

**utility**

|                      |
|----------------------|
| <b>COLLABORATORS</b> |
|----------------------|

|               |                           |                |                  |
|---------------|---------------------------|----------------|------------------|
|               | <i>TITLE :</i><br>utility |                |                  |
| <i>ACTION</i> | <i>NAME</i>               | <i>DATE</i>    | <i>SIGNATURE</i> |
| WRITTEN BY    |                           | March 28, 2025 |                  |

|                         |
|-------------------------|
| <b>REVISION HISTORY</b> |
|-------------------------|

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
|        |      |             |      |

# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>utility</b>                                 | <b>1</b> |
| 1.1      | utility.doc . . . . .                          | 1        |
| 1.2      | utility.library/AllocateTagItems . . . . .     | 1        |
| 1.3      | utility.library/Amiga2Date . . . . .           | 2        |
| 1.4      | utility.library/CallHookPkt . . . . .          | 2        |
| 1.5      | utility.library/CheckDate . . . . .            | 3        |
| 1.6      | utility.library/CloneTagItems . . . . .        | 4        |
| 1.7      | utility.library/Date2Amiga . . . . .           | 4        |
| 1.8      | utility.library/FilterTagChanges . . . . .     | 5        |
| 1.9      | utility.library/FilterTagItems . . . . .       | 6        |
| 1.10     | utility.library/FindTagItem . . . . .          | 7        |
| 1.11     | utility.library/FreeTagItems . . . . .         | 7        |
| 1.12     | utility.library/GetTagData . . . . .           | 8        |
| 1.13     | utility.library/MapTags . . . . .              | 9        |
| 1.14     | utility.library/NextTagItem . . . . .          | 10       |
| 1.15     | utility.library/PackBoolTags . . . . .         | 11       |
| 1.16     | utility.library/RefreshTagItemClones . . . . . | 13       |
| 1.17     | utility.library/SDivMod32 . . . . .            | 13       |
| 1.18     | utility.library/SMult32 . . . . .              | 14       |
| 1.19     | utility.library/Stricmp . . . . .              | 14       |
| 1.20     | utility.library/Strnicmp . . . . .             | 15       |
| 1.21     | utility.library/TagInArray . . . . .           | 16       |
| 1.22     | utility.library/ToLower . . . . .              | 17       |
| 1.23     | utility.library/ToUpper . . . . .              | 17       |
| 1.24     | utility.library/UDivMod32 . . . . .            | 18       |
| 1.25     | utility.library/UMult32 . . . . .              | 18       |

# Chapter 1

## utility

### 1.1 utility.doc

|                    |                        |              |
|--------------------|------------------------|--------------|
| AllocateTagItems() | FindTagItem()          | SMult32()    |
| Amiga2Date()       | FreeTagItems()         | Stricmp()    |
| CallHookPkt()      | GetTagData()           | Strnicmp()   |
| CheckDate()        | MapTags()              | TagInArray() |
| CloneTagItems()    | NextTagItem()          | ToLower()    |
| Date2Amiga()       | PackBoolTags()         | ToUpper()    |
| FilterTagChanges() | RefreshTagItemClones() | UDivMod32()  |
| FilterTagItems()   | SMod32()               | UMult32()    |

### 1.2 utility.library/AllocateTagItems

#### NAME

AllocateTagItems -- Allocate a TagItem array (or chain). (V36)

#### SYNOPSIS

```
tagList = AllocateTagItems( numItems )
D0                                     D0
```

```
struct TagItem *AllocateTagItems( ULONG numItems);
```

#### FUNCTION

Allocates the specified number of usable TagItems slots, and does so in a format that the function FreeTagItems can handle.

Note that to access the TagItems in 'tagList', you should use the function NextTagItem(). This will insure you respect any chaining (TAG\_MORE) that the list uses, and will skip any TAG\_IGNORE items that AllocateTagItems() might use to stash size and other information.

#### INPUTS

numItems - the number of TagItem slots you want to allocate.

