

amigaguide

COLLABORATORS

	<i>TITLE :</i> amigaguide		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 23, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	amigaguide	1
1.1	amigaguide.doc	1
1.2	amigaguide.library/--background--	1
1.3	amigaguide.library/--rexxhost--	1
1.4	amigaguide.library/AddAmigaGuideHostA	2
1.5	amigaguide.library/AmigaGuideSignal	4
1.6	amigaguide.library/CloseAmigaGuide	5
1.7	amigaguide.library/GetAmigaGuideAttr	6
1.8	amigaguide.library/GetAmigaGuideMsg	6
1.9	amigaguide.library/GetAmigaGuideString	7
1.10	amigaguide.library/LockAmigaGuideBase	8
1.11	amigaguide.library/OpenAmigaGuideA	8
1.12	amigaguide.library/OpenAmigaGuideAsyncAamigaguide.library/OpenAmigaGuideAsyncA	10
1.13	amigaguide.library/RemoveAmigaGuideHostAamigaguide.library/RemoveAmigaGuideHostA	11
1.14	amigaguide.library/ReplyAmigaGuideMsg	12
1.15	amigaguide.library/SendAmigaGuideCmdA	12
1.16	amigaguide.library/SendAmigaGuideContextAamigaguide.library/SendAmigaGuideContextA	13
1.17	amigaguide.library/SetAmigaGuideAttrsA	14
1.18	amigaguide.library/SetAmigaGuideContextAamigaguide.library/SetAmigaGuideContextA	15
1.19	amigaguide.library/UnlockAmigaGuideBaseamigaguide.library/UnlockAmigaGuideBase	16

Chapter 1

amigaguide

1.1 amigaguide.doc

```
--background--
--rexxhost--
AddAmigaGuideHostA ()
AmigaGuideSignal ()
CloseAmigaGuide ()
GetAmigaGuideAttr ()
GetAmigaGuideMsg ()
GetAmigaGuideString ()
LockAmigaGuideBase ()
OpenAmigaGuideA ()
OpenAmigaGuideAsyncA ()
RemoveAmigaGuideHostA ()
ReplyAmigaGuideMsg ()
SendAmigaGuideCmdA ()
SendAmigaGuideContextA ()
SetAmigaGuideAttrsA ()
SetAmigaGuideContextA ()
UnlockAmigaGuideBase ()
```

1.2 amigaguide.library/--background--

PURPOSE

The amigaguide.library provides a means whereby developers can easily add on-line help abilities to their applications.

1.3 amigaguide.library/--rexxhost--

HOST INTERFACE

amigaguide.library provides an ARexx function host interface that enables ARexx programs to take advantage of the features of AmigaGuide. The functions provided by the interface are directly related to the functions described herein, with the differences mostly being in the way they are called.

The function host library vector is located at offset -30 from the library. This is the value you provide to ARexx in the AddLib() function call.

FUNCTIONS

ShowNode (PUBSCREEN, DATABASE, NODE, LINE, XREF)

LoadXRef (NAME)

GetXRef (NODE)

1.4 amigaguide.library/AddAmigaGuideHostA

NAME

AddAmigaGuideHostA - Add a dynamic node host. (V34)

SYNOPSIS

```
handle = AddAmigaGuideHostA (hook, name, attrs);
d0          a0      d0      a1
```

```
AMIGAGUIDEHOST AddAmigaGuideHostA (struct Hook *, STRPTR,
                                   struct TagItem *);
```

FUNCTION

This function adds a callback hook to the dynamic node list.

A dynamic node allows an application to incorporate context-sensitive or live project data within their help system.

INPUTS

hook - The callback hook.
 name - Name of the AmigaGuideHost database that you are adding.
 The name must be unique.
 attrs - Additional attributes. None are defined at this time.

RETURNS

Returns NULL if unable to add the dynamic node host, otherwise returns a pointer to a handle that will eventually be passed to RemoveAmigaGuideHost() to remove the dynamic node host from the list.

NOTES

When AmigaGuide attempts to resolve a LINK command, it performs the following sequence of events.

- Splits the name into a path, a database and a node (only the node is required).
- Opens the database.
- Performs the following searches until the node is found:
 - Search the local database.
 - Search the local cross reference list.
 - Search the local dynamic node host.
 - Search the global help file (system help).
 - Search the global cross reference list.

