memory

**RESULT**

iniFile - Valid initialized INI file structure ready to be evaluated

**NOTES**

Used internally. Comes also in handy when you're going to read the file by yourself when it's crunched (so you can read XPK packed INIs).

**BUGS****SEE ALSO**

```
iniOpenDefault
(),
iniOpenFile
(),
iniOpenFromFH
(),
iniClose
(),

iniSaveFile
(),
iniSaveToFH
(), <libraries/ini_lib.h>
```

## 1.42 ini.library/iniPutByteA

**NAME**

iniPutByteA -- writes an (U)BYTE array into an context item array.

**SYNOPSIS**

```
success = iniPutByteA( ContextStr, ContextItemLine, Array, Entries,
D0                A0                A1                A2                D0
```

```
Format, Len, ZeroSep );
D1      D2      D3:8
```

```
BOOL iniPutByteA( struct iniContext *, struct iniContextItemLine *,
    BYTE *, ULONG, ULONG, ULONG, UBYTE );
```

#### FUNCTION

Writes the values of the given (U)BYTE table to the specified context item array.

#### INPUTS

ContextStr - The context structure where context line should be put  
ContextItemLine - The context item line where the array is  
Array - An (U)BYTE array where to take the values from  
Entries - Number of entries to write  
Format - Format of array entries to write out:  
INI\_FORMAT\_DEC - Use decimal with no precedor  
INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
INI\_FORMAT\_BIN - Use binary with % precedor  
INI\_FORMAT\_OCT - Use octal with & precedor  
INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
INI\_FORMAT\_YN - Use N for zero, Y for all others  
INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
INI\_UNSIGNED - Add this to indicate unsigned integer  
Len - Forced length of outputted string or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if line could be written else FALSE

#### EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
BYTE MyArray[4] = {-2, -1, 0, 1};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }
}
```

```

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniPutByteA ( context, contextitem, MyArray, sizeof (MyArray),
             INI_FORMAT_DEC, 3L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 25, 50, 75, 100

   then it will become:
   [MyContext]
   MyItem = -002, -001, 000, 001

   Entries which can't be stored are left unchanged.
*/

```

#### NOTES

This function is currently relatively slow. Especially with arrays with more than 16 entries.

#### BUGS

#### SEE ALSO

```

    iniPutWordA
    (),
    iniPutLongA
    (),
    iniWriteByteA
    (),
    iniGetByteA
    (),
<libraries/ini_lib.h>

```

## 1.43 ini.library/iniPutFloat

#### NAME

iniPutFloat -- Puts a quick floating point value into given item line

#### SYNOPSIS

```

success = iniPutFloat( ContextStr, ContextItemLine, Value,
D0                A0                A1                D0
                    Format, Len, ZeroSep );
D1                D2                D3:8

```

```

    BOOL iniPutFloat( struct iniContext *, struct iniContextItemLine *,
        LONG, ULONG, UBYTE );

```

## FUNCTION

Writes a quick float value into the given context item line

## INPUTS

ContextStr - Context structure where quick float value should be put  
 ContextItemLine - Context item line where to store quick float  
 Value - Value to be written  
 FltFormat - Format of the floating point value. Can be any of:  
     INI\_FLOAT\_FORMAT\_DEC - Use decimal with point separator  
     INI\_FLOAT\_UNSIGNED - Add this to indicate unsigned quick float  
 IntLen - Forced length of integer part or NULL for no force.  
 FracLen - Forced length of fractional part or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

## RESULT

success - TRUE if successful write else FALSE

## EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniPutFloat ( context, contextitem, 0x28000, INI_FLOAT_FORMAT_DEC,
    0L, 3L, ' ' );

/* After this, ENVARC:MyPrefs.INI will contain:
    [MyContext]
    MyItem = 2.500

    If the context or the context item doesn't exist, the value won't
    be written.

```

---

\*/

#### NOTES

This function is called from  
 iniWriteFloat  
 ().

#### BUGS

#### SEE ALSO

```

iniPutLong
(),
iniPutStr
(),
iniPutFloatA
(),
iniGetFloat
(),

iniWriteFloat
(),
iniReadFloat
(), <libraries/ini_lib.h>

```

## 1.44 ini.library/iniPutFloatA

#### NAME

iniPutFloatA -- Puts quick floating point value(s) into item line(s)

#### SYNOPSIS

```

success = iniPutFloatA( ContextStr, ContextItemLine, Array, Entries,
D0                A0          A1          A2          D0
                   FltFormat, IntLen,  FracLen, ZeroSep );
                   D1          D2          D3          D4:8

```

```

BOOL iniPutFloatA( struct iniContext *, struct iniContextItemLine *,
LONG *, ULONG, ULONG, ULONG, ULONG, UBYTE );

```

#### FUNCTION

Writes one or more quick float value(s) from an array into the given context item line

#### INPUTS

ContextStr - Context structure where quick float values should be put  
 ContextItemLine - Context item line where to store quick floats

Array - The array where to take the quick float values from  
 Entries - Number of array entries. If the array in the INI file is bigger, the remaining entries will be ignored.  
 FltFormat - Format of the floating point value. Can be any of:  
   INI\_FLOAT\_FORMAT\_DEC - Use decimal with point separator  
   INI\_FLOAT\_UNSIGNED - Add this to indicate unsigned quick float  
 IntLen - Forced length of integer part or NULL for no force.  
 FracLen - Forced length of fractional part or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyFloat[4] = {-0x10000, -0x8000, 0, 0x8000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
  /* If not, create it! */
  if (!( contextitem = CreateContextItem ( "MyItem" )))
  {
    puts ( "Couldn't create my context item !" );

    exit ( 20 );
  }

  /* Make context available for access */
  AddContextItem ( context, contextitem );
}

iniPutFloatA ( context, contextitem, MyFloat,
              (sizeof (MyFloat) / sizeof (LONG)),
              INI_FLOAT_FORMAT_DEC, 3L, 4L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = 13.5, 17.25, 1.116, 3.1416

  then it will become:
  [MyContext]
  MyItem = -001.0000, -000.5000, 000.0000, 000.5000

  Entries which can't be stored are left unchanged.
  */
```

#### NOTES

This function is called from

```
iniWriteFloatA
()
```

This function is currently relatively slow. Especially with arrays with more than 16 entries.

BUGS

SEE ALSO

```
iniPutLongA
(),
iniPutStrA
(),
iniPutFloat
(),
iniGetFloatA
(),

iniWriteFloatA
(),
iniReadFloatA
(), <libraries/ini_lib.h>
```

## 1.45 ini.library/iniPutLong

NAME

iniPutLong -- Puts a long integer value into the context item line

SYNOPSIS

```
success = iniPutLong( ContextStr, ContextItemLine, Value, Format,
D0                A0                A1                D0        D1
                    Len, ZeroSep );
D2        D3:8
```

```
BOOL iniPutLong( struct iniContext *, struct iniContextItemLine *,
LONG, ULONG, ULONG, UBYTE );
```

FUNCTION

Writes a long integer value into the given context item line

INPUTS

ContextStr - Context structure where the long integers should be put  
ContextItemLine - Context item line where to store long integer  
Value - Value to be written  
Format - Format of the outputted string. Can be any of:  
INI\_FORMAT\_DEC - Use decimal with no precursor

INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
 INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
 INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
 INI\_FORMAT\_BIN - Use binary with % precedor  
 INI\_FORMAT\_OCT - Use octal with & precedor  
 INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
 INI\_FORMAT\_YN - Use N for zero, Y for all others  
 INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
 INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
 INI\_UNSIGNED - Add this to indicate unsigned integer  
 Len - Forced length of outputted string or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if value could successfully be written or FALSE

#### EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniPutLong ( context, contextitem, 13750, INI_FORMAT_HEX_0X,
            8L, ' ' );

/* After this, ENVARC:MyPrefs.INI will contain:
   [MyContext]
   MyItem = 0x000035B6

   If the context or the context item doesn't exist, the value won't
   be written.
*/

```

#### NOTES

This function is called from  
     iniWriteLong



() .