#### RESULT

tagList - the allocated chain of TagItem structures. Will

```
return NULL if unsuccessful.
```

BUGS

SEE ALSO

```
FreeTagItems(), CloneTagItems()
```

### 1.3 utility.library/Amiga2Date

NAME

```
Amiga2Date -- Calculate the date from a timestamp. (V36)
```

SYNOPSIS

```
Amiga2Date( AmigaTime, Date )
            D0          A0
```

```
void Amiga2Date( ULONG, struct ClockData * );
```

FUNCTION

Fills in a ClockData structure with the date and time calculated from a ULONG containing the number of seconds from 01-Jan-1978 to the date.

INPUTS

AmigaTime - the number of seconds from 01-Jan-1978.

RESULTS

Date - filled in with the date/time specified by AmigaTime.

NOTES

SEE ALSO

```
CheckDate(), Date2Amiga()
```

BUGS

### 1.4 utility.library/CallHookPkt

NAME

```
CallHookPkt -- Invoke a Hook function callback. (V36)
```

SYNOPSIS

```
return = CallHookPkt( hook, object, paramPkt )
D0          A0      A2      A1
```

```
ULONG CallHookPkt( struct Hook *hook, VOID *object, VOID *paramPkt );
```

FUNCTION

Performs the callback standard defined by a Hook structure. This function is really very simple; it effectively performs a JMP to Hook->h\_Entry.

---

It is probably just as well to do this operation in an assembly language function linked in to your program, possibly from a compiler supplied library or a builtin function.

It is anticipated that C programs will often call a 'varargs' variant of this function which will be named CallHook. This function must be provided in a compiler specific library, but an example of use would be:

```
returnval = CallHook( hook, dataobject, COMMAND_ID, param1, param2 );
```

This function CallHook can be implemented in many C compilers like this:

```
CallHook( hook, object, command, ... )
struct Hook      *hook;
VOID             *object;
ULONG            command;
{
    return ( CallHookPkt( hook, object, (VOID *) &command ) );
}
```

#### INPUTS

|          |  |
|----------|--|
| Hook     | - pointer to Hook structure as defined in utility/hooks.h  |
| Object   | - useful data structure in the particular context the hook is being used for.  |
| ParamPkt | - pointer to a parameter packet (often built on the stack); by convention this packet should start off with a longword command code, and the remaining data in the packet depends on that command. |

#### RESULT

|        |   |
|--------|---|
| return | - The meaning of the value returned in D0 depends on the context in which the Hook is being used. |
|--------|---|

#### NOTES

The functions called through this function should follow normal register conventions unless EXPLICITLY documented otherwise (and they have a good reason too).

#### BUGS

#### SEE ALSO

utility/hooks.h

## 1.5 utility.library/CheckDate

#### NAME

CheckDate -- Checks ClockData struct for legal date. (V36)

#### SYNOPSIS

```
AmigaTime = CheckDate( Date )
D0                      A0
```

```
ULONG CheckDate( struct ClockData * );
```

#### FUNCTION

Determines if the Date is a legal date and returns the number of seconds to Date from 01-Jan-1978 if it is.

#### INPUTS

Date - pointer to a ClockData structure.

#### RESULTS

AmigaTime - 0 if Date invalid; otherwise, the number of seconds to Date from 01-Jan-1978.

#### NOTES

#### BUGS

The wday field of the ClockData structure is not checked.

#### SEE ALSO

Amiga2Date(), Date2Amiga()

## 1.6 utility.library/CloneTagItems

#### NAME

CloneTagItems -- Copies a TagItem list. (V36)

#### SYNOPSIS

```
newTagList = CloneTagItems( tagList )
D0          A0
```

```
struct TagItem *CloneTagItems( struct TagItem *tagList );
```

#### FUNCTION

Copies the essential contents of a tagItem list. Internally, it uses AllocateTagItems() so that you can use FreeTagItems().

#### INPUTS

tagList - TagItem list to clone.

#### RESULT

newTagList - resultant copy.

#### BUGS

#### SEE ALSO

AllocateTagItems(), FreeTagItems(), RefreshTagItemClones()

## 1.7 utility.library/Date2Amiga

#### NAME

Date2Amiga -- Calculate seconds from 01-Jan-1978. (V36)

---

## SYNOPSIS

```
AmigaTime = Date2Amiga( Date )
D0          A0
```

```
ULONG Date2Amiga( struct ClockData * );
```

## FUNCTION

Calculates the number of seconds from 01-Jan-1978 to the date specified in the ClockData structure.

## INPUTS

Date - pointer to a ClockData structure containing the date of interest.