Search the global dynamic node hosts.

SEE ALSO

RemoveAmigaGuideHostA()

EXAMPLE

```
/* Hook dispatcher */
ULONG __asm hookEntry(
    register __a0 struct Hook *h,
    register __a2 VOID *obj,
    register __a1 VOID *msg)
{
    /* Pass the parameters on the stack */
    return ((h->h_SubEntry)(h, obj, msg));
}

ULONG __saves
dispatchAmigaGuideHost (struct Hook *h, STRPTR db, Msg msg)
{
    struct opNodeIO *onm = (struct opNodeIO *) msg;
    ULONG retval = 0;

    switch (msg->MethodID)
    {
        /* Does this node belong to you? */
        case HM_FINDNODE:
        {
            struct opFindHost *ofh = (struct opFindHost *) msg;

            kprintf("Find [%s] in %sn", ofh->ofh_Node, db);

            /* Return TRUE to indicate that it's your node,
             * otherwise return FALSE. */
            retval = TRUE;
        }
        break;

        /* Open a node. */
        case HM_OPENNODE:
            kprintf("Open [%s] in %sn", onm->onm_Node, db);

            /* Provide the contents of the node */
            onm->onm_DocBuffer = TEMP_NODE;
            onm->onm_BuffLen = strlen(TEMP_NODE);

            /* Indicate that we were able to open the node */
            retval = TRUE;
            break;

        /* Close a node, that has no users. */
        case HM_CLOSENODE:
            kprintf("Close [%s] in %sn", onm->onm_Node, db);

            /* Indicate that we were able to close the node */
            retval = TRUE;
            break;
    }
}
```

```

        /* Free any extra memory */
        case HM_EXPUNGE:
            kprintf("Expunge [%s]n", db);
            break;

        default:
            kprintf("Unknown method %ldn", msg->MethodID);
            break;
    }

    return (retval);
}

main(int argc, char **argv)
{
    struct Hook hook;
    AMIGAGUIDEHOST hh;

    /* Open the library */
    if (AmigaGuideBase = OpenLibrary("amigaguide.library", 33))
    {
        /* Initialize the hook */
        hook.h_Entry = hookEntry;
        hook.h_SubEntry = dispatchAmigaGuideHost;

        /* Add the AmigaGuideHost to the system */
        if (hh = AddAmigaGuideHost(&hook, "ExampleHost", NULL))
        {
            /* Wait until we're told to quit */
            Wait(SIGBREAKF_CTRL_C);

            /* Try removing the host */
            while (RemoveAmigaGuideHost(hh, NULL) > 0)
            {
                /* Wait a while */
                Delay(5);
            }
        }

        /* close the library */
        CloseLibrary(AmigaGuideBase);
    }
}

```

BUGS

When a dynamic node host is first added it will receive a HM_FINDNODE message with an onm_Node of "Main". The AGA_HelpGroup attribute will always be zero for this particular message.

1.5 amigaguide.library/AmigaGuideSignal

NAME

AmigaGuideSignal - Obtain aysnc AmigaGuide signal. (V34)

SYNOPSIS

```
signal = AmigaGuideSignal ( handle );  
d0                                a0
```

```
ULONG AmigaGuideSignal (AMIGAGUIDECONTEXT);
```

FUNCTION

This function returns the signal bit to Wait on for AmigaGuideMsg's for a particular AmigaGuide database.

INPUTS

handle - Handle to a AmigaGuide system.

EXAMPLE

```
ULONG sigw, sigh;  
AMIGAGUIDECONTEXT handle;  
  
/* get the signal bit to wait on for a AmigaGuide message */  
sigh = AmigaGuideSignal(handle);  
  
/* add the signal bit into the total signals to wait on */  
sigw |= sigh;
```

RETURNS

signal - Signal bit to Wait on.

SEE ALSO

OpenAmigaGuideAsyncA(), GetAmigaGuideMsg(), ReplyAmigaGuideMsg()

1.6 amigaguide.library/CloseAmigaGuide

NAME

CloseAmigaGuide - Close a AmigaGuide client. (V34)

SYNOPSIS

```
CloseAmigaGuide (handle);  
a0
```

```
VOID CloseAmigaGuide (AMIGAGUIDECONTEXT);
```

FUNCTION

Closes a synchronous, or asynchronous, AmigaGuide client.

