Libraries

COLLABORATORS

	TITLE : Libraries	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Libraries

1.1 Amiga® RKM Libraries: A Linker Libraries

This section describes the amiga.lib and debug.lib libraries. Unlike the libraries described in the other chapters of the manual, these are not shared run-time libraries. Code from the linker libraries is inserted by the linker into your final program. Only the functions you use are pulled into your code. These libraries are typically supplied by your language or compiler vendor.

Amiga.lib Debug.lib

1.2 A Appendix A: Linker Libraries / Amiga.lib

This is the main Amiga scanned linker library, linked with most programs for the Amiga. The major components of amiga.lib are:

* stubs — Functions for each Amiga ROM routine that copy arguments from the stack to the CPU registers -- thereby enabling stack-based C compilers to call register-based Amiga ROM routines.

* offsets — The negative offset from the library base for each Amiga function. These are called Library Vector Offsets (_LVO).

* Exec — C functions which simplify many Exec procedures such as the creation and deletion of tasks, ports, and I/O request structures.

* clib — C support functions including pseudo-random number generation and a limited set of file and stdio functions designed to work directly with AmigaDOS file handles.

* Math — C functions which provide some basic conversions to and from Fast Floating Point (FFP) format numbers.

```
  * Graphics     C support functions to add and remove tasks from the
                 vertical-blanking interval interrupt server chain.

  * ARexx        C support functions for ARexx variable handling and
                 message checking.

    NOTE:
    -----
    The Timer, Commodities, and Intuition support functions listed below
    are valid only for use with Release 2.04 (V37) or a later version of
    the system software.

  * Timer        C support functions to do common timer device
                 operations.

  * Commodities  C functions which support the Commodities system.
                 Included are functions to deal with ToolTypes, and to
                 create various Commodities objects.

  * Intuition    Functions which provide support for Intuition's hook
                 and Boopsi sub-systems.

Most applications link with and use at least one function in amiga.lib.
The functions available are as follows.

 Exec Support      Math          Arexx      Commodities
 Clib              Graphics      Timer      Intuition
```

1.3 A / Amiga.lib / Exec Support

```
BeginIO()
    This function takes an IORequest and passes it directly to the
    BEGINIO vector of the proper device.  This works exactly like
    SendIO(), but does not clear the io_Flags field first. This function
    does not wait for the I/O to complete.

CreateExtIO() and DeleteExtIO()
    CreateExtIO() allocates memory for and initializes a new I/O request
    block of a program-specified number of bytes.  The number of bytes
    must be the size of a legal IORequest (or extended request) or very
    nasty things will happen. DeleteExtIO() frees up an I/O request as
    allocated by CreateExtIO(). The mn_Length field determines how much
    memory to deallocate.

CreatePort() and DeletePort()
    CreatePort() allocates and initializes a new message port. The
    message list of the new port will be prepared for use via NewList().
    The port will be set to signal your task when a message arrives
    (PA_SIGNAL). DeletePort() deletes the port created by CreatePort().
    All messages that may have been attached to that port must already
    have been replied to.

CreateStdIO() and DeleteStdIO()
    CreateStdIO() allocates memory for and initializes a new IOStdReq
    structure. DeleteStdIO() frees up an IOStdReq allocated by
```

```
    CreateStdIO().

CreateTask() and DeleteTask()
    These functions simplify creation and deletion of subtasks by
    dynamically allocating and initializing the required structures and
    stack space.  They also add the task to Exec's task list with the
    given name and priority.  A tc_MemEntry list is provided so that all
    stack and structure memory allocated by CreateTask() is automatically
    deallocated when the task is removed. Before deleting a  task with
    DeleteTask(), you must first make sure that the task is not currently
    executing any system code which might try to signal the task after it
    is gone.

NewList()
    Prepares a List structure for use; the list will be empty and ready
    to use.
```

1.4 A / Amiga.lib / Clib

```
FastRand()
    Generates a pseudo-random number.  The seed value is taken from
    stack, shifted left one position, exclusive-or'ed with hex value
    $1D872B41 and returned.

RangeRand()
    RangeRand() accepts a value from 1 to 65535, and returns a value
    within that range (a 16-bit integer).  Note that this function is
    implemented in C.

fclose()
    Closes a file.

fgetc()
    Gets a character from a file.

fprintf()
    Prints a formatted output line to a file.

fputc()
    Puts character to file.

fputs()
    Writes a string to file.

getchar()
    Gets a character from stdin.

printf()
    Puts format data to stdout.

putchar()
    Puts character to stdout.

puts()
    Puts a string to stdout, followed by newline.
```

sprintf()
    Formats data into a string (see Exec RawDoFmt()).

1.5 A / Amiga.lib / Math

afp()
    Converts ASCII string variable into fast floating-point.

arnd()
    ASCII round-off of the provided floating-point string."

dbf()
    Accepts a dual-binary format floating-point number and converts it to
    FFP format.

fpa()
    Accepts an FFP number and the address of the ASCII string where its
    converted output is to be stored. The number is converted to a NULL
    terminated ASCII string and stored at the address provided.
    Additionally, the base ten (10) exponent in binary form is returned.

1.6 A / Amiga.lib / Graphics

AddTOF() and RemTOF()
    AddTOF() adds a task to the vertical-blanking interval interrupt
    server chain. This frees C programmers from the burden of having to
    write an assembly language routine to perform this function. The task
    can be removed with RemTOF().