## RESULTS

AmigaTime - the number of seconds from 01-Jan-1978 to the date specified in Date.

## NOTES

This function does no sanity checking of the data in Date.

## SEE ALSO

Amiga2Date(), CheckDate()

## BUGS

## 1.8 utility.library/FilterTagChanges

## NAME

FilterTagChanges -- Eliminate TagItems which specify no change. (V36)

## SYNOPSIS

```
FilterTagChanges( changeList, oldValues, apply)
                  A0          A1          D0
```

```
void FilterTagChanges( struct TagItem *changeList,
                      struct TagItem *oldValues, LONG apply );
```

## FUNCTION

Eliminate items from a "change list" that specify values already in effect in existing list. Optionally update the existing list if the Boolean 'Apply' is true.

The elimination is done by changing the ti\_Tag field to TAG\_IGNORE. So, this function may change the input tag list(s).

## INPUTS

changeList - specification of new tag-value pairs.  
 oldValues - a list of existing tag item pairs.  
 apply - Boolean specification as to whether the values in oldValues are to be updated to the values in changeList.

## RESULT

None.



**EXAMPLE**

Assume you have an attribute list for an object (oldValues) which looks like this:

```
ATTR_Size,  "large",
ATTR_Color, "orange",
ATTR_Shape, "square",
```

If you receive a new TagList containing some changes (changeList), which looks like this:

```
ATTR_Size,  "large",
ATTR_Shape, "triangle"
```

If you call FilterTagChanges(), changeList will be modified to contain only those attributes which are different from the oldValues. All other tagitems will have their tag-values set to TAG\_IGNORE. The resulting changeList will become:

```
TAG_IGNORE, "large",
ATTR_Shape, "triangle"
```

If apply was set to TRUE, oldValues would be:

```
ATTR_Size,  "large"
ATTR_Color, "orange"
ATTR_Shape, "triangle"
```

**BUGS****SEE ALSO****1.9 utility.library/FilterTagItems****NAME**

FilterTagItems - Remove selected items from a TagItem list. (V36)

**SYNOPSIS**

```
nvalid = FilterTagItems(tagList, tagArray, logic)
D0                      A0          A1          D0
```

```
ULONG FilterTagItems(struct TagItem *tagList, Tag *tagArray,
                     LONG logic);
```

**FUNCTION**

Removes TagItems from a TagItem list (by changing ti\_Tag to TAG\_IGNORE) depending on whether its ti\_Tag value is found in an array of TagValues.

If the 'logic' parameter is TAGFILTER\_AND, then all items not appearing in the list are excluded.

If 'logic' is TAGFILTER\_NOT, then items not found in the array are preserved, and the ones in the array are cast out.

## INPUTS

tagList            - input list of tag items which is to be filtered  
                    by having selected items changed to TAG\_IGNORE.  
tagArray           - an array of Tag values, terminated by TAG\_END,  
                    as specified in the notes on TagInArray().  
logic              - specification whether items in TagArray are to  
                    be included or excluded in the filtered result.

## RESULT

nvalid            - number of valid items left in resulting filtered  
                    list.

## BUGS

## SEE ALSO

TagInArray()

## 1.10 utility.library/FindTagItem

## NAME

FindTagItem -- Scans TagItem list for a Tag. (V36)

## SYNOPSIS

```
tag = FindTagItem( tagVal, tagList)
D0                    D0            A0
```

```
struct TagItem *FindTagItem( Tag tagVal, struct TagItem *tagList );
```

## FUNCTION

Scans a TagItem "List", which is in fact a chain of arrays  
of TagItem structures as defined in utility/tagitem.h.  
Returns a pointer to the FIRST item with ti\_Tag matching the  
'TagVal' parameter.

## INPUTS

tagVal            - Tag value to search for.  
tagList           - beginning of TagItem list to scan.

## RESULT

Returns a pointer to the item with ti\_Tag matching 'TagVal'.  
Returns NULL if there is no match or if TagList is NULL.

## BUGS

## SEE ALSO

utility/tagitem.h, GetTagData(), PackBoolTags(), NextTagItem()

## 1.11 utility.library/FreeTagItems

## NAME

FreeTagItems -- Frees allocated TagItem lists. (V36)

## SYNOPSIS

```
FreeTagItems( tagList )
              A0
```

```
void FreeTagItems( struct TagItem *tagList );
```

## FUNCTION

Frees the memory of a TagItem list allocated either by AllocateTagItems() or CloneTagItems().

