NIB V1.00

Loki/SatanicDreams

COLLABORATORS			
	TITLE : NIB V1.00		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Loki/SatanicDreams	October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

NIB V1.00

1.1 Welcome to the NIB Docs...

```
                NEW ICONS BLITZ - V1.00 - (C) ROBERT HUTCHINSON 27/04/1999
[Loki/SatanicDreamsSoftware 1999]
NIB, the complete replacement for the RI icon.library commands. Very easy
to use system which allows DIRECT newicon.library support within your
applications. With commands for standard and newicon images.

1.00  Introduction
 1.01
               Description
              ........................WTF is this then?
 1.02
               Disclaimer
              ...................In the event of NIB-GBH
 1.03
               Requirements
              ............................What y'al need!
 1.04
               IMPORTANT NOTES
              ..................MAKE SURE YOU READ THIS!
 1.05
               Library Number
              ..........................Blitz lib number
 1.06
               Licence
              ........................How can I use NIB
 1.07
               Author
              ..............................That be meee!

2.0  Installation
 2.01
               Where to copy stuff
              ......................Check this first dewd
 2.02
               Your DefLibs
              ......................What are those then?
 2.03
               Conflicting LibNums?
```

NIB (C) Rob Hutchinson 1999 - http://www.satanicdreams.freeserve.co.uk/
NewIcons (C) Team NewIcons - http://www.amiganet.org/NewIcons/

1.2 Description

Description of NIB

NewIconsBlitz's purpose is to allow Blitz programmers the ability to directly access the newicon.library to display NewIcon and Standard Amiga icons in an easy and friendly way.

The beauty behind accessing the newicon.library directly is that your application's user doesn't have to be running the 'NewIcons' program to see the NewIcons. And NIB handles icons in an object orientated way. To save hastle and confusion.

NIB (newicon.library) can be used for all manner of things, from simply displaying icons in a window, to using it as a simple but extremely effective way of loading auto screen-remapped Graphical User Interface elements. Because of the speed of chunk image data remapping, the newicon.library is perfect for such things.

1.3 Disclaimer

Disclaimer

This software is provided as-is, without warranty of any kind, either expressed or implied. In no event will the author(s) be liable for direct, indirect, incidental or consequential damages or data loss resulting from the use or application of this software. The entire risk as to the results and performance of this software is assumed by the user!

1.4 Requirements

Requirements

NIB Requires:

  BlitzBasic2.1+   (Well duh :))
  newicon.library1 (Within BlitzLibs:AmigaLibs/ And remade DefLibs.)

Your Executables will require to run:

  icon.library   (Standard Commodore)
  newicon.library (For newicons [Version 4+ perferably])

Not too much to ask? I'm not 100% sure from the NI documents if the
newicon.library is distributable with your applications or not.
But I still advise you to take it up with the author:

  Eric Sauvageau: merlin@linux.tc3net.com

1.5 NOTES!

IMPORTANT NOTES

o Images or icons should NOT be freed until you have finished with them!
  IE, if you have loaded a newicon, remapped it and drawn it to a window,
  DO NOT free the images UNTIL that window has either been closed or
  the image overwritten (you have finished with the image.)
  The reason for this is because the newicon.library allocates pens
  from your selected screen, and if you free the images when they are
  still 'on-screen', the images could change colour. YUK!

o LOGICAL ORDER OF NEWICON DISPLAY:

    1, Load icon
    2, Check structure for held images
    3, Remap chunky data to the screen
    4, Draw image(s) to window
    5, Free images
    6, Free icon

o NIB uses Blitz Window objects, but wouldn't take 5 minutes to
  convert to NCS window or the like.. And you might see this in
  next version of NIB.

o AmigaLibs.RES, AmigaLibsII.RES or AmigaLibsIII.RES (NCS) should
  be resident in the compiler menu for you program to function.