BUGS

SEE ALSO

```

iniPutFloat
(),
iniPutStr
(),
iniPutLongA
(),
iniGetLong
(),

iniWriteLong
(),
iniReadLong
(), <libraries/ini_lib.h>

```

## 1.46 ini.library/iniPutLongA

NAME

iniPutLongA -- Puts long integer value(s) into context item line(s)

SYNOPSIS

```

success = iniPutLongA( ContextStr, ContextItemLine, Array, Entries,
D0                A0                A1                A2                D0
                    Format, Len, ZeroSep );
                    D1                D2                D3:8

```

```

BOOL iniPutLongA( struct iniContext *, struct iniContextItemLine *,
LONG *, ULONG, ULONG, ULONG, UBYTE );

```

FUNCTION

Writes one or more long integer value(s) from the given array into the specified context item line(s).

INPUTS

ContextStr - Context structure where the long integers should be put  
ContextItemLine - Context item line where to store long integers  
Array - The array where to take the long integer values from  
Entries - Number of array entries.  
Format - Format of the outputted string. Can be any of:  
INI\_FORMAT\_DEC - Use decimal with no precursor  
INI\_FORMAT\_DEC\_CHAR - Use decimal with # precursor  
INI\_FORMAT\_HEX - Use hexadecimal with \$ precursor

INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
 INI\_FORMAT\_BIN - Use binary with % precedor  
 INI\_FORMAT\_OCT - Use octal with & precedor  
 INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
 INI\_FORMAT\_YN - Use N for zero, Y for all others  
 INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
 INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
 INI\_UNSIGNED - Add this to indicate unsigned integer  
 Len - Forced length of outputted string or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
LONG MyArray[4] = {-200000, -100000, 0, 100000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
  /* If not, create it! */
  if (!( contextitem = CreateContextItem ( "MyItem" )))
  {
    puts ( "Couldn't create my context item !" );

    exit ( 20 );
  }

  /* Make context available for access */
  AddContextItem ( context, contextitem );
}

iniPutLongA ( context, contextitem, MyArray,
             (sizeof (MyArray) / sizeof (LONG)),
             INI_FORMAT_DEC, 0L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = 12345678, 76543210, 50000, -12345678

  then it will become:
  [MyContext]
  MyItem = -200000, -100000, 0, 100000

  Entries which can't be stored are left unchanged.
*/

```

## NOTES

This function is called from  
    iniWriteLongA  
    ().

This function is currently relatively slow. Especially with arrays with more than 16 entries.

## BUGS

## SEE ALSO

```
    iniPutFloatA  
    (),  
    iniPutStrA  
    (),  
    iniPutLong  
    (),  
    iniGetLongA  
    (),  
  
    iniWriteLongA  
    (),  
    iniReadLongA  
    (), <libraries/ini_lib.h>
```

## 1.47 ini.library/iniPutStr

## NAME

iniPutStr -- Puts a string into context item line

## SYNOPSIS

```
success = iniPutStr( ContextStr, ContextItemLine, String );  
D0                    A0                    A1                    A2
```

```
BOOL iniPutStr( struct iniContext *, struct iniContextItemLine *,  
                STRPTR );
```

## FUNCTION

Writes a string into the given context item line.

## INPUTS

ContextStr - Context structure where string should be put  
ContextItemLine - Context item line where to store string  
String - String to be written

## RESULT

success - TRUE if writing was successful else FALSE

#### EXAMPLE

```
struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyMessages.INI", MODE_OLDFILE );

context = FindContext ( ini, "Messages" );

if (!( contextitem = FindContextItem ( context, "Basty" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "Basty" )))
    {
        puts ( "Couldn't create my context item !" );

        exit ( 20 );
    }

    /* Make context available for access */
    AddContextItem ( context, contextitem );
}

iniPutStr ( context, contextitem, "I love Zuzana Burkertová !" );

/* After this, ENVARC:MyMessages.INI will contain:
   [Messages]
   Basty = I love Zuzana Burkertová !

   If the context or the context item doesn't exist, the value won't
   be written.
*/
```

#### NOTES

This function is called from  
iniReadStr  
().

#### BUGS

#### SEE ALSO

```
iniPutLongA
(),
iniPutFloatA
(), iniPutStr(),
iniGetStr
(),
iniWriteStrA
```

```

    ),
    iniReadStrA
    ), <libraries/ini_lib.h>

```

## 1.48 ini.library/iniPutStrA

NAME

iniPutStrA -- Stores array(s) of string into the context item line(s)

SYNOPSIS

```

success = iniPutStrA( ContextStr, ContextItemLine, Array, Entries );
D0                A0                A1                A2                D0

```

```

BOOL iniPutStrA( struct iniContext *, struct iniContextItemLine *,
    STRPTR *, ULONG );

```

FUNCTION

Writes one or more strings into the given context item line from an specified array.

INPUTS

ContextStr - Context structure where strings should be put  
ContextItemLine - Context item line where to store strings  
Array - The array where to take the pointers to the strings from  
Entries - Number of array entries. If the array in the INI file is bigger, the remaining entries will be ignored.

RESULT

success - TRUE if accessing was successful else NULL.

EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
STRPTR MyStr[4] = {"String 1", "String 2", "String 3", "String 4"};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))
    {
        puts ( "Couldn't create my context item !" );
    }
}

```

```
    exit ( 20 );
}

/* Make context available for access */
AddContextItem ( context, contextitem );
}

iniPutStrA ( context, contextitem, MyStr,
            (sizeof (MyStr) / sizeof (STRPTR)) );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = Hello 1, Hello 2, Hello 3, Hello 4

   then it will become:
   [MyContext]
   MyItem = String 1, String 2, String 3, String 4

   Entries which can't be written are left unchanged.
*/
```

#### NOTES

This function is called from  
iniWriteStrA  
().

#### BUGS

#### SEE ALSO

```
iniPutLongA
(),
iniPutFloatA
(),
iniPutStr
(),
iniGetStrA
(),

iniWriteStrA
(),
iniReadStrA
(), <libraries/ini_lib.h>
```

## 1.49 ini.library/iniPutWordA

#### NAME

iniPutWordA -- writes a (U)WORD array into a context item array.