This function will also close all windows that were opened for the client.

INPUTS

handle - Handle to an AmigaGuide client.

SEE ALSO

OpenAmigaGuideA(), OpenAmigaGuideAsyncA()

1.7 amigaguide.library/GetAmigaGuideAttr

NAME

GetAmigaGuideAttr - Get an AmigaGuide attribute. (V34)

SYNOPSIS

```
retval = GetAmigaGuideAttr (tag, handle, storage);  
d0          d0    a0    a1
```

```
LONG GetAmigaGuideAttr (Tag, AMIGAGUIDECONTEXT, ULONG *);
```

FUNCTION

This function is used to obtain attributes from AmigaGuide.

INPUTS

tag - Attribute to obtain.
handle - Handle to an AmigaGuide system.
storage - Pointer to appropriate storage for the answer.

TAGS

AGA_Path (BPTR) - Pointer to the current path used by AmigaGuide.

AGA_XRefList (struct List *) - Pointer to the current cross reference list.

RETURNS

SEE ALSO

SetAmigaGuideAttrsA()

1.8 amigaguide.library/GetAmigaGuideMsg

NAME

GetAmigaGuideMsg - Receive async AmigaGuide message. (V34)

SYNOPSIS

```
msg = GetAmigaGuideMsg (handle);  
d0          a0
```

```
struct AmigaGuideMsg *GetAmigaGuideMsg (AMIGAGUIDECONTEXT);
```

FUNCTION

This function returns a SIPC message from the AmigaGuide system, if there is a message available.

INPUTS

handle - Handle to a AmigaGuide system.

EXAMPLE

```
AMIGAGUIDECONTEXT handle;  
struct AmigaGuideMsg *agm;
```

```

/* get a AmigaGuide message */
while (agm = GetAmigaGuideMsg(handle))
{
    /* process the event */
    switch (agm->agm_Type)
    {
        case ToolCmdReplyID:    /* a command has completed */
            if (agm->agm_Pri_Ret)
            {
                /* An error occurred, the reason is in agm_Sec_Ret.
                 * The command string is in agm_Data
                 */
            }
            break;

        case ToolStatusID:     /* status message */
            if (agm->agm_Pri_Ret)
            {
                /* an error occurred, the reason is in agm_Sec_Ret */
            }
            break;

        default:
            break;
    }

    /* reply to the AmigaGuide message */
    ReplyAmigaGuideMsg(agm);
}

```

RETURNS

msg - Pointer to a SIPC message or NULL if no message was available.

SEE ALSO

OpenAmigaGuideAsyncA(), AmigaGuideSignal(), ReplyAmigaGuideMsg()

1.9 amigaguide.library/GetAmigaGuideString

NAME

GetAmigaGuideString - Get an AmigaGuide string.

(V34)

SYNOPSIS

```

txt = GetAmigaGuideString (id);
d0                                d0

```

```

STRPTR GetAmigaGuideString (ULONG);

```

FUNCTION

This function is used to obtain a localized string given the ID.

INPUTS

ID -- Valid AmigaGuide string id.

RETURNS

A pointer to the string. NULL for an invalid string.

SEE ALSO

1.10 amigaguide.library/LockAmigaGuideBase

NAME

LockAmigaGuideBase - Lock an AmigaGuide client. (V34)

SYNOPSIS

```
key = LockAmigaGuideBase (AMIGAGUIDECONTEXT handle);
                             a0
```

```
LONG LockAmigaGuideBase (AMIGAGUIDECONTEXT);
```

FUNCTION

This function is used to lock the AmigaGuide context handle while working with data obtained with the the GetAmigaGuideAttr() function.

INPUTS

handle - AMIGAGUIDECONTEXT handle obtained with OpenAmigaGuideAsync().

RETURNS

Returns a key to pass to UnlockAmigaGuideBase().

SEE ALSO

UnlockAmigaGuideBase()

1.11 amigaguide.library/OpenAmigaGuideA

NAME

OpenAmigaGuideA - Open a synchronous AmigaGuide database.