## INPUTS

TagList                    - list to free. Must be created by functions specified. A value of NULL for 'tagList' is safe.

## RESULT

None.

## BUGS

## SEE ALSO

AllocateTagItems(), CloneTagItems()

## 1.12 utility.library/GetTagData

## NAME

GetTagData -- Obtain data corresponding to Tag. (V36)

## SYNOPSIS

```
value = GetTagData(tagVal, default, tagList)
D0                    D0            D1            A0
```

```
ULONG GetTagData(Tag TagVal, ULONG Default, struct TagItem *TagList)
```

## FUNCTION

Searches a TagItem list for a matching Tag value, and returns the corresponding ti\_Data value for the TagItem found. If none found, will return the value passed it as 'default'.

## INPUTS

tagVal                    - Tag value to search for.  
 default                   - value to be returned if tagVal is not found.  
 tagList                   - the TagItem list to search.

## RESULT

value                    - The ti\_Data value for first matching TagItem, or 'default' if a ti\_Tag matching 'Tag' is not found.

## BUGS

## SEE ALSO

utility/tagitem.h, FindTagItem(), PackBoolTags(), NextTagItem()

## 1.13 utility.library/MapTags

### NAME

MapTags -- Convert ti\_Tag values in a list via map pairing. (V36)

### SYNOPSIS

```
MapTags(tagList, mapList, includeMiss)
        A0          A1          D0
```

```
void MapTags(struct TagItem *tagList, struct TagItem mapList,
              LONG includeMiss);
```

### FUNCTION

Apply a "mapping list" mapList to tagList:

If the ti\_Tag field of an item in tagList appears as ti\_Tag in some item in mapList, overwrite ti\_Tag with the corresponding ti\_Data from the map list.

If a tag in tagList does not appear in the mapList, you can choose to have it removed by changing it to TAG\_IGNORE. Do this by setting includeMiss to FALSE.

If you want to have items which do not appear in the mapList survive the mapping as-is, set includeMiss to 1.

This is central to gadget interconnections where you want to convert the tag values from one space (the sender) to another (the receiver).

### INPUTS

|             |  |
|-------------|--|
| tagList     | - Input list of tag items which is to be mapped to Tag values as specified in mapList.   |
| mapList     | - a "mapping list" tagItem list which pairs Tag values expected to appear in tagList with new values to be substituted in the ti_Tag fields of tagList. May be NULL, which means that all items in tagList will be eliminated. |
| includeMiss | - 0 to remove tags from tagList not in mapList, 1 to remove  |

### RESULT

None.

### EXAMPLE

```
/* Consider this source list: */
struct TagItem list[] = {
    { MY_SIZE,      71 },
    { MY_WEIGHT,    200 },
    { TAG_END,      } };

/* And the mapping list: */
struct TagItem map[] = {
    { MY_SIZE,      HIS_TALL },
    { TAG_END,      } };
```

```

/* Then after MapTags( list, map, 0 ), 'list' will become: */
    { HIS_TALL, 71 },
    { TAG_IGNORE, },
    { TAG_END, }

/* Then after MapTags( list, map, 1 ), 'list' will become: */
    { HIS_TALL, 71 },
    { MY_WEIGHT, 200 },
    { TAG_END, }

```

#### NOTES

The procedure will change the values of the input tag list tagList (but not mapList).

You can "filter" a list by passing includeMiss as 0, and having the data items in the map list equal the corresponding tags.

You can perform the inverse filter ("everything but") by passing includeMiss equal to 1, and creating a map item for every tag you want to filter out, pairing it with a mapped data value of TAG\_IGNORE.

For safety and "order independence" of tag item arrays, if you attempt to map some tag to the value TAG\_END, the value TAG\_IGNORE will be substituted instead.