---

## SYNOPSIS

```

success = iniPutWordA( ContextStr, ContextItemLine, Array, Entries,
D0                A0                A1                A2                D0
                    Format, Len, ZeroSep );
                    D1                D2                D3:8

```

```

BOOL iniPutWordA( struct iniContext *, struct iniContextItemLine *,
WORD *, ULONG, ULONG, ULONG, UBYTE );

```

## FUNCTION

Writes a context item array and stores the write words from a (U)WORD table you specified.

## INPUTS

ContextStr - The context structure where the context line is in  
ContextItemLine - The context item line where the array is  
Array - An (U)WORD array where to store the values  
Entries - Number of entries to read (further entries will be ignored)  
Format - Format of the outputted string. Can be any of:  
INI\_FORMAT\_DEC - Use decimal with no precedor  
INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
INI\_FORMAT\_BIN - Use binary with % precedor  
INI\_FORMAT\_OCT - Use octal with & precedor  
INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
INI\_FORMAT\_YN - Use N for zero, Y for all others  
INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
INI\_UNSIGNED - Add this to indicate unsigned integer  
Len - Forced length of outputted string or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

## RESULT

success - TRUE if accessing was successful else NULL.

## EXAMPLE

```

struct iniFile *ini;
struct iniContext *context;
struct iniContextItemLine *contextitem;
WORD MyArray[4] = {-2000, -1000, 0, 1000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

if (!( contextitem = FindContextItem ( context, "MyItem" )))
{
    /* If not, create it! */
    if (!( contextitem = CreateContextItem ( "MyItem" )))

```

```
{
    puts ( "Couldn't create my context item !" );

    exit ( 20 );
}

/* Make context available for access */
AddContextItem ( context, contextitem );
}

iniPutWordA ( context, contextitem, MyArray,
             (sizeof (MyArray) / sizeof (WORD)),
             INI_FORMAT_DEC_CHAR, 0L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 1000, 2000, 3000, -10000

   then it will become:
   [MyContext]
   MyItem = -#2000, -#1000, #0, #1000

   Entries which can't be stored are left unchanged.
*/
```

#### NOTES

This function is currently relatively slow. Especially with arrays with more than 16 entries.

#### BUGS

#### SEE ALSO

```
    iniPutByteA
    ( ),
    iniPutLongA
    ( ),
    iniWriteWordA
    ( ),
    iniReadWordA
    ( ),
<libraries/ini_lib.h>
```

## 1.50 ini.library/iniReadByteA

#### NAME

iniReadByteA -- reads a context item array into a (U)BYTE array.

---



## SYNOPSIS

```
success = iniReadByteA( iniFile, ContextName, ItemName, Array,
D0                A0      A1          A2      A3
                   Entries, Flags );
                   D0      D1
```

```
BOOL iniReadByteA( struct iniFile *, STRPTR, STRPTR, BYTE *,
                  ULONG, ULONG );
```

## FUNCTION

Searches a context item in a context you specified by name and stores the read bytes into a (U)BYTE table you specified.

## INPUTS

iniFile - INI file to be evaluated  
ContextName - Name of the context where context item is  
v32+: ContextName can be NULL. In this case all are searched  
ItemName - Name of the context item to be searched  
Array - An (U)BYTE array where to store the values  
Entries - Number of entries to read (further entries will be ignored)  
Flags - Search flags. They're currently defined as:  
INIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.

## RESULT

success - TRUE if line could be evaluated else FALSE

## EXAMPLE

```
struct iniFile *ini;
BYTE MyArray[4] = {-2, -1, -0, 1};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniReadByteA ( ini, "MyContext", "MyItem", MyArray,
              sizeof (MyArray), 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = 25, 50, 75, 100

  then
  MyArray[4] = {25, 50, 75, 100};
  Entries which can't be evaluated are left unchanged.
*/
```

## NOTES

Make sure that the given array is big enough to hold all values or some memory area may be overwritten.

---

Fields which can't be evaluated are left unchanged.

BUGS

SEE ALSO

```

    iniReadWordA
    (),
    iniReadLongA
    (),
    iniGetByteA
    (),
    iniWriteByteA
    (),
    <libraries/ini_lib.h>

```

## 1.51 ini.library/iniReadFloat

NAME

iniReadFloat -- Reads a quick floating point value

SYNOPSIS

```

QFloatValue = iniReadFloat( iniFile, ContextName, ItemName, Default,
D0                      A0      A1      A2      D0
                          Flags );
D1

```

```

LONG iniReadFloat( struct iniFile *, STRPTR, STRPTR, LONG, ULONG );

```

FUNCTION

Searches the INI file for the desired context and the desired context items and returns its quick floating point value.

INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Default - Default value to take if contents could not be evaluated
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

## RESULT

QFloatValue - The quick float value extracted out

## EXAMPLE

```
struct iniFile *ini;
LONG MyFloat;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

MyFloat = iniReadFloat ( ini, "MyContext", "MyItem", 0x28000, 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = 3.0

  then MyFloat will contain 0x30000. If this context or context item
  is not available, MyFloat will contain 0x28000 (default) instead.
  */
```

## NOTES

Only the first four fractional digits are evaluated. However, the 5th digit is evaluated for rounding purposes.

## BUGS

## SEE ALSO

```
    iniReadLong
    ( ),
    iniReadStr
    ( ),
    iniReadFloatA
    ( ),
    iniWriteFloat
    ( ),
    GetFloat ( ),
    iniPutFloat
    ( ), <libraries/ini_lib.h>
```

## 1.52 ini.library/iniReadFloatA

## NAME

iniReadFloatA -- Reads quick floating point value(s) into an array

## SYNOPSIS

```
success = iniReadFloatA( iniFile, ContextName, ItemName, Array,
```

```

D0                                A0      A1      A2      D0
                                Entries, Flags );
                                D1      D2

BOOL iniReadFloatA( struct iniFile *, STRPTR, STRPTR, LONG *,
                   ULONG, ULONG );

```

## FUNCTION

Searches the context given for the string given in context item and reads one or more quick float value(s) and stores them into the specified array.

## INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Default - Default value to take if contents could not be evaluated
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

## RESULT

success - TRUE if accessing was successful else NULL.

## EXAMPLE

```

struct iniFile *ini;
LONG MyFloat[4] = {-0x10000, -0x8000, 0, 0x8000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniReadFloatA ( ini, "MyContext", "MyItem", MyFloat,
                (sizeof (MyFloat) / sizeof (LONG)), 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 1.0, 1.5, 2.0, 2.5

   then
   MyFloat[4] = {0x10000, 0x18000, 0x20000, 0x28000};
   Entries which can't be evaluated are left unchanged.
*/

```

## NOTES

This function is currently relatively slow. Especially with arrays with more than 16 entries. Only the first four fractional digits are evaluated. However, the 5th digit is evaluated for rounding purposes. Array fields which can't be evaluated (e.g. bad syntax) are left

unchanged. So it's good to fill the array with default values first.

BUGS

SEE ALSO

```

iniWriteLongA
(),
iniWriteStrA
(),
iniReadFloat
(),
iniWriteFloatA
(),

iniGetFloatA
(),
iniPutFloatA
(), <libraries/ini_lib.h>

```

## 1.53 ini.library/iniReadLong

NAME

iniReadLong -- Reads a long integer value

SYNOPSIS

```

LongValue = iniReadLong( iniFile, ContextName, ItemName, Default,
D0                      A0      A1      A2      D0
                        Flags );
D1

```

```

LONG iniReadLong( struct iniFile *, STRPTR, STRPTR, LONG, ULONG );

```

FUNCTION

Searches the INI file for the desired context and the desired context items and returns its long integer value.

INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Default - Default value to take if contents could not be evaluated
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.

```

INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.

**RESULT**

LongValue - The long integer extracted out

**EXAMPLE**

```
struct iniFile *ini;
LONG MyLong;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

context = FindContext ( ini, "MyContext" );

MyLong = iniReadLong ( ini, "MyContext", "MyItem", 12345678, 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = -256

  then MyLong will contain -256. If this context or context item
  is not available, MyLong will contain 12345678 (default) instead.
*/
```

**NOTES****BUGS****SEE ALSO**

```
iniReadLong(),
  iniReadStr
  (),
  iniReadFloatA
  (),
  iniWriteFloat
  (),

  iniGetFloat
  (),
  iniPutFloat
  (), <libraries/ini_lib.h>
```

## 1.54 ini.library/iniReadLongA

**NAME**

iniReadLongA -- Reads long integer value(s) into an array

---

## SYNOPSIS

```

success = iniReadLongA( iniFile, ContextName, ItemName, Array,
D0                A0                A1                A2                D0
                    Entries, Flags );

```

```

BOOL iniReadLongA( struct iniFile *, STRPTR, STRPTR, LONG *,
LONG, ULONG );

```

## FUNCTION

Searches the context given for the string given in context item and reads one or more long integer value(s) and stores them into the specified array.

## INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Default - Default value to take if contents could not be evaluated
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

## RESULT

```

success - TRUE if accessing was successful else NULL.

```

## EXAMPLE

```

struct iniFile *ini;
LONG MyArray[4] = {4096, 65536, 16777216, 2147483647};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniReadLongA ( ini, "MyContext", "MyItem", MyArray,
                (sizeof (MyArray) / sizeof (LONG)), 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
[MyContext]
MyItem = -4096, -65536, -16777216, -2147483648

then
MyArray[4] = {-4096, -65536, -16777216, -2147483648};
Entries which can't be evaluated are left unchanged.
*/

```

## NOTES

Array fields which can't be evaluated (e.g. bad syntax) are left unchanged. So it's good to fill the array with default values first.

BUGS

SEE ALSO

```

iniPutLongA
(),
iniPutStrA
(),
iniGetFloat
(),
iniPutFloatA
(),

iniReadFloatA
(),
iniWriteFloatA
(), <libraries/ini_lib.h>

```

## 1.55 ini.library/iniReadStr

NAME

iniReadStr -- Gets a string

SYNOPSIS

```

String = iniReadStr( iniFile, ContextName, ItemName, Default,
D0                A0      A1      A2      A3
                  Flags );
D0

```

```

STRPTR iniReadStr( struct iniFile *, STRPTR, STRPTR, STRPTR, ULONG );

```

FUNCTION

Searches the given context item in the given context and returns, if found the context item data as a string.

INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Default - Default string to take if contents could not be evaluated
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```



## RESULT

String - The string value extracted out

## EXAMPLE

```
struct iniFile *ini;
STRPTR MyStr;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

MyStr = iniReadStr ( ini, "MyContext", "MyItem", "MyString", 0L );

puts ( MyStr );

iniFreeNameStr ( MyStr );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = Hello world!

   then MyStr will contain "Hello world!". If this context or context
   item is not available, MyStr will contain "MyString" (default)
   instead.
*/
```

## NOTES

The string returned must be deallocated with  
iniFreeNameStr  
( ) after  
use.

## BUGS

## SEE ALSO

```
iniReadLongA
(),
iniReadFloatA
(),
iniWriteStr
(),
iniGetStrA
(),

iniPutStrA
(), <libraries/ini_lib.h>
```

## 1.56 ini.library/iniReadStrA

NAME

iniReadStrA -- Extracts strings out of an array

### SYNOPSIS

```

success = iniReadStrA( iniFile, ContextName, ItemName, Array,
D0                A0      A1          A2      A3
                    Entries, Flags );
                    D0      D1

```

```

BOOL iniReadStrA( struct iniFile *, STRPTR, STRPTR, STRPTR *, ULONG,
                ULONG );

```

### FUNCTION

Searches for the given context item in the given context and reads the string(s) into the specified array.

### INPUTS

iniFile - INI file to be evaluated  
ContextName - Name of the context where context item is  
v32+: ContextName can be NULL. In this case all are searched  
ItemName - Name of the context item to be searched  
Array - Array where to put the pointers to the strings  
Flags - Search flags. They're currently defined as:  
INIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.

### RESULT

success - TRUE if accessing was successful else NULL.

### EXAMPLE

```

struct iniFile *ini;
STRPTR MyStr[4] = {NULL, NULL, NULL, NULL};

MyStr[0] = iniAllocNameStr ( "String 1" );
MyStr[1] = iniAllocNameStr ( "String 2" );
MyStr[2] = iniAllocNameStr ( "String 3" );
MyStr[3] = iniAllocNameStr ( "String 4" );

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniReadStrA ( ini, "MyContext", "MyItem", MyStr,
            (sizeof (MyStr) / sizeof (STRPTR)), 0L );

printf ( "%s, %s, %s, %s\n", MyStr[0], MyStr[1], MyStr[2], MyStr[3]);

iniFreeNameStr ( MyStr[0] );

```

```

iniFreeNameStr ( MyStr[1] );
iniFreeNameStr ( MyStr[2] );
iniFreeNameStr ( MyStr[3] );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = Hello 1, Hello 2, Hello 3, Hello 4

   then
   MyStr[4] = {"Hello 1", "Hello 2", "Hello 3", "Hello 4"};
   Entries which can't be evaluated are left unchanged.
*/

```

#### NOTES

Array fields which can't be evaluated (e.g. bad syntax) are left unchanged. So it's good to fill the array with default strings first. All array fields must be deallocated with

```

iniFreeNameStr
() when they

```

are not required anymore. This means that the default entries of the array must be

```

iniAllocNameStr
() strings!

```

#### BUGS

#### SEE ALSO

```

iniReadLongA
(),
iniReadFloatA
(),
iniReadStr
(),
iniWriteStrA
(),

iniGetStrA
(),
iniPutStrA
(), <libraries/ini_lib.h>

```

## 1.57 ini.library/iniReadWordA

NAME

iniReadWordA -- reads a context item array into a (U)WORD array.

#### SYNOPSIS

```

success = iniReadWordA( iniFile, ContextName, ItemName, Array,

```

---

```

D0                A0      A1          A2      A3
                  Entries, Flags );
                  D0      D1

```

```

BOOL iniReadWordA( struct iniFile *, STRPTR, STRPTR, WORD *,
                  ULONG, ULONG );

```

#### FUNCTION

Searches a context item in a context you specified by name and stores the read bytes into a (U)WORD table you specified.

#### INPUTS

```

iniFile - INI file to be evaluated
ContextName - Name of the context where context item is
    v32+: ContextName can be NULL. In this case all are searched
ItemName - Name of the context item to be searched
Array - An (U)WORD array where to store the values
Entries - Number of entries to read (further entries will be ignored)
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

#### RESULT

success - TRUE if line could be evaluated else FALSE

#### EXAMPLE

```

struct iniFile *ini;
WORD MyArray[4] = {16, 64, 256, 4096};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniReadWordA ( ini, "MyContext", "MyItem",
              (sizeof (MyArray) / sizeof (WORD)), 0L );

/* Let's say, ENVARC:MyPrefs.INI contains:
   [MyContext]
   MyItem = 10000, 1000, 10, 1

   then
   MyArray[4] = {10000, 1000, 10, 1};
   Entries which can't be evaluated are left unchanged.
*/

```

#### NOTES

Make sure that the given array is big enough to hold all values or some memory area may be overwritten.

