identify.library

A library that converts cryptical IDs to a human-readable form (developer documentation)
Version 13.0, 9 September 2001

Richard Körber

1 Copyright

Please read the following parts carefully.

If you do not agree to these Copyright notes, you must delete this archive and all related files.

COPYRIGHT

NOTE: You accept the following terms by starting the software, even for a test drive only.

Identify is © Copyright 1996-2001 by Richard Körber. All rights reserved.

You only have the right to use the software, but no rights on the software itself. Disassembling, resourcing and all other ways of reverse engineering is forbidden. This means the expansion database in particular!

FREEWARE

Identify is FreeWare. You are allowed to use the packet without paying a fee or similar to the author.

COPYING

You can copy the packet as long as it remains entire and unchanged.

You are allowed to compress the packet using a customary compression software (as lha, lzh, lzx, dms). You must not compress single files of the packet (e.g. PowerPacker or Imploder).

PARTIAL COPYING

You are allowed to add the files `identify.library`, `rexxidentify.library` and `InstallIfy` to your own program packet, without the need to add the whole Identify packet. In this case, you *must* mention in the documentation that you are using Identify by Richard Körber, and where a full Identify distribution is available.

DISTRIBUTION

You must not exceed an usual price on the market for your working and material. This means a maximum of 5 DM (or the equivalent amount in other currencies) for disks and 35 DM for CD-ROMs containing a PD software collection.

I explicitly permit the distribution via AmiNet, Meeting Pearls, Fred Fish and other well-known PD series.

LIABILITY

You are using the program as it is, with all flaws, and on your own risk! I grant no warranty for the software meeting a special purpose. This software may cause financial damage or harm people.

LIMITATIONS

You are not allowed to use this software and its results

- for fascism or military purposes
- if you do not agree to the copyright note

In this case you must delete the software and all related and generated files immediately!

TRADEMARKS

All Copyrights and Trademarks are held by their owners.

2 My Address

Please send all bug-reports, board descriptions, missing graphic OS, keyfiles, flames and so on to one of the following e-mail addresses:

INTERNET

I'm reachable through Internet via these E-Mail addresses:

```
shred@shredzone.de
rkoerber@gmx.de
shred@web.de
```

Check my home page for the latest release and other programs. The URL is:

```
http://www.shredzone.de
http://shredzone.home.pages.de
```

SNAIL MAIL

You can send me a snail mail letter, too. My address is:

```
Richard Körber
Paffrather Straße 40
51465 Bergisch Gladbach
Germany
```

Please enclose a "1,10 DM" stamp if you live in Germany and want to get a reply. Do NOT send disks.

3 ARexx

Since V6.0 an ARexx function library is available. You can now also use Identify in your ARexx scripts.

To do so, you just have to install the `rexxidentify.library` into `libs:`. The library is included into ARexx by using a `CALL AddLib("/libs/rexxidentify.library",0,-30,0)` command.

These functions are available since Release 1:

`ID_Release()`

Returns the release- and version numbers of the `rexxidentify.library`. The format is: '`<release> <version>.<revision>(<date>)`' (Example: '`1 1.0(23.4.97)`'). You should use the `<release>` part to check out if a function or option is available (e.g. `Word(ID_Release(),1)`). The result can also be easily split up into the single contents by using the `PARSE` command.

`ID_NumBoards()`

Returns the number of expansion boards added to the system. You can easily construct a loop for `ID_Expansion`, using this function.

`ID_Expansion(<board>,<result code>)`

Returns the appropriate result to the board number `<board>` (0 to `ID_NumBoards()`-1) and the `<result code>`. Result codes are:

`MANUF`	Manufacturer name
`PROD`	Product name
`CLASS`	Product class (localized)
`ADDRESS`	Memory address of the expansion (hexadecimal)
`SIZE`	Reserved amount of memory for the expansion board (decimal, KBytes)
`SHUTUP`	Has the expansion been shut up? (0:No, 1:Yes)
`SECONDARY`	(since Release 4) Checks if the entry is primary (Result: `Primary`) or secondary (Result: `Secondary`).
`CLASSID`	(since Release 5) Returns the Class ID of the expansion board, see include files (decimal).

`ID_Function(<library>,<offset>)`

Returns the name of the `<library>`'s function and the offset `<offset>`. See the Shell program `Function`.

`ID_Alert(<code>,<result code>)`

Returns the appropriate result to the alert code `<code>` (hexadecimal string) and the `<result code>`. Result codes are:

`DEAD`	Deadend or Recovery?
`SUBSYS`	System causing the alert
`GENERAL`	General alert class
`SPEC`	Specific alert class

See the Shell program `Guru`.

`ID_Hardware(<result code>,{<option>,...})`

Returns a description of the hardware. `<result code>` are:

`SYSTEM`	Used Amiga model (e.g. '`Amiga 4000`')
`CPU`	Used CPU
`FPU`	Used FPU, if available
`MMU`	Used MMU, if available
`OSVER`	AmigaOS ROM version
`EXECVER`	exec.library version
`WBVER`	Workbench version, if available
`ROMSIZE`	Size of the AmigaOS ROM.
`CHIPSET`	Available chip set (e.g. '`AGA`')
`GFXSYS`	Used graphics system (e.g. '`CyberGraphX`')
`CHIPRAM`	Total size of Chip RAM (includes virtual RAM)
`FASTRAM`	Total size of Fast RAM (includes virtual RAM)
`RAM`	Total size of total RAM (includes virtual RAM)
`SETPATCHVER`	SetPatch version, if available.
`AUDIOSYS`	Used audio system (e.g. '`AHI`')
`OSNR`	Used OS version (e.g. '`3.5`')
`VMMCHIPRAM`	Size of virtual Chip RAM
`VMMFASTRAM`	Size of virtual Fast RAM

`VMMRAM` Size of virtual RAM

`PLNCHIPRAM`
Size of physical Chip RAM

`PLNFASTRAM`
Size of physical Fast RAM

`PLNRAM` Size of physical RAM

`VBR` Address of the processor vectors

`LASTALERT`
Last system alert

`VBLANKFREQ`
VBlank interrupt frequency

`POWERFREQ`
Power frequency

`ECLOCK` Special system clock's frequency

`SLOWRAM` Size of the A500 and A2000's special Fast RAM.