#### BUGS

#### SEE ALSO

## 1.14 utility.library/NextTagItem

#### NAME

NextTagItem -- Iterate TagItem lists. (V36)

#### SYNOPSIS

```

next_tag = NextTagItem( tagItemPtr )
D0                      A0

```

```

struct TagItem *NextTagItem( struct TagItem **tagItemPtr );

```

#### FUNCTION

Iterates through a (chained) array of TagItem structures, skipping and chaining as dictated by system tags. TAG\_SKIP will cause it to skip the entry and the next, TAG\_IGNORE ignores that single entry, and TAG\_MORE has a pointer to another array of tags (and terminates the current array!) TAG\_DONE also terminates the current array. Each call returns either the next tagitem you should examine, or NULL at the end.

#### INPUTS

```

tagItemPtr    - doubly-indirect reference to a TagItem structure.
                The pointer will be changed to keep track of the
                iteration.

```

## RESULT

```
next_tag      - Each TagItem in the array or chain of arrays that
                should be processed according to system Tag values
                (in utility/tagitem.h) is returned in turn with
                successive calls.
```

### EXAMPLE

```
Iterate( struct TagItem *tags );
{
    struct TagItem *tstate;
    struct TagItem *tag;

    tstate = tags;
    while ( tag = NextTagItem( &tstate ) )
    {
        switch ( tag->ti_Tag )
        {
            case TAG1:
                ...
                break;
            case TAG2:
                ...
                break;
            ...
        }
    }
}
```

## NOTES

Do NOT use the value of `*tagItemPtr`, but rather use the pointer returned by `NextTagItem()`.

BUGS

SEE ALSO

```
utility/tagitem.h, GetTagData(), PackBoolTags(), FindTagItem()
```

### 1.15 utility.library/PackBoolTags

NAME \_\_\_\_\_

PackBoolTags -- Builds a "Flag" word from a TagList. (V36)

## SYNOPSIS

```
boolflags = PackBoolTags( initialFlags, tagList, boolMap )
```

|    |    |    |    |
|----|----|----|----|
| D0 | D0 | A0 | A1 |
|----|----|----|----|

```

ULONG PackBoolTags( ULONG initialFlags, struct TagItem *tagList,
                    struct TagItem *boolMap );

```

FUNCTION

Picks out the Boolean TagItems in a TagItem list and converts them into bit-flag representations according to a correspondence defined by the TagItem list 'BoolMap.'

A Boolean TagItem is one where only the logical value of

the `ti_Data` is relevant. If this field is 0, the value is FALSE, otherwise TRUE.

#### INPUTS

- `initialFlags` - a starting set of bit-flags which will be changed by the processing of TRUE and FALSE Boolean tags in `tagList`.
- `tagList` - a `TagItem` list which may contain several `TagItems` defined to be "Boolean" by their presence in `boolMap`. The logical value of `ti_Data` determines whether a `TagItem` causes the bit-flag value related by `boolMap` to set or cleared in the returned flag longword.
- `boolMap` - a `TagItem` list defining the Boolean Tags to be recognized, and the bit (or bits) in the returned longword that are to be set or cleared when a Boolean Tag is found to be TRUE or FALSE in `tagList`.

#### RESULT

- `boolflags` - the accumulated longword of bit-flags, starting with `InitialFlags` and modified by each Boolean `TagItem` encountered.

#### EXAMPLE

```
/* define some nice user tag values ... */
enum mytags { tag1 = TAG_USER+1, tag2, tag3, tag4, tag5 };

/* this TagItem list defines the correspondence between Boolean tags
 * and bit-flag values.
 */
struct TagItem boolmap[] = {
    { tag1, 0x0001 },
    { tag2, 0x0002 },
    { tag3, 0x0004 },
    { tag4, 0x0008 },
    { TAG_DONE }
};

/* You are probably passed these by some client, and you want
 * to "collapse" the Boolean content into a single longword.
 */

struct TagItem boolexample[] = {
    { tag1, TRUE },
    { tag2, FALSE },
    { tag5, Irrelevant },
    { tag3, TRUE },
    { TAG_DONE }
};

/* Perhaps 'boolflags' already has a current value of 0x800002. */
boolflags = PackBoolTags( boolflags, boolexample, boolmap );

/* The resulting new value of 'boolflags' will be 0x800005. */
```

#### BUGS

There are some undefined cases if there is duplication of a given Tag in either list. It is probably safe to say that the \*last\* of identical Tags in TagList will hold sway.