Fields which can't be evaluated are left unchanged.

BUGS

SEE ALSO

```
    iniReadByteA
    ( ),
    iniReadLongA
    ( ),
    iniGetWordA
    ( ),
    iniWriteWordA
    ( ),
<libraries/ini_lib.h>
```

## 1.58 ini.library/iniRemContext

NAME

iniRemContext -- removes the last context from an INI file structure

SYNOPSIS

```
iniRemContext( iniFile );
               A0

void iniRemContext( struct iniFile *);
```

FUNCTION

Removes a previously generated and added context from a specified INI file structure. The entry removed is the last one.

INPUTS

iniFile - Pointer to INI structure where to remove from

NOTES

This function *\*DOESN'T\** do any deallocations. It just removes the node from the context list.

BUGS

SEE ALSO

```
    iniCreateContext
    ( ),
    iniFreeContext
    ( ),
```

---

```
iniAddContext
(),

iniInsertContext
(),
iniDeleteContext
(), <libraries/ini_lib.h>
```

## 1.59 ini.library/iniRemContextItem

### NAME

iniRemContextItem -- removes the last context item line from a context structure

### SYNOPSIS

```
iniRemContextItem( ContextStr );
                    A0

void iniRemContextItem( struct iniContext *);
```

### FUNCTION

Removes a previously generated context item line from a context structure. The context item line removed is the last one.

### INPUTS

ContextStr - Pointer to context structure where to remove from

### NOTES

This function just removes the node, it *\*DOESN'T\** deallocate any memory.

### BUGS

### SEE ALSO

```
iniCreateContextItem
(),
iniFreeContextItem
(), iniRemContextItem(),

iniInsertContextItem
(),
iniDeleteContextItem
(), <libraries/ini_lib.h>
```

## 1.60 ini.library/iniSaveFile

NAME

iniSaveFile -- Saves an .INI file from an INI structure to disk

SYNOPSIS

```
written = iniSaveFile( iniFile, name, accessMode );  
D0                A0                D1    D2
```

```
ULONG iniSaveFile( struct iniFile *, STRPTR, LONG );
```

FUNCTION

Saves an INI file to disk using the current INI structure. This function usually is called when the user selects 'Save' in an application.

INPUTS

iniFile - INI structure to be saved  
name - Name of the INI file to be created.  
accessMode - Write mode of file (see <libraries/dos.h> for details)

RESULT

written - Number of bytes written in total or -1 on error.

EXAMPLE

```
struct iniFile *ini;  
ULONG Length;  
  
/* Open old INI file */  
  
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );  
  
/* Write some value. Create contexts and/or  
context item if necessary */  
  
iniWriteLong ( ini, "MyContext", "MyItem", 5L, 0L );  
  
/* Now write back the INI file to disk */  
  
Length = iniSaveFile ( ini, "ENVARC:MyPrefs.INI", MODE_NEWFILE );  
  
iniClose ( ini );
```

NOTES

BUGS

---

SEE ALSO

```
iniOpen(),
    iniOpenDefault
    (),
    iniOpenFromFH
    (),
    iniOpenMem
    (),

iniClose
(), <libraries/ini_lib.h>, <libraries/dos.h>
```

## 1.61 ini.library/iniSaveToFH

NAME

iniSaveToFH -- Saves an INI structure to an already opened file

SYNOPSIS

```
written = iniSaveToFH( fh, iniFile );
D0                      A0  A1
```

```
ULONG iniSaveToFH( BPTR, struct iniFile *);
```

FUNCTION

Writes the INI data from the specified INI structure to the file already opened. The file won't be closed after writing, so you can add more data manually.

INPUTS

fh - BPTR to an file handle of the already opened file to be written.  
iniFile - INI structure to be saved

RESULT

written - Number of bytes written in total or -1 on error.

NOTES

The file is not closed after the data is written. Called from

```
iniSaveFile
() after opening the file.
```

BUGS

---



SEE ALSO

```
iniOpenDefault
(),
iniOpenFile
(),
iniOpenFromFH
(),
iniOpenMem
(),

iniClose
(),
iniSaveFile
(), <libraries/ini_lib.h>
```

## 1.62 ini.library/iniSetNameStr

NAME

iniSetNameStr -- sets a name string in a given structure offset

SYNOPSIS

```
namestring = iniSetNameStr( StructPos, namestring );
D0                                     A0
```

```
STRPTR iniSetNameStr( STRPTR *, STRPTR namestring );
```

FUNCTION

Stores a name string into an structure position. This is required if you use own strings in the library handlers.

INPUTS

StructPos - A memory pointer where the string pointer should be assigned.  
namestring - The already iniAllocNameStr()ed name string to be assigned.

RESULT

namestring - The name string stored or NULL on error.

EXAMPLE

```
STRPTR MyStr[4] = {NULL, NULL, NULL, NULL}
STRPTR TmpStr;

/* Fill in the buffers with some shit */

MyStr[0] = iniAllocNameStr ( "String 1" );
```



## FUNCTION

Allocates a name string out of a standard NULL-terminated C-String and assigns the allocated pointer to the structure position. This is required if you use own strings in the library handlers.

## INPUTS

StructPos - A memory pointer where the string pointer should be assigned.  
string - The C-String to be assigned.

## RESULT

namestring - The name string stored or NULL on error.

## EXAMPLE

```
STRPTR MyStr[4] = {NULL, NULL, NULL, NULL}

/* Fill in the buffers with some shit */

MyStr[0] = iniAllocNameStr ( "String 1" );
MyStr[1] = iniAllocNameStr ( "String 2" );
MyStr[2] = iniAllocNameStr ( "String 3" );
MyStr[3] = iniAllocNameStr ( "String 4" );

/* Now we have to change the strings for some reasons. We don't have
   to care about the old values, they're freed automatically. */

iniSetString ( (STRPTR *) &(MyStr[0]), "Changed 1" );
iniSetString ( (STRPTR *) &(MyStr[1]), "Changed 2" );
iniSetString ( (STRPTR *) &(MyStr[2]), "Changed 3" );
iniSetString ( (STRPTR *) &(MyStr[3]), "Changed 4" );
```

## NOTES

The namestring is a copy of string, but it's freed via  
iniFreePMem  
( )

## BUGS

## SEE ALSO

```
iniAllocNameStr
(),
iniFreeNameStr
(),
iniSetNameStr
()
```

## 1.64 ini.library/iniStrToFloat

NAME

iniStrToFloat -- Converts a string to a quick float value.