1.6 Library Number

                LIBRARY NUMBER

Blitz uses library numbering for different librarys, and the

```
newicon.library1 has lib number 208 assigned to it. This has
been allocated with the official blitz library number list.
Although the chances are this lib number wont clash with any
others (2 libraries using the same number). You are advised
to use DefLibMan (on aminet) to check libraries for duplicate
library numbers.

If you are having any problems with the library number, see
the
                Conflicting LibNums?
                section.
```

1.7 Licence

```
LICENCE

Of course there is no licence for using NIB in your programs,
though I would like to have an email from you if you use NIB,

  Me: loki@sdsoft.freeserve.co.uk

I have no idea what the distribution regulations are on the
newicon.library itself, and whether you are allowed to
distribute the newicon.library with your own applications:

  Eric Sauvageau: merlin@linux.tc3net.com

The docs arnt to clear on the distribute of the library.
Well I couldn't find anything :)
```

1.8 Author

```
AUTHOR

NIB is copyright Robert Hutchinson of SDSoftware 1999,

  loki@sdsoft.freeserve.co.uk

Please send any bug reports, changes to the above address,
also if you are having ANY problems with NIB I would be
only too happy to help ;)

Any other questions regarding the newicon.library should be
directed to the NewIcons Team,

   Eric Sauvageau: merlin@linux.tc3net.com
   Phil Vedovatti: vedovatt@u.washington.edu
     Ariel Magnum: arielmag@actcom.co.il
  Nicola Salmoria: MC6489@mclink.it
```

1.9 Where to copy stuff

```
                INSTALLATION

Step 1, Copy the 'NewIconsBlitz/NIBBlitzLibs/AmigaLibs/newicon.library1'
        file to 'BlitzLibs:AmigaLibs/'

Step 2, Open DefLibMan (AmiNet(Dev/Basic)), and check to see if the
        newicon.library1's library number (208) is highlighted in white.
        If this is the case, see the
               Conflicting LibNums?
               section.
        Else Select 'All' and 'Make DefLibs'.

Step 3, Open Blitz and then open the file:
        NewIconsBlitz:NIBIncludes/NIB_Include.BB2
        If this file has ?????'s (bad tokens) in it, or wont compile
        straight through until termination, see the
               Bad Tokens
                        section. (BTW - Make sure AmigaLibs.RES is resident in  ↩
                           the compiler
        menu!)

And should be ready for use :)
```

1.10 Your DefLibs

```
YOUR DEFLIBS

Your deflibs are kept in your Blitz2: directory. This file contains
all your library token offsets, and library settings.

Your deflibs can be remade with the DefLibMan program on aminet
(Dev/Basic), everytime you install/uninstall a new library you
should remake your deflibs.

You should also check for duplicate library numbers in DefLibMan
before selecting 'All' and 'Make DefLibs'.
```

1.11 Conflicting LibNums?

```
CONFLICTING LIBNUMS

If you have confilicting library numbers, there are 2 ways to solve
this problem:

1, Get a copy of the NewIconV4 package of either aminet,
   http://www.amiganet.org/NewIcons/ or AFCD/CUCD. Then
   convert the FD file in Developers/Include/FD .
   (Preferably with NewFDConvert from aminet Dev/Basic)
   Giving it a free lib number.
```

```
2, Fire up a copy of DiskModTools, AZap or similar HexEditor.
   Load in the newicon.library1 file and search for the hex
   value of 208 ($D0) changing it to another (free) lib
   number (in hex of course, hex values can be converted
   easily with the calculator in blitz!).

   NOTE: This method isn't always 100% effective, and can cause
         bad tokens (incorrect offsets) so if you STILL have
         problems after changing this lib num and re-creating
         your deflibs, delete the newicon.library1 file and
         use option number 1 :)
```

1.12 Bad Tokens

```
BAD TOKENS

If you are getting bad/incorrect tokens within Blitz's editor.
And the lib number is DEFINATLY *NOT* conflicting, then have
a look at the ASCII versions of the includes and examples in
the ASCII/ directories. Then load them into blitz and save
them out as BB2 files.
```

1.13 Include.

```
NOTES ABOUT THE INCLUDE

Your program should have the followind include at the top:

  INCLUDE "<Path>/NewIconsBlitz/NIBMulti/NIB_Include.BB2"
  ; Where <Path> is the directory the NIB package is located.
  ; Could be Blitz2: ?

Then your program will have access to the NIB commands.
```

1.14 NIB_ExampleNIB.BB2

```
NIB_EXAMPLENIB.BB2

This example is held within NewIconsBlitz/NIBExamples/
and shows you exactly how to load specific NI data and display
it in a window with the NIB commands. If no NI data is found
in the icon, the program terminates.

NOTE: You may have to change the location of the icon to load
      and the include file path!
```