SEE ALSO

utility/tagitem.h, GetTagData(), FindTagItem(), NextTagItem()

## 1.16 utility.library/RefreshTagItemClones

NAME

RefreshTagItemClones -- Rejuvenates a clone from the original. (V36)

SYNOPSIS

```
RefreshTagItemClones( cloneTagItems, originalTagItems )
                      A0                A1
```

```
void RefreshTagItemClones( struct TagItem *cloneTagItems,
                          struct TagItem *originalTagItems );
```

FUNCTION

If (and only if) the tag items 'cloneTagItems' were created from 'originalTagItems' by CloneTagItems(), and if originalTagItems has not been changed in any way, you can reset the clone list to its original state by using this function.

INPUTS

CloneTagItems - return value from CloneTagItems(originalTagItems).  
OriginalTagItems- a tag list that hasn't changed.

RESULT

None.

EXAMPLE

BUGS

SEE ALSO

CloneTagItems(), AllocateTagItems(), FreeTagItems()

## 1.17 utility.library/SDivMod32

NAME

SDivMod32 -- Signed 32 by 32 bit division and modulus. (V36)

SYNOPSIS

```
Quotient:Remainder = SDivMod32( Dividend, Divisor )
D0                D1                D0                D1
```

```
LONG SDivMod32( LONG, LONG );
```

FUNCTION

Divides the signed 32 bit dividend by the signed 32 bit divisor

---



and returns a signed 32 bit quotient and remainder.

#### INPUTS

Dividend            - signed 32 bit dividend.  
Divisor             - signed 32 bit divisor.

#### RESULTS

Quotient            - signed 32 quotient of the division.  
Remainder          - signed 32 remainder of the division.

#### NOTES

#### SEE ALSO

SMult32(), UDivMod32(), UMult32()

#### BUGS

## 1.18 utility.library/SMult32

#### NAME

SMult32 -- Signed 32 by 32 bit multiply with 32 bit result. (V36)

#### SYNOPSIS

```
Result = SMult32( Arg1, Arg2 )
D0      D0      D1
```

```
LONG SMult32( LONG, LONG );
```

#### FUNCTION

Returns the signed 32 bit result of multiplying Arg1 by Arg2.

#### INPUTS

Arg1, Arg2            - signed multiplicands.

#### RESULTS

Result                - the signed 32 bit result of multiplying  
                         Arg1 by Arg2.

#### NOTES

#### SEE ALSO

SDivMod32(), UDivMod32(), UMult32()

#### BUGS

## 1.19 utility.library/Stricmp

#### NAME

Stricmp -- Case-insensitive string compare. (V37)

#### SYNOPSIS

```
res = Stricmp(string1, string2)
```

---

D0                      A0              A1

LONG Stricmp(char \*, char \*);

#### FUNCTION

Stricmp compares two strings, ignoring case. It handles all internationalization issues. If the strings have different lengths, the shorter is treated as if it were extended with zeros.

#### INPUTS

string1, string2 - strings to be compared

#### RESULT

res - negative if string1 is below string2, 0 if they're the same, and positive if string1 is above string2.

#### NOTES

Commodore is planning a localization library which will take care of most details pertaining to system integration into different cultures, locales, and territories.

This function will automatically be replaced by a localized version whenever the locale.library is loaded in memory. As such, the collating order may change depending on the locale currently defined by the user. Take this fact into consideration when using this function, and do not rely on obtaining specific collating sequences.

#### BUGS

#### SEE ALSO

Strnicmp()

## 1.20 utility.library/Strnicmp

#### NAME

Strnicmp-- Case-insensitive string compare, length-limited. (V37)

#### SYNOPSIS

res = Strnicmp(string1, string2, length)

D0                      A0              A1              D0

LONG Strnicmp(char \*, char \*, LONG length);

#### FUNCTION

Strnicmp compares two strings, ignoring case. It handles all internationalization issues. If the strings have different lengths, the shorter is treated as if it were extended with zeros. It never compares more than <length> characters.

#### INPUTS

string1, string2 - strings to be compared

length                      - maximum number of characters to examine

#### RESULT

---

res - negative if string1 is below string2, 0 if they're the same, and positive if string1 is above string2.

#### NOTES

Commodore is planning a localization library which will take care of most details pertaining to system integration into different cultures, locales, and territories.