SYNOPSIS

```
QFloat = iniStrToFloat( String, Default );  
D0          A0          D0
```

```
LONG iniStrToFloat( STRPTR, LONG );
```

FUNCTION

This function is used to convert a standard ASCII string to a quick float value. The string may have signs (+/-) and a decimal point. A quick float value has in it's upper 16-bits the decimal value and in the lower 16-bits the fraction. That means, the highest possible accuracy is 1/65536. If the string can't be converted for any reason, the default value is used.

INPUTS

String - The string containing the quick float value.  
Default - Default quick float value to use if error.

RESULT

QFloat - The converted quick float value or the default value.

EXAMPLE

```
LONG QFloat;
```

```
QFloat = iniStrToFloat ( "3.14159", 0x10000 );
```

QFloat will be 0x3243F (3.1416). If an error would have occurred, QFloat would default to 0x10000 (1.0).

NOTES

The string's value may not exceed -65536/+65535 or an overflow error will occur. The value after the period will only be interpreted up to 4 digits. However, the 5th digit will be interpreted for rounding purposes.

BUGS

SEE ALSO

```
iniStrToInt  
(,  
iniIntToStr  
(,
```

```
iniFloatToStr  
( ), <libraries/ini_lib.h>
```

## 1.65 ini.library/iniStrToInt

### NAME

iniStrToInt -- Converts a string to an (un)signed 32-bit integer.

### SYNOPSIS

```
Integer = iniStrToInt( String, Default );  
D0                A0        D0
```

```
LONG iniStrToInt( STRPTR, LONG );
```

### FUNCTION

This function is used to convert a standard ASCII string to a standard 32-bit (un)signed integer value. The string may have signs (+/-) and can be in hexadecimal, decimal, binary and octal formats. Hexadecimal strings are preceded with \$ or 0x. Binary strings with %, octal strings with & and decimal strings start either with nothing or with a #. In addition to this, the following strings will return -1: Yes, Y, True, On  
0: No, N, False, Off

The match isn't case sensitive, so e.g. YES will also return -1. This makes evaluations easier of context items like:  
EnableFunction = Yes.

This means that iniReadLong, etc. automatically take care of this.

### INPUTS

String - The string containing the integer value.  
Default - Default integer value to use if error.

### RESULT

Integer - The converted integer value or the default value.

### EXAMPLE

```
LONG IntValue;
```

```
IntValue = iniStrToInt ( "%1111000011110000", 0x10000 );
```

IntValue will be 0xF0F0 (61680) after calling this function.  
If an error would have occurred during conversion, IntValue would default to 0x10000 (65536).

### NOTES

---

However, the string's value may not exceed -4,294,967,296 and +4,294,967,295 (32 bit limit) or an overflow error will occur. This function is used for all string to integer conversions.

BUGS

SEE ALSO

```
iniStrToFloat
(),
iniIntToStr
(),
iniFloatToStr
(), <libraries/ini_lib.h>
```

## 1.66 ini.library/iniWriteByteA

NAME

iniWriteByteA -- writes an (U)BYTE array into an context item array.

SYNOPSIS

```
success = iniWriteByteA( iniFile, ContextName, ItemName, Array,
D0                A0      A1      A2      A3
                   Entries, Flags, Format, Len, ZeroSep );
                   D0      D1      D2      D3      D4:8
```

```
BOOL iniWriteByteA( struct iniFile *, STRPTR, STRPTR, BYTE *,
                   ULONG, ULONG, ULONG, ULONG, UBYTE );
```

FUNCTION

Writes the values of the given (U)BYTE table to the specified context item in the given context.

INPUTS

iniFile - INI structure of the INI file which should be affected  
ContextName - The context name (C-String) in which context to store  
v32+: the context will be created if it's not existant  
ItemName - The context item name (C-String) in which the context item  
lies to write to.  
v32+: the context item will be created if it's not existant  
Array - An (U)BYTE array where to take the values from  
Entries - Number of entries to write  
Flags - Search flags. They're currently defined as:  
INIF\_ContextCase - Set this flag if the search of the context  
name should be case sensitive.  
INIF\_ContextItemCase - Set this flag if the search of the context  
item name should be case sensitive.

Format - Format of array entries to write out:  
 INI\_FORMAT\_DEC - Use decimal with no precedor  
 INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
 INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
 INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
 INI\_FORMAT\_BIN - Use binary with % precedor  
 INI\_FORMAT\_OCT - Use octal with & precedor  
 INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
 INI\_FORMAT\_YN - Use N for zero, Y for all others  
 INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
 INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
 INI\_UNSIGNED - Add this to indicate unsigned integer  
 Len - Forced length of outputted string or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if line could be written else FALSE

#### EXAMPLE

```
struct iniFile *ini;
BYTE MyArray[4] = {-2, -1, 0, 1};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteByteA ( ini, "MyContext", "MyItem", MyArray,
               sizeof (MyArray), INI_FORMAT_DEC, 3L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
  [MyContext]
  MyItem = 25, 50, 75, 100

  then it will become (even if MyContext or MyItem don't exist yet):
  [MyContext]
  MyItem = -002, -001, 000, 001

  Entries which can't be stored are left unchanged.
  */
```

#### NOTES

This function calls  
 iniPutByteA  
 () which is currently relatively slow.  
 Especially with arrays with more than 16 entries.

#### BUGS

#### SEE ALSO

```
iniWriteWordA
(),
iniWriteLongA
```

```

    (),
    iniPutByteA
    (),
    iniReadByteA
    (),
<libraries/ini_lib.h>

```

## 1.67 ini.library/iniWriteFloat

NAME

iniWriteFloat -- Writes a quick floating point value into given item line

SYNOPSIS

```

success = iniWriteFloat( iniFile, ContextName, ItemName, Value,
D0                A0      A1      A2      D0
                    Flags, FltFormat, IntLen, FracLen, ZeroSep );
                    D1      D2      D3      D4      D5:8

```

```

BOOL iniWriteFloat( struct iniFile *, STRPTR, STRPTR, LONG, ULONG,
    ULONG, ULONG, UBYTE );

```

FUNCTION

Writes a quick float value into the given context item, belonging to the specified context.

INPUTS

iniFile - INI structure of the INI file to be accessed  
ContextName - Name of the context where context item lies  
v32+: the context will be created if it's not existant  
ItemName - Name of the context item where context item line~lies  
v32+: the context item will be created if it's not existant  
Value - Quick float value to be written  
Flags - Search flags. They're currently defined as:  
INIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.  
FltFormat - Format of the floating point value. Can be any of:  
INI\_FLOAT\_FORMAT\_DEC - Use decimal with point separator  
INI\_FLOAT\_UNSIGNED - Add this to indicate unsigned quick float  
IntLen - Forced length of integer part or NULL for no force.  
FracLen - Forced length of fractional part or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

RESULT

success - TRUE if successful write else FALSE



## EXAMPLE

```
struct iniFile *ini;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteFloat ( ini, "MyContext", "MyItem", 0x28000,
                INI_FLOAT_FORMAT_DEC, 0L, 3L, ' ' );

/* After this, ENVARC:MyPrefs.INI will contain:
   [MyContext]
   MyItem = 2.500

   If the context or the context item doesn't exist, they will be
   created in order to be written.
*/
```

## NOTES

This function calls from  
iniPutFloat  
().