SYNOPSIS

```
handle = OpenAmigaGuideA (nag, attrs);
d0                      a0    a1
```

```
AMIGAGUIDECONTEXT OpenAmigaGuideA (struct NewAmigaGuide *,
                                   struct TagItem *);
```

```
handle = OpenAmigaGuide (nag, tag1, ...);
```

```
AMIGAGUIDECONTEXT OpenAmigaGuide (struct NewAmigaGuide *,
                                   Tag tag1, ...);
```

FUNCTION

Opens a AmigaGuide database, complete with the first viewing window, for synchronous activity.

Before you call `OpenAmigaGuide()`, you must initialize a `NewAmigaGuide` structure. `NewAmigaGuide` is a structure that contains all the information needed to open a database. The `NewAmigaGuide` structure must be retained until the call returns.

The function will not return until the user closes all the windows.

INPUTS

`nag` - Pointer to an instance of a `NewAmigaGuide` structure. That structure is initialized with the following data.

`nag_Lock`
Lock on the directory that the database is located in.
Not needed if `nag_Name` contains the complete path name.

`nag_Name`
Name of the `AmigaGuide` database.

`nag_Screen`
Screen to open the viewing windows on, `NULL` for the Workbench screen.

`nag_PubScreen`
Pointer to the name of the public screen to open on.
Must already be opened.

`nag_HostPort`
Name of the applications' `ARexx` port (currently not used).

`nag_ClientPort`
Base name to use for the databases' `ARexx` port.

`nag_Flags`
Used to specify the requirements of this database. The flags are defined in `<libraries/amigaguide.h>`.

`nag_Context`
NULL terminated array of context nodes, in the form of:

```
/* context array */
STRPTR context[] =
{
    "MAIN",
    "INTRO",
    "GADGETS",
    NULL
};
```

The context array is not copied, but referenced, therefore must remain static throughout the useage of the `AmigaGuide` system. This array is only referenced when using the `SetAmigaGuideContext()` function.

`nag_Node`
Node to start at (does not work with `OpenAmigaGuideAsync()`).

nag_Line
Line to start at (does not work with OpenAmigaGuideAsync()).

nag_Extens
Used by V37 and beyond to pass additional arguments.

nag_Client
This is a private pointer, MUST be initialized to NULL.

attrs - Additional attributes.

TAGS

AGA_HelpGroup (ULONG) -- Unique identifier used to identify the AmigaGuide help windows. See OpenWindow() and GetUniqueID().

Default for this tag is NULL. Applicability is (I). (V39)

EXAMPLE

```
/* Short example showing synchronous AmigaGuide access */
LONG ShowAmigaGuideFile (STRPTR name, STRPTR node, LONG line)
{
    struct NewAmigaGuide nag = {NULL};
    AMIGAGUIDECONTEXT handle;
    LONG retval = 0L;

    /* Fill in the NewAmigaGuide structure */
    nag.nag_Name = name;
    nag.nag_Node = node;
    nag.nag_Line = line;

    /* Open the AmigaGuide client */
    if ( handle = OpenAmigaGuide(&nag, NULL))
    {
        /* Close the AmigaGuide client */
        CloseAmigaGuide(handle);
    }
    else
    {
        /* Get the reason for failure */
        retval = IoErr();
    }

    return (retval);
}
```

RETURNS

handle - Handle to a AmigaGuide system.

SEE ALSO

OpenAmigaGuideAsyncA(), CloseAmigaGuide()

1.12 amigaguide.library/OpenAmigaGuideAsyncAamigaguide.library/OpenAmigaGuide

NAME

OpenAmigaGuideAsyncA - Open an AmigaGuide database async (V34)

SYNOPSIS

```
handle = OpenAmigaGuideAsyncA (nag, attrs);
d0                      a0    d0

AMIGAGUIDECONTEXT OpenAmigaGuideAsyncA (struct NewAmigaGuide *,
                                         struct TagItem *);

handle = OpenAmigaGuideAsync (nag, tag1, ...);

AMIGAGUIDECONTEXT OpenAmigaGuideAsyncA (struct NewAmigaGuide *,
                                         Tag tag1, ...);
```

FUNCTION

Opens an AmigaGuide database for asynchronous use.

The NewAmigaGuide structure, and its pointers, must stay valid until an ActiveToolID or ToolStatusID message is received by the calling process.

This function actually spawns OpenAmigaGuide() as another process, so, for further documentation, refer to the OpenAmigaGuide() function.