This function will automatically be replaced by a localized version whenever the locale.library is loaded in memory. As such, the collating order may change depending on the locale currently defined by the user. Take this fact into consideration when using this function, and do not rely on obtaining specific collating sequences.

#### BUGS

#### SEE ALSO

Stricmp()

## 1.21 utility.library/TagInArray

#### NAME

TagInArray -- Check if a Tag value appears in a Tag array. (V36)

#### SYNOPSIS

```
BOOL TagInArray( tag, tagArray )
D0          D0      A0
```

```
BOOL TagInArray( Tag tag, Tag *tagArray);
```

#### FUNCTION

Perform a quick scan to see if a tag value appears in an array terminated with TAG\_END. Returns TRUE if the value is found.

The 'tagArray' must be terminated by TAG\_END. It should NOT contain other system tag values, such as TAG\_MORE or TAG\_IGNORE. Note that this is an array of Tag values, NOT an array of TagItems.

This is sort of a "one shot" version of FilterTagItems().

#### INPUTS

tag                   - Tag value to search array for.  
tagArray             - a simple array terminated by TAG\_END.

#### RESULT

Boolean success of search.

#### BUGS

#### SEE ALSO

FilterTagItems()

---

## 1.22 utility.library/ToLower

### NAME

ToLower - Convert a character to lowercase. (V37)

### SYNOPSIS

```
char = ToLower(char)
D0          D0

char ToLower(char);
```

### FUNCTION

Converts a character to lowercase, handling international character sets.

### INPUTS

char - character to be converted.

### RESULT

char - lowercase version of input character.

### NOTES

Commodore is planning a localization library which will take care of most details pertaining to system integration into different cultures, locales, and territories.

This function will automatically be replaced by a localized version whenever the locale.library is loaded in memory. As such, the resulting converted character may change depending on the locale currently defined by the user. Take this fact into consideration when using this function, and do not rely on obtaining specific conversions.

### BUGS

### SEE ALSO

## 1.23 utility.library/ToUpper

### NAME

ToUpper - Convert a character to uppercase. (V37)

### SYNOPSIS

```
char = ToUpper(char)
D0          D0

char ToUpper(char);
```

### FUNCTION

Converts a character to uppercase, handling international character sets.

### INPUTS

char - character to be converted.

---

## RESULT

char - uppercase version of input character.

## NOTES

Commodore is planning a localization library which will take care of most details pertaining to system integration into different cultures, locales, and territories.

This function will automatically be replaced by a localized version whenever the locale.library is loaded in memory. As such, the resulting converted character may change depending on the locale currently defined by the user. Take this fact into consideration when using this function, and do not rely on obtaining specific conversions.

## BUGS

## SEE ALSO

## 1.24 utility.library/UDivMod32

## NAME

UDivMod32 -- Unsigned 32 by 32 bit division and modulus. (V36)

## SYNOPSIS

```
Quotient:Remainder = UDivMod32( Dividend, Divisor )
D0                D1                D0                D1
```

```
ULONG UDivMod32( ULONG, ULONG );
```

## FUNCTION

Divides the unsigned 32 bit dividend by the unsigned 32 bit divisor and returns a unsigned 32 bit quotient and remainder.

## INPUTS

|          |                             |
|----------|-----------------------------|
| Dividend | - unsigned 32 bit dividend. |
| Divisor  | - unsigned 32 bit divisor.  |

## RESULTS

|           |  |
|-----------|--|
| Quotient  | - unsigned 32 quotient of the division.  |
| Remainder | - unsigned 32 remainder of the division. |

## NOTES

## SEE ALSO

SDivMod32(), SMult32(), UMult32()

## BUGS

## 1.25 utility.library/UMult32

---

## NAME

UMult32 -- Unsigned 32 by 32 bit multiply with 32 bit result. (V36)

## SYNOPSIS

```
Result = UMult32( Arg1, Arg2 )
D0          D0      D1
```

```
ULONG UMult32( ULONG, ULONG );
```

## FUNCTION

Returns the unsigned 32 bit result of multiplying Arg1 by Arg2.

## INPUTS

Arg1, Arg2                      - unsigned multiplicands.

## RESULTS

Result                          - the unsigned 32 bit result of  
                                 multiplying Arg1 by Arg2.

## NOTES

## SEE ALSO

SDivMod32(), SMult32(), UDivMod32()

## BUGS