08326e38-0

COLLABORATORS

	TITLE : 08326e38-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 31, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

08326e38-0

1.1 No title

MED libraries for HiSoft Pascal (v1.1)

by

Daniel~Mealha~Cabrita

1 november, 1997

1.2 what's this?

Well.. HS-Pascal came supporting only standard OS 2.x libraries so there isn't much choices for playing music:

- To use ready-made MOD player asm routines inside your Pascal program. (hardware banging is not my style..)

- Develop your own player routines (some people likes to reinvent the wheel).

- Did i forgot any other way?

There was a ready-made library for playing MED modules, the medplayer.library that makes all the playing job.. but it had support only for C programs (and i simply don't like the C language itself).

So i wrote my own Pascal support for medplayer.library, following the exatly same implementation style that you know with all other OS libraries supported by HiSoft.

What you're waiting for?.. Let's make some noise using Pascal!

1.3 DISCLAIMER

I, the author of medplayer.library support for HS-Pascal exclude myself from ANY responsability from the consequences (direct or indirect) of using my code.

I'm not affiliated with authors of MED, medplayer.library and relationed products.

I'm not affiliated with HS-Pascal author or HiSoft.

I ONLY authorize the using of this product for particular purpouses. For other purpouses, contact the author.

*** USE THIS CODE I'VE WRITTEN AT YOUR OWN RISK ***

Using my code in your program means acceptance of this terms.

Words about DISTRIBUTION:

I authorize this file being available online at Aminet, Aminet CDs and FredFish CDs or disquettes.

This file can be distributed from one particular to another too.

This file can be included in magazines' companion disk or CDs ONLY IF i receive the respective issue and its companion disk(s) or CD(s) (i strongly prefer the CD).

For other ways of distribution, contact the author for negotiating permission.