## BUGS

## SEE ALSO

```
iniWriteLong
(),
iniWriteStr
(),
iniWriteFloatA
(),
iniReadFloat
(),

iniPutFloat
(),
iniGetFloat
(), <libraries/ini_lib.h>
```

## 1.68 ini.library/iniWriteFloatA

NAME  
iniWriteFloatA -- Writes quick floating point value(s) into  
item line(s)

## SYNOPSIS

```

success = iniWriteFloatA( iniFile, ContextName, ItemName, Array,
D0          A0          A1          A2          A3
              Entries, Flags, FltFormat, IntLen, FracLen,
D0          D1          D2          D3          D4:8
              ZeroSep );
D5

```

```

BOOL iniWriteFloatA( struct iniFile *, STRPTR, STRPTR, LONG *,
                    ULONG, ULONG, ULONG, ULONG, ULONG, UBYTE );

```

#### FUNCTION

Writes one or more quick float value(s) from an array into the given context item, belonging to the specified context.

#### INPUTS

iniFile - INI structure where to assign write to  
ContextName - The name of the context to be affected  
v32+: the context will be created if it's not existant  
ItemName - The name of the context item where to write array to  
v32+: the context item will be created if it's not existant  
Array - The array where to write the quick float values to  
Entries - Number of array entries. If the array in the INI file is bigger, the remaining entries will be ignored.  
Flags - Search flags. They're currently defined as:  
INIIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
INIIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.  
FltFormat - Format of the floating point value. Can be any of:  
INI\_FLOAT\_FORMAT\_DEC - Use decimal with point separator  
INI\_FLOAT\_UNSIGNED - Add this to indicate unsigned quick float  
IntLen - Forced length of integer part or NULL for no force.  
FracLen - Forced length of fractional part or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```

struct iniFile *ini;
LONG MyFloat[4] = {-0x10000, -0x8000, 0, 0x8000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteFloatA ( ini, "MyContext", "MyItem", MyFloat,
                (sizeof (MyFloat) / sizeof (LONG)),
                INI_FLOAT_FORMAT_DEC, 3L, 4L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
[MyContext]
MyItem = 13.5, 17.25, 1.116, 3.1416

```

```

then it will become:
[MyContext]
MyItem = -001.0000, -000.5000, 000.0000, 000.5000

If the context or the context item do not exist, they will be
created in order to be written.
*/

```

## NOTES

This function calls  
`iniPutFloatA`  
`()` which is currently relatively slow.  
Especially with arrays with more than 16 entries.

## BUGS

## SEE ALSO

```

iniWriteLongA
(),
iniWriteStrA
(),
iniWriteFloat
(),
iniReadFloatA
(),

iniPutFloatA
(),
iniGetFloatA
(), <libraries/ini_lib.h>

```

## 1.69 ini.library/iniWriteLong

## NAME

`iniWriteLong` -- Writes a long integer value into the context item line

## SYNOPSIS

```

success = iniWriteLong( iniFile, ContextName, ItemName, Value,
D0                A0      A1      A2      D0
                    Flags, Format, Len, ZeroSep );
                    D1      D2      D3      D4:8

```

```

BOOL iniWriteLong( struct iniFile *, STRPTR, STRPTR, LONG, ULONG,
ULONG, UBYTE );

```

## FUNCTION

Writes a long integer value into the specified context item line, belonging to the given context.

#### INPUTS

iniFile - INI structure of the INI file which should be affected  
ContextName - Name of the context where the value should be stored  
v32+: the context will be created if it's not existant  
ItemName - Name of the context item where to store value to  
v32+: the context item will be created if it's not existant  
Value - Value to be written  
Flags - Search flags. They're currently defined as:  
INIIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
INIIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.  
Format - Format of the outputted string. Can be any of:  
INI\_FORMAT\_DEC - Use decimal with no precedor  
INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
INI\_FORMAT\_BIN - Use binary with % precedor  
INI\_FORMAT\_OCT - Use octal with & precedor  
INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
INI\_FORMAT\_YN - Use N for zero, Y for all others  
INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
INI\_UNSIGNED - Add this to indicate unsigned integer  
Len - Forced length of outputted string or NULL for no force.  
ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if value could successfully be written or FALSE

#### EXAMPLE

```
struct iniFile *ini;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteLong ( ini, "MyContext", "MyItem", 13750,
              INI_FORMAT_HEX_0X, 8L, ' ' );

/* After this, ENVARC:MyPrefs.INI will contain:
[MyContext]
MyItem = 0x000035B6

If the context or the context item do not exist, they will be
created in order to be written.
*/
```

#### NOTES

This function calls  
iniPutLong

() .

BUGS

SEE ALSO

```

iniWriteFloat
(),
iniWriteStr
(),
iniWriteLongA
(),
iniReadLong
(),

iniPutLong
(),
iniGetLong
(), <libraries/ini_lib.h>

```

## 1.70 ini.library/iniWriteLongA

NAME

iniWriteLongA -- writes an (U)LONG array into an context item array.

SYNOPSIS

```

success = iniWriteLongA( iniFile, ContextName, ItemName, Array,
D0                      A0      A1      A2      A3
                        Entries, Flags, Format, Len, ZeroSep );
                        D0      D1      D2      D3      D4:8

```

```

BOOL iniWriteLongA( struct iniFile *, STRPTR, STRPTR, LONG *,
                    ULONG, ULONG, ULONG, ULONG, UBYTE );

```

FUNCTION

Writes the values of the given (U)LONG table to the specified context item in the given context.

INPUTS

```

iniFile - INI structure of the INI file which should be affected
ContextName - The context name (C-String) in which context to store
v32+: the context will be created if it's not existant
ItemName - The context item name (C-String) in which the context item
lies to write to.
v32+: the context item will be created if it's not existant
Array - An (U)LONG array where to take the values from
Entries - Number of entries to write

```

Flags - Search flags. They're currently defined as:

- INIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.
- INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.

Format - Format of array entries to write out:

- INI\_FORMAT\_DEC - Use decimal with no precedor
- INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor
- INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor
- INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor
- INI\_FORMAT\_BIN - Use binary with % precedor
- INI\_FORMAT\_OCT - Use octal with & precedor
- INI\_FORMAT\_YESNO - Use No for zero, Yes for all others
- INI\_FORMAT\_YN - Use N for zero, Y for all others
- INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others
- INI\_FORMAT\_ONOFF - Use Off for zero, On for all others
- INI\_UNSIGNED - Add this to indicate unsigned integer

Len - Forced length of outputted string or NULL for no force.

ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if line could be written else FALSE

#### EXAMPLE

```
struct iniFile *ini;
LONG MyArray[4] = {-200000, -100000, 0, 100000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteLongA ( ini, "MyContext", "MyItem", MyArray,
               (sizeof (MyArray) / sizeof (LONG)),
               INI_FORMAT_DEC, 0L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
[MyContext]
MyItem = 12345678, 76543210, 50000, -12345678

then it will become:
[MyContext]
MyItem = -200000, -100000, 0, 100000

If the context or the context item do not exist, they will be
created in order to be written.
*/
```

#### NOTES

This function calls  
 iniPutLongA  
 () which is currently relatively slow.  
 Especially with arrays with more than 16 entries.

#### BUGS

SEE ALSO