1.15 NIB_ExampleSTD.BB2

```
NIB_EXAMPLESTD.BB2

This example is held within NewIconsBlitz/NIBExamples/
and shows you exactly how to load specific STD data and display
it in a window with the NIB commands. If no STD data is found
in the icon, the program terminates.

 BTW: STD is just standard, IE 4Col/MWB icons.

NOTE: You may have to change the location of the icon to load
      and the include file path!
```

1.16 NIB_ExampleTCHK.BB2

```
NIB_EXAMPLETCHK.BB2

This example is held within NewIconsBlitz/NIBExamples/
and shows you exactly how to check for different types of icon
data held within a .info file. Then displays the NI data if found,
and the STD data otherwise in a window with the NIB commands.
You can override the NI display of the icon in this example by
setting #NIB_OVERRIDE to 1.

NOTE: You may have to change the location of the icon to load
      and the include file path!
```

1.17 Updates to NIB

```
UPDATES

You can find the latest version of NIB on AmiNet (dev/basic)
or our site: http://www.satanicdreams.freeserve.co.uk/
```

1.18 History

```
HISTORY

1.00 - First release (This version)
```

1.19 Future

```
FUTURE

The future of NIB is unknown, as to whether or not I will update it
depends on if I update it personally. I will be using it in
```

```
IconExtreme3.05 (out as mailware soon!). If I update the routines
for any reason (bugs, improvements) whilst finishing work on IE305
then I shall re-release the updated version.

Things I'd like to implement:

  o Easy Standard/NewIcon data stipping.
    So you can remove the STD or NI data from the structure and
    save the icon out. Effectivly removing the specific icon type
    from the outputted object.

  o IconToBitmap (Blitz bitmap), This I WILL implement ;)..
    Soon enough.

  o IconToShape, images to a Blitz shape object

  o IconToBrush, saving of an Icon image to a brush. Simple?

  o Transparent drawing.
```

1.20 NIBOpenIcon{}

```
NAME:
   NIBOpenIcon{}

SYNOPSIS:
   succ.b=NIBOpenIcon{Icon#,IconPath$}
   This command is a function!

FUNCTION:
   NIBOpenIcon{} loads an icons data into memory, from disk.
   The icon can be loaded from disk with or without the
   the '.info' suffix. If you do add the .info suffix it will
   get stripped by the command. It is probably faster to leave
   the '.info' suffix off if you can.

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM. To load the icon
               data into. Note #NIB_MAXIMUM can be changed at the
               top of the include. If you use more than the
               set maximum, the chances are you will get
               'Array subscript out of range' errors from the
               debugger.
   IconPath$ = A .info file on disk to load.

RETURNS:
   True for success or False for fail. If the command fails
   it almost certainly means the icon doesn't exist or there
   wasn't enough memory to load it.
```

1.21 NIBCheckStruct{}

```
                NAME:
   NIBCheckStruct{}

SYNOPSIS:
   imgs.b=NIBCheckStruct{Icon#}
   This command is a function!

FUNCTION:
   NIBCheckStruct should always follow the
              NIBOpenIcon{}
               command!
   It is used to check the structure of the icon Icon#.
   This to find out if there is NewIcon image data in the icon.
   If the function returns 0. You should then proceed to use

              STDCheckStruct
               to see how many standard images are
   held within the structure.

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.

RETURNS:
   0 = No NI data.
   1 = Normal image only.
   2 = Selected image and normal image.
```

1.22 STDCheckStruct{}

```
                NAME:
   STDCheckStruct{}

SYNOPSIS:
   simgs.b=STDCheckStruct{Icon#}
   This command is a function!

FUNCTION:
   STDCheckStruct{} works in the same way as
              NIBCheckStruct{}
               except
   it checks the structure of the STD data in the icon Icon#

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.

RETURNS:
   0 = No STD data.
   1 = Normal image only.
   2 = Selected image and normal image.
```

1.23 STDFindWidth{}

```
                NAME:
   STDFindWidth{}

SYNOPSIS:
   iconwid.w=STDFindWidth{Icon#}
   This command is a function!

FUNCTION:
   STDFindWidth{} will return to you in a WORD type variable the
   width of the icon Icon#'s STD data. Loaded with
            NIBOpenIcon{}
            .

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.

RETURNS:
   A WORD containing the width of the loaded STD data.

SEE ALSO:

            STDFindHeight{}
```