1.4 medplayer.library manual

```
*************************************************************************
*   Instructions for using "medplayer.library" V2, by Teijo Kinnunen     *
*************************************************************************

"medplayer.library" is a shared library, which can be used to load and play
MED/OctaMED  4-channel modules You can call its functions from any language
which supports library calls (C, Assembler, Basic...) (Note:  All functions
must be called by the same task that opened the
        library (using GetPlayer()). The library uses the task pointer
        to keep track of the current user of the library.)

First  you  must  install  "medplayer.library" to your LIBS:  drawer (click
Install_Libraries  to  do that).  You can also load it with ARP's "loadlib"
command.

The main advantage of using medplayer.library instead of modplayer.a player
routine  is  that you won't have to modify your program when new MED player
routines   are   released.    The   user   will   just   have   to   replace
medplayer.library  and  ZAP...   (sorry,  ZAP ;-) your program will support
HexaMED  V63.132  (or  whatever  :-)  modules...   At the moment of writing
(1.1.1992)  I  have been working on OctaMED Professional, which will have a
new  module  format  incompatible  with  MMD0  (it'll be MMD1).  If you use
medplayer.library, your programs will be compatible with MMD1s.

There's  a  header  file  'libproto.h'  that  contains  the  prototypes and
#pragmas for use with SAS/Lattice C V5.

Here's  the  complete  list of the functions in "medplayer.library" (in RKM
autodoc-style):

-------------------------------------------------------------------------
-------------------------------------------------------------------------

GetPlayer

NAME
        GetPlayer -- get and initialize the player routine

SYNOPSIS
        error = GetPlayer(midi)
        D0                D0

FUNCTION
        This routine allocates the audio channels and CIAB timer A/B
        and prepares the interrupt. If "midi" is nonzero, serial
        port is allocated and initialized. You should call this
        routine when your programs starts up.

INPUTS
        midi =  0 no midi, 1 set up midi. When you use a song that
                has only Amiga samples, there's no reason to allocate
                the serial port. Then set midi to 0.

RESULT
```

```
        If everything is OK, GetPlayer() returns zero. If initialization
        failed or somebody else is currently using the library, then
        GetPlayer() returns nonzero value.
        NOTE: Even if GetPlayer() returned an error, you can still call
              the library functions without making harm. They just won't
              work (except LoadModule(), UnLoadModule(), RelocModule()
              and GetCurrentModule(), which always work).

SEE ALSO
        FreePlayer

------------------------------------------------------------------------------
------------------------------------------------------------------------------

FreePlayer

NAME
        FreePlayer -- free the resources allocated by GetPlayer()

SYNOPSIS
        FreePlayer()

FUNCTION
        This routine frees all resources allocated by GetPlayer().
        Remember always call this routine before your program
        exits. It doesn't harm to call this if GetPlayer() failed.
        If you don't call this function during exit, audio channels,
        timer etc. will remain allocated until reboot.

SEE ALSO
        GetPlayer

------------------------------------------------------------------------------
------------------------------------------------------------------------------

PlayModule

NAME
        PlayModule -- play module from the beginning

SYNOPSIS
        PlayModule(module)
                   A0

FUNCTION
        This routine starts to play the module from the beginning.
        The module can be obtained by calling LoadModule() or it can
        be incorporated directly into your program. The module has to
        be relocated before calling PlayModule()!

INPUTS
        module = pointer to the module.

SEE ALSO
        ContModule, StopPlayer

------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------

ContModule

NAME
        ContModule -- continue playing the module from where it stopped

SYNOPSIS
        ContModule(module)
                   A0

FUNCTION
        ContModule() functions just like PlayModule() except if you
        have stopped playing with StopPlayer(), the playing will
        continue where it stopped. When you play the module first
        time, you should use PlayModule().

INPUTS
        module = pointer to module.

SEE ALSO
        PlayModule, StopPlayer

------------------------------------------------------------------------------
------------------------------------------------------------------------------

StopPlayer

NAME
        StopPlayer -- stops playing immediately

SYNOPSIS
        StopPlayer()

FUNCTION
        Stop.

SEE ALSO
        PlayModule, ContModule

------------------------------------------------------------------------------
------------------------------------------------------------------------------

DimOffPlayer

NOTE
        This is an obsolete function! It was removed in V2 of the library,
        and it now just does StopPlayer()!

------------------------------------------------------------------------------
------------------------------------------------------------------------------

SetTempo

NAME
        SetTempo -- modify the playing speed
```

```
SYNOPSIS
        SetTempo(tempo)
                 D0

FUNCTION
        If you want to modify the playback speed, you can call this one.
        The tempo value should be 1 - 240. Note that tempos 1 - 10 are
        recognized as SoundTracker tempos. This function usually has no
        use.

INPUTS
        tempo = new tempo

------------------------------------------------------------------------
------------------------------------------------------------------------

LoadModule

NAME
        LoadModule -- load a MED module from disk and relocate it

SYNOPSIS
        module = LoadModule(name)
        D0                  A0

FUNCTION
        When you want to load a module from disk, call this function.
        The function loads only MED modules (MMD0). It doesn't load
        Tracker-modules, MED songs or object files. Only MMD0's
        (MMD0 is the identification word at the beginning of the file).
        Because the module contains many pointers, they must be
        relocated. This function relocates the module automatically.
        If you include the module as a binary file converted with
        Objconv, YOU must relocate it. This is an easy thing to do.
        Just call RelocModule().

INPUTS
        name = pointer to file name (null-terminated)

RESULT
        module = pointer to module. If failed to load for some reason
                 (disk error, out of memory, not a module), zero will
                  be returned.

SEE ALSO
        UnLoadModule

------------------------------------------------------------------------
------------------------------------------------------------------------

UnLoadModule

NAME
        UnLoadModule -- frees the module from memory

SYNOPSIS
        UnLoadModule(module)
```

```
                A0 FUNCTION
        When you don't need the module anymore, you MUST free the
        memory it has used. Use this routine for it. Remember to
        stop the player before unloading the module it is playing.

        NOTE: unload only those modules which are loaded with
        LoadModule(). If you attempt to free module which is a part
        of the program, you will cause guru 81000009/81000005.

INPUTS
        module = pointer to module. If zero, nothing happens.

SEE ALSO
        LoadModule

------------------------------------------------------------------------------
------------------------------------------------------------------------------

GetCurrentModule

NAME
        GetCurrentModule -- returns the address of module currently playing

SYNOPSIS
        module = GetCurrentModule()
        D0

FUNCTION
        Simply returns the pointer of the module, which is currently
        playing (or if player is stopped, which was played last). This
        works also if some other task is currently playing. In this case,
        because of multitasking, you should have no use for the value
        (the module can be already unloaded). You may ask what use this
        function has. Well, I'm not sure, but because this function
        takes only 2 machine language instructions (8 bytes of memory)
        there's not much harm of it.

RESULT
        module = pointer to current module

------------------------------------------------------------------------------
------------------------------------------------------------------------------

ResetMIDI

NAME
        ResetMIDI -- reset all pitchbenders and modulation wheels and
                     ask player to resend the preset values

SYNOPSIS
        ResetMIDI()

FUNCTION
        This function resets pitchbenders and modulation wheels on all
        MIDI channels. It also asks the player to send again the
        preset change requests for all instruments, so that the presets
        will be correct if the user has changed them. It performs the
```

```
        same function as MED's Ctrl-Space.

------------------------------------------------------------------------------
------------------------------------------------------------------------------
NOTE: THE FOLLOWING FUNCTIONS ARE ONLY AVAILABLE IN MEDPLAYER.LIBRARY V2 OR
LATER, BE SURE THAT YOU'RE REALLY USING V2 OF THE LIBRARY e.g. MEDPlayerBase
= OpenLibrary("medplayer.library",2);
------------------------------------------------------------------------------
------------------------------------------------------------------------------

SetModnum

NAME
        SetModnum -- select the number of the song (in multi-module)

SYNOPSIS
        SetModnum(modnum)
                  D0

FUNCTION
        Use this function to set the number of song you want to play.
        Call this before PlayModule(). 0 is the first song, 1 is the
        second, and so on. If the module is not a multi-module, this
        function has no affection.

------------------------------------------------------------------------------
------------------------------------------------------------------------------

RelocModule

NAME
        RelocModule -- relocate module

SYNOPSIS
        RelocModule(module)
                    A0

FUNCTION
        This function relocates the module. It should be used if you've
        incorporated the module with Objconv program before using the
        module. Note that LoadModule() automatically relocs everything.

------------------------------------------------------------------------------
******************************************************************************
```