```

        iniWriteByteA
        (),
        iniWriteWordA
        (),
        iniPutLongA
        (),
        iniReadLongA
        (),
<libraries/ini_lib.h>

```

## 1.71 ini.library/iniWriteStr

NAME

iniWriteStr -- Writes a string into a context item line

SYNOPSIS

```

success = iniWriteStr( iniFile, ContextName, ItemName, String,
D0                      A0          A1          A2          A3
                      Flags );
D0

```

```

BOOL iniWriteStr( struct iniFile *, STRPTR, STRPTR, STRPTR, ULONG );

```

FUNCTION

Writes a string into the given context item belonging to the specified context.

INPUTS

```

iniFile - INI structure of the INI file to be written
ContextName - Name of the context where the item lies
    v32+: the context will be created if it's not existant
ItemName - Name of the context item where string should be put
    v32+: the context item will be created if it's not existant
String - String to be written
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

RESULT

```

success - TRUE if writing was successful else FALSE

```

## EXAMPLE

```
struct iniFile *ini;

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyMessages.INI", MODE_OLDFILE );

iniWriteStr ( ini, "Messages", "Basty",
             "I love Zuzana Burkertová !" );

/* After this, ENVARC:MyMessages.INI will contain:
   [Messages]
   Basty = I love Zuzana Burkertová !

   If the context or the context item do not exist, they will be
   created in order to be written.
*/
```

## NOTES

This function calls  
iniPutStr  
().

## BUGS

## SEE ALSO

```
iniWriteLongA
(),
iniWriteFloatA
(), iniWriteStr(),
iniReadStr
(),

iniPutStrA
(),
iniGetStrA
(), <libraries/ini_lib.h>
```

## 1.72 ini.library/iniWriteStrA

## NAME

iniWriteStrA -- Writes an array of string(s) into the context item  
line(s)

## SYNOPSIS

```
success = iniWriteStrA( iniFile, ContextName, ItemName, Array,
D0                A0                A1                A2                A3
```



```

    Entries, Flags );
    D0      D1

```

```

BOOL iniWriteStrA( struct iniFile *, STRPTR, STRPTR, STRPTR *,
    ULONG, ULONG );

```

#### FUNCTION

Writes one or more strings into the given context item line, belonging to the specified context from an given array.

#### INPUTS

```

iniFile - INI file structure
ContextName - Name of the context where context item lies
    v32+: the context will be created if it's not existant
ItemName - Name of the context item to be accessed
    v32+: the context item will be created if it's not existant
Array - The array where to take the pointers of the strings from
Entries - Number of array entries. If the array in the INI file is
    bigger, the remaining entries will be ignored.
Flags - Search flags. They're currently defined as:
    INIF_ContextCase - Set this flag if the search of the context
        name should be case sensitive.
    INIF_ContextItemCase - Set this flag if the search of the context
        item name should be case sensitive.

```

#### RESULT

success - TRUE if accessing was successful else NULL.

#### EXAMPLE

```

struct iniFile *ini;
STRPTR MyStr[4] = {"String 1", "String 2", "String 3", "String 4"};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteStrA ( ini, "MyContext", "MyItem" MyStr,
    (sizeof (MyStr) / sizeof (STRPTR)) );

/* Let's say, ENVARC:MyPrefs.INI contains:
[MyContext]
MyItem = Hello 1, Hello 2, Hello 3, Hello 4

then it will become:
[MyContext]
MyItem = String 1, String 2, String 3, String 4

If the context or the context item do not exist, they will be
created in order to be written.
*/

```

#### NOTES

This function calls

---

```
iniPutStrA
().
```

BUGS

SEE ALSO

```
iniWriteLongA
(),
iniWriteFloatA
(),
iniWriteStr
(),
iniReadStrA
(),

iniPutStrA
(),
iniGetStrA
(), <libraries/ini_lib.h>
```

## 1.73 ini.library/iniWriteWordA

NAME

iniWriteWordA -- writes an (U)WORD array into an context item array.

SYNOPSIS

```
success = iniWriteWordA( iniFile, ContextName, ItemName, Array,
D0                      A0          A1          A2          A3
                        Entries, Flags, Format, Len, ZeroSep );
                        D0          D1          D2          D3          D4:8
```

```
BOOL iniWriteWordA( struct iniFile *, STRPTR, STRPTR, WORD *,
                    ULONG, ULONG, ULONG, ULONG, UBYTE );
```

FUNCTION

Writes the values of the given (U)WORD table to the specified context item in the given context.

INPUTS

iniFile - INI structure of the INI file which should be affected  
ContextName - The context name (C-String) in which context to store  
v32+: the context will be created if it's not existant  
ItemName - The context item name (C-String) in which the context item  
lies to write to.  
v32+: the context item will be created if it's not existant  
Array - An (U)WORD array where to take the values from

Entries - Number of entries to write  
 Flags - Search flags. They're currently defined as:  
     INIF\_ContextCase - Set this flag if the search of the context name should be case sensitive.  
     INIF\_ContextItemCase - Set this flag if the search of the context item name should be case sensitive.  
 Format - Format of array entries to write out:  
     INI\_FORMAT\_DEC - Use decimal with no precedor  
     INI\_FORMAT\_DEC\_CHAR - Use decimal with # precedor  
     INI\_FORMAT\_HEX - Use hexadecimal with \$ precedor  
     INI\_FORMAT\_HEX\_0X - Use hexadecimal with 0x precedor  
     INI\_FORMAT\_BIN - Use binary with % precedor  
     INI\_FORMAT\_OCT - Use octal with & precedor  
     INI\_FORMAT\_YESNO - Use No for zero, Yes for all others  
     INI\_FORMAT\_YN - Use N for zero, Y for all others  
     INI\_FORMAT\_TRUEFALSE - Use False for zero, True for all others  
     INI\_FORMAT\_ONOFF - Use Off for zero, On for all others  
     INI\_UNSIGNED - Add this to indicate unsigned integer  
 Len - Forced length of outputted string or NULL for no force.  
 ZeroSep - Zero character for IntLen leading zeroes. Usually " " or "0"

#### RESULT

success - TRUE if line could be written else FALSE

#### EXAMPLE

```

struct iniFile *ini;
WORD MyArray[4] = {-2000, -1000, 0, 1000};

/* Let's open an INI file */
ini = iniOpenFile ( "ENVARC:MyPrefs.INI", MODE_OLDFILE );

iniWriteWordA ( ini, "MyContext", "MyItem", MyArray,
               (sizeof (MyArray) / sizeof (WORD)),
               INI_FORMAT_DEC_CHAR, 0L, '0' );

/* Let's say, ENVARC:MyPrefs.INI contains:
[MyContext]
MyItem = 1000, 2000, 3000, -10000

then it will become:
[MyContext]
MyItem = -#2000, -#1000, #0, #1000

If the context or the context item do not exist, they will be
created in order to be written.
*/

```

#### NOTES

This function calls  
     iniPutWordA  
     () which is currently relatively slow.  
 Especially with arrays with more than 16 entries.

BUGS

SEE ALSO

```
    iniWriteByteA  
    ( ,  
    iniWriteLongA  
    ( ,  
    iniPutWordA  
    ( ,  
    iniReadWordA  
    ( ,  
<libraries/ini_lib.h>
```