1.7 A / Amiga.lib / ARexx

GetRexxVar() and SetRexxVar()
    GetRexxVar() attempts to extract the value of a variable from a
    running ARexx script/program. SetRexxVar() will attempt to set the
    value of a particular variable in a running ARexx script.

CheckRexxMsg()
    This function checks to make sure that a RexxMsg is from ARexx
    directly. Messages used by the Rexx Variable Interface (RVI) routines
    are required to be directly from ARexx.

1.8 A / Amiga.lib / Timer

TimeDelay()
    This function waits for a specified period of time before returning
    to the the caller.

1.9 A / Amiga.lib / Commodities

ArgArrayInit() and ArgArrayDone()
    ArgArrayInit() returns an array of strings suitable for sending to icon.library/FindToolType(). This array will be the ToolTypes array of the program's icon, if it was started from Workbench. It will just be 'argv' if the program was started from a shell. ArgArrayDone() frees memory and does cleanup required after a call to ArgArrayInit().

ArgInt() and ArgString()
    These functions look for a particular entry in a ToolType array returned by ArgArrayInit() and return the integer (ArgInt()) or string (ArgString()) associated with that entry. A default value can be passed to each function which will be returned in the event that the requested entry could not be found in the ToolType array.

CxCustom()
    This function creates a custom commodity object. The action of this object on receiving a commodity message is to call a function of the application programmer's choice.

CxDebug()
    This function creates a Commodities debug object. The action of this object on receiving a Commodities message is to print out information about the message through the serial port (using the debug.lib/KprintF() routine). A specified 'id' will also be displayed.

CxFilter()
    Creates a Commodities input event filter object that matches a description string. The description string is in the same format as strings expected by commodities.library/SetFilter(). If the description string is NULL, the filter will not match any messages.

CxSender()
    This function creates a Commodities sender object. The action of this object on receiving a Commodities message is to copy the Commodities message into a standard Exec Message, to put a supplied id in the message as well, and to send the message off to the message port.

CxSignal()
    This function creates a Commodities signal object. The action of this object on receiving a Commodities message is to send a signal to a task. The caller is responsible for allocating the signal and determining the proper task ID.

CxTranslate()
    This function creates a Commodities translator object. The action of this object on receiving a Commodities message is to replace that message in the commodities network with a chain of Commodities input messages.

HotKey()
    This function creates a triad of commodity objects to accomplish a high-level function.

    The three objects are a filter, which is created to match by

CxFilter(), a sender created by CxSender(), and a translator which is created by CxTranslate(), so that it swallows any commodity input event messages that are passed down by the filter.

This is the simple way to get a message sent to your program when the user performs a particular input action.

InvertString()
This function returns a linked list of input events which would translate into the string using the supplied keymap (or the system default keymap if the supplied keymap is NULL).

This chain should eventually be freed using FreeIEvents().

FreeIEvents()
This function frees a linked list of input events as obtained from InvertString().

1.10 A / Amiga.lib / Intuition

CallHook() and CallHookA()
These functions invoke hooks. CallHook() expects a parameter packet ("message") on the stack, while CallHookA() takes a pointer to the message.

DoMethod() and DoMethodA()
Boopsi support functions that ask a specified Boopsi object to perform a specific message. The message is passed in the function call for DoMethodA() and on the stack for DoMethod(). The message is invoked on the object's true class.

DoSuperMethod() and DoSuperMethodA()
Boopsi support functions that ask a Boopsi object to perform a supplied message as if it was an instance of its superclass. The message is passed in the function call for DoSuperMethodA() and on the stack for DoSuperMethod().

CoerceMethod() and CoerceMethodA()
Boopsi support functions that ask a Boopsi object to perform a supplied message as if it was an instance of some other class. The message is passed in the function call for CoerceMethodA() and on the stack for CoerceMethod.

SetSuperAttrs()
Boopsi support function which invokes the OM_SET method on the superclass of the supplied class for the supplied object. Allows the ops_AttrList to be supplied on the stack (i.e. in a varargs way).

1.11 A Appendix A: Linker Libraries / Debug.lib

This link library contains standard I/O (stdio) style functions for communicating with a serial terminal connected to the Amiga via its

built-in serial port. Typically this terminal will be a 9600-baud, 8 data-bits, one stop-bit connection to an external terminal or an Amiga running a terminal package. The debug.lib functions allow you to output messages and prompt for input (even from within low level task or interrupt code) without disturbing the Amiga's display and or current state (other than the serial hardware itself).

No matter how badly the system may have crashed, these functions can usually get a message out. A similar debugging library (currently called ddebug.lib) is available for sending debugging output to the parallel port. This is useful for debugging serial applications. Ddebug.lib is not documented here. It contains functions similar to debug.lib but with names starting with 'D' instead of 'K'.

Debug.lib is a link library that provides useful diagnostic functions that are handy for developing code. It includes the following functions:

KCmpStr()
    Compare two null-terminated strings.

KGetChar()
    Get a character from the console.

KGetNum()
    Get a number from the console.

KMayGetChar()
    Return a character if present, but don't wait.

KPrintF()
    Print formatted data to the console.

KPutChar()
    Put a character to the console.

KPutStr()
    Put a string to the console.

Please refer to the Amiga ROM Kernel Reference Manual: Includes and Autodocs for a detailed description of the functions.