INPUTS

nag - Pointer to a valid NewAmigaGuide structure.
(see OpenAmigaGuide() for documentation on its useage).

attrs - Additional attributes. See OpenAmigaGuideA().

RETURNS

handle - Handle to an AmigaGuide system.

SEE ALSO

OpenAmigaGuideA(), CloseAmigaGuide()

1.13 amigaguide.library/RemoveAmigaGuideHostAamigaguide.library/RemoveAmigaG

NAME

RemoveAmigaGuideHostA - Remove a dynamic node host. (V34)

SYNOPSIS

```
use = RemoveAmigaGuideHostA (key, attrs)
d0                      a0    a1

LONG RemoveAmigaGuideHostA (AMIGAGUIDEHOST, struct TagItem *);

use = RemoveAmigaGuideHost (key, tag1, ...);

LONG RemoveAmigaGuideHost (AMIGAGUIDEHOST, Tag, ...);
```

FUNCTION

This function removes a dynamic node host, that was added by

AddAmigaGuideHost(), from the system.

INPUTS

key - Key that was returned by AddAmigaGuideHost().

attrs - Additional attributes. None are defined at this time.

RETURNS

use - Number of outstanding clients of this database. You can not exit until use==0.

SEE ALSO

AddAmigaGuideHostA()

1.14 amigaguide.library/ReplyAmigaGuideMsg

NAME

ReplyAmigaGuideMsg - Reply to an AmigaGuide message. (V34)

SYNOPSIS

```
ReplyAmigaGuideMsg ( msg );
                   a0
```

```
VOID ReplyAmigaGuideMsg (struct AmigaGuideMsg *msg);
```

FUNCTION

This function is used to reply to an AmigaGuide SIPC message.

INPUTS

msg - Pointer to a SIPC message returned by a previous call to GetAmigaGuideMsg().

SEE ALSO

OpenAmigaGuideAsyncA(), AmigaGuideSignal(), GetAmigaGuideMsg()

1.15 amigaguide.library/SendAmigaGuideCmdA

NAME

SendAmigaGuideCmdA - Send a command string to AmigaGuide (V34)

SYNOPSIS

```
success = SendAmigaGuideCmdA (handle, cmd, attrs );
d0              a0      d0      d1
```

```
BOOL SendAmigaGuideCmdA (AMIGAGUIDECONTEXT, STRPTR, struct TagItem *);
```

```
success = SendAmigaGuideCmd (handle, cmd, tag1, ...);
```

```
BOOL SendAmigaGuideCmd (AMIGAGUIDECONTEXT, STRPTR, Tag);
```

FUNCTION

This function sends a command string to an AmigaGuide system. The

command can consist of any valid AmigaGuide action command.

The following are the currently valid action commands:

ALINK <name> - Load the named node into a new window.

LINK <name> - Load the named node.

RX <macro> - Execute an ARexx macro.

RXS <cmd> - Execute an ARexx string file. To display a picture, use 'ADDRESS COMMAND DISPLAY <picture name>', to display a text file 'ADDRESS COMMAND MORE <doc>'.

CLOSE - Close the window (should only be used on windows that were started with ALINK).

QUIT - Shutdown the current database.

INPUTS

handle - Handle to an AmigaGuide system.

cmd - Command string.

attrs - Future expansion, must be set to NULL for now.

TAGS

AGA_Context (ULONG) - Data is used as an index into nag_Context array. This is used to build and send a LINK command.

EXAMPLE

```
/* bring up help on a particular subject */
SendAmigaGuideCmd(handle, "LINK MAIN", NULL);
```

RETURNS

Returns TRUE if the message was sent, otherwise returns FALSE.

BUGS

ALINK does not open a new window when using V39.

SEE ALSO

1.16 amigaguide.library/SendAmigaGuideContextAamigaguide.library/SendAmigaGuideContextA

NAME

SendAmigaGuideContextA - Align an AmigaGuide system on the context ID.
(V34)

SYNOPSIS

```
success = SendAmigaGuideContextA (handle, attrs);
d0                                a0                                d0
```

```
BOOL SendAmigaGuideContextA (AMIGAGUIDECONTEXT, struct TagItem *);
```

```
success = SendAmigaGuideContext (handle, tag1, ...);
```



```
BOOL SendAmigaGuideContext (AMIGAGUIDECONTEXT, Tag, ...);
```

FUNCTION

This function is used to send a message to an AmigaGuide system to align it on the current context ID.