1.24 STDFindHeight{}

```
                NAME:
   STDFindHeight{}

SYNOPSIS:
   iconhei.w=STDFindHeight{Icon#}
   This command is a function!

FUNCTION:
   STDFindHeight{} will return to you in a WORD type variable the
   height of the icon Icon#'s STD data. Loaded with
            NIBOpenIcon{}
              INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.

RETURNS:
   A WORD containing the height of the loaded STD data.

SEE ALSO:

            STDFindWidth{}
```

1.25 STDToImages{}

```
                NAME:
   STDToImages{}

SYNOPSIS:
   STDToImages{Icon#}
   This command is a statement!

FUNCTION:
   STDToImages{} is used to initialize the Standard image data.
   This should always be called before
             STDDrawImage{}
             , and
   only needs to be called once per loaded icon!

   NB: Remember no remapping of STD icons is done!

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.

SEE ALSO:

              STDDrawImage{}
```

1.26 STDDrawImage{}

```
NAME:
   STDDrawImage{}

SYNOPSIS:
   succ.b=STDDrawImage{Icon#,Type,Window,X,Y}
   This command is a function!

FUNCTION:
   This command will draw the initialized STD data to a specified
   window. Both the Normal and Selected images have to be drawn
   individually. (IE, call the command twice.)

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
   Type   = The type of image to draw:
               #NIB_NORMAL   - Draws the normal image data.
               #NIB_SELECTED - Draws the selected image data.
   Window = The window number to draw the image to.
   X      = The X position in the window to draw the data at.
   Y      = The Y position in the window to draw the data at.

RETURNS:
   True or False. This command will only return false if
   STDToImages{} hasn't been run before-hand, you have no
   icon loaded or there was a constant other than the above
   passed to the function.

NOTES:
```

```
   Dont rely on the failsafe!
```

1.27 NIBRemapChunkyData{}

```
                NAME:
   NIBRemapChunkyData{}

SYNOPSIS:
   NIBRemapChunkyData{Icon#,Type,Screen}
   This command is a statement!

FUNCTION:
   This command will remap the loaded chunky image data ready for
   you to draw to a window with
             NIBDrawImage{}
             .

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
   Type   = The type to remap:
              #NIB_NORMAL   - Remap only the normal image data.
              #NIB_SELECTED - Remap only the selected image data.
              #NIB_BOTH     - Remap both types (as above).
   Screen = The number of the screen you with to remap the data
            to. (Found with commands such as Screen, FindScreen
                 and WBToScreen).

SEE ALSO:

             NIBDrawImage{}
```

1.28 NIBDrawImage{}

```
NAME:
   NIBDrawImage{}

SYNOPSIS:
   succ.b=NIBDrawImage{Icon#,Type,Window,X,Y}
   Command is a function!

FUNCTION:
   This command will draw the initialized NI data to a specified
   window. Both the Normal and Selected images have to be drawn
   individually. (IE, call the command twice.)

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
   Type   = The type of image to draw:
              #NIB_NORMAL   - Draws the normal image data.
              #NIB_SELECTED - Draws the selected image data.
   Window = The window number to draw the image to.
```

```
   X      = The X position in the window to draw the data at.
   Y      = The Y position in the window to draw the data at.

RETURNS:
   True or False. This command will only return false if
   NIBRemapChunkyData{} hasn't been run before-hand, you have no
   icon loaded or there was a constant other than the above
   passed to the function.

NOTES:
   Dont rely on the failsafe!
```

1.29 NIBFreeImages{}

```
                NAME:
   NIBFreeImages{}

SYNOPSIS:
   NIBFreeImages{Icon#,Type,Screen}
   This command is a statement!

FUNCTION:
   This command frees the remapped chunky images from memory,
   and should only be used after you have finished with your icon,
   as it frees the pens on the screen. Screen should also be
   the same as specified in
              NIBRemapChunkyData{}
              .
   If you free the imagedata and you still have the image on the
   window, the image may change colour!

   If your screen has since been closed, use:
   FreeRemappedImage_ *NIBImage(<Icon#>*2),0 ;for normal image freeing
    and
   FreeRemappedImage_ *NIBImage((<Icon#>*2)+1),0 ;for selected image freeing

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
   Type   = The type of image to free:
               #NIB_NORMAL   - Frees only the normal image.
               #NIB_SELECTED - Frees only the selected image.
               #NIB_BOTH     - Frees both images.
   Screen = The screen you remaped the images to.
```