1.5 about the author


```
 Daniel Mealha Cabrita
 dancab@base.com.br
 dancab@polbox.com  - as second option (try to avoid this)
 http://www.geocities.com/SiliconValley/Park/8342/

 sorry.. no snail mail address provided.

 Curitiba - PR
```

BRAZIL

1.6 some instructions

First of all: you MUST know how to deal with AmigaOS-style units in HS-Pascal; i'll not teach this here for you.

You'll need:

- Amiga, HS-Pascal with its default OS units.

- medplayer.library V1+ (V2+ is recommended)

- MedPlayer.unit must be on HSPUnits:Units directory

After opening medplayer.library you must provide its pointer to MedPlayerBase variable.

Before anything, you must make medplayer.library to allocate audio channels using GetPlayer function (and later, closing it using FreePlayer).

There's some functions that only works with medplayer.library V2+ and it will bring unpredictable results if called with V1.

Due to its obsolence, the 'DimOffPlayer' library function is NOT implemented on this unit (what for?).

For now there's NO support for eventually new functions provided by V3+ of medplayer.library (can you send the docs???).

Remember closing everything you opened before program exit.

And YES.. you must compile this unit before using it! (generating the .unit file and placing it into HSPUnits:Units).

1.7 acknowledgements

MED, medplayer.library and relationed products
© 1992-97 Teijo Kinnunen & Ray Burt-Frost

HSPascal (written by Keith Wilson)
Copyright © 1994 HiSoft, D-House & Christen Fihl

Amiga®, AmigaOS® and relationed products are trademarks of Amiga International Inc.