This function effectively does a:

```
SendAmigaGuideCmd(handle 'LINK ContextArray[contextID]', NULL);
```

INPUTS

handle - Handle to an AmigaGuide system.

future - Future expansion, must be set to NULL for now.

EXAMPLE

```
struct IntuiMessage *imsg;

...

case RAWKEY:
    switch (imsg->Code)
    {
        case 95:
            /* bring up help on a particular subject */
            SendAmigaGuideContext(handle, NULL);
            break;
        ...
    }
    break;

...
```

RETURNS

success - Returns TRUE if the message was sent, otherwise returns FALSE.

SEE ALSO

SetAmigaGuideContextA(), SendAmigaGuideCmdA()

1.17 amigaguide.library/SetAmigaGuideAttrsA

NAME

SetAmigaGuideAttrsA - Set an AmigaGuide attribute. (V34)

SYNOPSIS

```
retval = SetAmigaGuideAttrsA (handle, attrs);
d0                                a0      a1
```

```
LONG SetAmigaGuideAttrsA (AMIGAGUIDECONTEXT, struct TagItem *);
```

```
retval = SetAmigaGuideAttrs (handle, tag1, ...);
```

```
LONG SetAmigaGuideAttrs (AMIGAGUIDECONTEXT, Tag, ...);
```

FUNCTION

This function is used to set AmigaGuide attributes.

INPUTS

handle - Pointer to an AmigaGuide handle.

attrs - Attribute pairs to set.

TAGS

AGA_Activate (BOOL) - AmigaGuide activates the window when it receives a LINK command. This tag allows the application developer to turn that feature off and on.

SEE ALSO

GetAmigaGuideAttr()

1.18 amigaguide.library/SetAmigaGuideContextAamigaguide.library/SetAmigaGuideCo

NAME

SetAmigaGuideContextA - Set the context ID for an AmigaGuide system.
(V34)

SYNOPSIS

```
success = SetAmigaGuideContextA ( handle, context, attrs );
d0                                a0                d0                d1
```

```
BOOL SetAmigaGuideContextA (AMIGAGUIDECONTEXT, ULONG, struct TagItem *
);
```

```
success = SetAmigaGuideContext (handle, context, tag1, ...);
```

```
BOOL SetAmigaGuideContext (AMIGAGUIDECONTEXT, ULONG, Tag, ...);
```

FUNCTION

This function, and the SendAmigaGuideContext() function, are used to provide a simple way to display a node based on a numeric value, instead of having to build up a slightly more complex command string.

INPUTS

handle - Handle to an AmigaGuide system.

context - Index value of the desired node to display.

future - Future expansion, must be set to NULL for now.

EXAMPLE

```
/* sample context table */
STRPTR ContextArray[] =
{
    "MAIN",
    "FILEREQ",
    "PRINT",
    "ABOUT",
```

```

        NULL
    };

    /* quickie defines */
#define HELP_MAIN      0
#define HELP_FILEREQ   1
#define HELP_PRINT     2
#define HELP_ABOUT     3

    ...

    struct NewAmigaGuide nag = {NULL};

    /* initialize the context table */
    nag.nag_Context = ContextArray;

    ...

    /* bring up help on a particular subject */
    SetAmigaGuideContext(handle, HELP_ABOUT, NULL);

RETURNS
    success - Returns TRUE if a valid context ID was passed,
              otherwise returns FALSE.

SEE ALSO
    SendAmigaGuideContextA(), SendAmigaGuideCmdA()

```

1.19 amigaguide.library/UnlockAmigaGuideBaseamigaguide.library/UnlockAmigaGuide

NAME
 UnlockAmigaGuideBase - Unlock an AmigaGuide client. (V34)

SYNOPSIS

```

    UnlockAmigaGuideBase (key);
                        d0

    VOID UnlockAmigaGuideBase (LONG);

```

FUNCTION
 This function is used to release a lock obtained with LockAmigaGuideBase().

INPUTS
 key - Value returned by LockAmigaGuideBase().

SEE ALSO
 LockAmigaGuideBase().