1.30 NIBFreeIcon{}

```
NAME:
   NIBFreeIcon{}

SYNOPSIS:
   succ.b=NIBFreeIcon{Icon#}
```

```
   This command is a statement!

FUNCTION:
   This will free a loaded icon, and should be called after
   you have finished with your icon. Regardless of whether you
   have use STD or NI images.

   The images should be freed with the relevent commands
   before calling this command.

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
```

1.31 NIBFindWidth{}

```
NAME:
   NIBFindWidth{}

SYNOPSIS:
   nibwid.w=NIBFindWidth{Icon#}
   This command is a function!

FUNCTION:
   This command returns to you the width of the NI image in
   the icon Icon#`s data. It can be called before or after the
   remap commands.

RETURNS:
   A WORD value containing the width of the NI image.

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
```

1.32 NIBFindHeight{}

```
NAME:
   NIBFindHeight{}

SYNOPSIS:
   nibwid.w=NIBFindHeight{}
   This command is a function!

FUNCTION:
   This command returns to you the height of the NI image in
   the icon Icon# data. It can be called before or after the
   remap commands.

RETURNS:
   A WORD value containing the height of the NI image.

INPUTS:
   Icon#  = A number from 0 to #NIB_MAXIMUM.
```

1.33 NIBSaveIcon{}

```
NAME:
   NIBSaveIcon{}

SYNOPSIS:
   succ.b=NIBSaveIcon{Icon#,FilePath$}
   This command is a Function!

FUNCTION:
   This command will save back to disk an icon file. Encoding the
   chunky image data into the tooltypes of the icon if it
   is availible. This rountine will save back any availible data.
   Simply if the data has been freed and the pointers reset
   the command will save only the STD data (and visa-versa)

INPUTS:
   Icon#     = A number from 0 to #NIB_MAXIMUM.
   FilePath$ = Path of the .info file to save to on disk.

RETURNS:
   True or False. If it fails the chances are it's because
   the device/path doesn't exist.

NOTES:
   I havent 100% tested this command, but is should work fine!
```

1.34 NIBCheckLibs{}

```
NAME:
   NIBCheckLibs{}

SYNOPSIS:
   succ.b=NIBCheckLibs{}
   This command is a Function!

FUNCTION:
   This command despite being at the bottom of the guide is the
   most important one you have availible! It checks for the
   newicon.library and icon.library, both required by your
   executable!!

RETURNS:
   True or False. If it fails, this means the user doesn't have
   either V40+ of the newicon.library or V39+ of the icon.library
   and your program should promptly terminate!

NOTES:
   I haven't checked the lowest possible version of the icon.library
   the user must have in order for you to use STD icons.. So
   if you have a version lower than V39 and it still works change
   the version it checks for in the include.
```

1.35 Greets to

```
GREETS GO OUT TO:

All the members of SDS:
  HawkEye, HotCakes, GazChap, EcheXiO, Scider, Pagman, aMIGA_dUDE,
  mULDERfBI, Budda, BootBlock, Fanta, Ribs, Labbe, Kev, Wrecker
  and last but not least Phil Price.

Members of the BlitzList:
  David McMinn, Rui Carvalho, Rick Hodger, Donnovan Reeve, Liz Tucker,
  Simon Hitchen, James.L.Boyd, S.BeardWood, Curt Esser, Amorel,
  Paul West, Anders Hasselqvist, Simon Archer, Paul Burkey,
  Anton Reinauer, Chris "PrinceD" Deeney, Tony Rolfe, Julian Kinraid,
  Jamie Solomons, Peter Price, Dave "C00lie", Stefan Lebed, Peter Thor,
  Chris Jarvis, John Mason and Scott.

  (I GOTTA have missed someone there?)

Other Peeps,
  Materia_Keeper, Fairs, Ben Vost,

People keeping the Amiga BBS scene alive:
  Leon (D-Tracker BBS) [Humm, kinda says summat dont it? ahahah]

Sorry if I forgot ppl ;)
```

1.36 Thanks to

```
THANKS GO OUT TO:

Whoever emails me telling me they are using NIB (But doubt anyone
will, will they?? Go on, you know you want too! ;))

David McMinn, for some additional help with rountines I haven't
yet incorporated. And for getting me into all this OS stuff.
Big thanks. TA.

Acid and RWE for Blitz2.

My lovely Miggy :)

And OF COURSE, Team NewIcons, whom without this wouldn't have
been in anyway possible :)
```