`GARY` Gary version

`RAMSEY` Ramsey version

`BATTCLOCK`
Battery backed up clock available?

`CHUNKYPLANAR`
Does a chunky planar hardware exist?

`POWERPC` Is a PowerPC available?

`PPCCLOCK` The clock of the PowerPC, in MHz units.

`CPUREV` (since Release 5) Returns the revision of the built-in CPU, if available.

`CPUCLOCK` (since Release 5) Returns the CPU clock, in MHz units.

`FPUCLOCK` (since Release 5) Returns the FPU clock, if available, in MHz units.

`RAMACCESS`
(since Release 6) Returns the access time of the motherboard RAM (unit ns), if available.

`RAMWIDTH` (since Release 6) Returns the width of the motherboard RAM (bit), if available.

`RAMCAS` (since Release 6) Returns the CAS mode of the motherboard RAM, if available.

`RAMBANDWIDTH`
(since Release 6) Returns the motherboard RAM bandwidth, if available.

`TCPIP` (since Release 7) Returns the used TCP/IP stack, if started.

`PPCOS` (since Release 7) Returns the PowerPC OS (PowerUp, WarpOS), if available.

`AGNUS` (since Release 7) Returns the Agnus chip revision, if available.

`AGNUSMODE`
(since Release 7) Returns the Agnus chip mode (PAL or NTSC).

`DENISE` (since Release 8) Returns the Denise chip version, if available.

`DENISEREV`
(since Release 8) Returns the Denise chip revision, if available.

`BOINGBAG` (since Release 10) Returns the BoingBag number, if available.

`EMULATED` (since Release 11) Returns 0 if this is a real Amiga. Any other value means that the Amiga is emulated.

`XLVERSION`
(since Release 11) The major version number of the AmigaXL emulator. 0 if AmigaXL is not available.

`HOSTOS` (since Release 11) The host operating system of the emulated Amiga. Currently only available on AmigaXL systems.

`HOSTVER` (since Release 11) The host operating system version of the emulated Amiga. Currently only available on AmigaXL systems.

`HOSTMACHINE`
(since Release 11) The host architecture of the emulated Amiga. Currently only available on AmigaXL systems.

`HOSTCPU` (since Release 11) The host CPU of the emulated Amiga. Currently only available on AmigaXL systems.

`HOSTSPEED`
(since Release 11) The host CPU speed of the emulated Amiga. Currently only available on AmigaXL systems.

These `<options>` are allowed:

`EMPTYNA` Returns an empty string if the item is not available. Otherwise, a localized 'not available' kind of string is returned.

`NOLOCALE` The return string is always in English, independent of the current language.

See also the AutoDocs of `IdHardware()`.

These functions are available since Release 2:

`ID_ExpName(<manufid>,<prodid>,<result code>)`
: Returns the appropriate result to the manufacturer `<manufid>` (0 to 65535), product `<prodid>` (0 to 255) and the `<result code>`. Note that Identify cannot differ between expansions with the same product ID in this access mode. Result codes are:

`MANUF`	Manufacturer name
`PROD`	Product name
`CLASS`	Product class (localized)

These functions are available since Release 4:

`ID_LockCX()`
: Fetches a copy of all currently present commodities and returns a slot for it. You must always provide this slot to the other functions.

`ID_CountCX(<slot>)`
: Results is the number of commodities found.

`ID_GetCX(<slot>,<nr>,<result code>)`
: Returns the appropriate result to the commodity `<nr>` of the slot `<slot>`. Result codes are:

`NAME`	Name of the commodity
`TITLE`	Title of the commodity
`DESC`	A short description
`GUI`	Result is '`1`' if the commodity provides a GUI, '`0`' otherwise.
`ACTIVE`	Result is '`1`' if the commodity is active, '`0`' otherwise.

`ID_UnlockCX(<slot>)`
: Frees the slot. You must not use it after that!

These functions are available since Release 6:

`ID_Update()`
: Actualizes the hardware information. Please use it wisely (see AutoDocs).

Some example programs you'll find in the `arexx` drawer.

4 InstallIfy

Since V9.0, a new tool has been added.

InstallIfy allows you to import identify's hardware information into Installer scripts.

So you can find out in an Installer script, if e.g. a PowerPC is available, what PowerPC OS is used, or if sufficient processor power is available.

InstallIfy should be invoked with the installer's RUN function. The name for the desired hardware information is provided, and the appropriate numerical result is returned as DOS return code. For example:

```
(set rc (run "c:InstallIfy POWERPC" (safe)))
(if (<> rc 0)       ; PowerPC
  (message "A PowerPC is available!")
)
```

These options are available:

`FIELD` Here, the name of the desired hardware field is passed. This are the same as for the ARexx `ID_Hardware` command. See the include files to find out how the return code has to be interpreted.

`UPDATE` If this option is set, the `identify.library` hardware information cache will be flushed. Use this option wisely, see `ListExp`.

`HELP` Shows a short help page and exits.

There is an example Installer script in the developer packet. Just have a look at it and see how InstallIfy is used.

Concept Index

Table of Contents