thxplay

COLLABORATORS			
	TITLE : thxplay		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		May 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

thxplay

1.1 thxplay.doc

```
--overview--
thxFree()
thxGetNumSongs()
thxGetVolume()
thxInit()
thxNoteFX()
thxPause()
thxPlay()
thxPlayNote()
thxPlaytime()
thxSetSong()
thxSetVolume()
thxSignalEnd()
thxSongEnded()
thxStop()
thxStopNote()
thxSyncByte()
thxWind()
```

1.2 thxplay.library/--overview--

```
PURPOSE
    To provide an interface to the THX2 player.

OVERVIEW
    THX2 is a 'chip' music tracker by Martin Wodok (Dexter/Abyss). It
    comes with a rather cumbersome binary replayer, so you may play THX2
    songs in your own programs. This provides an easy and powerful
    interface to the THX2 player, providing a wide range of functions.

     - Allocation: thxInit(), thxFree()
     - Playing:    thxPlay(), thxStop(), thxPause(), thxWind()
     - Volume:     thxGetVolume(), thxSetVolume()
     - Multisong:  thxGetNumSongs(), thxSetSong()
     - Sound FX:   thxPlayNote(), thxStopNote(), thxNoteFX()
```

```
 - Misc:       thxSignalEnd(), thxSongEnded(), thxSyncByte(),
               thxPlaytime()
```

You need to read a THX song in from disk or 'incbin' it. You should load it into PUBLIC memory, however it does not have to be CHIP memory.

The music play is, as you would expect, interrupt-driven, and asynchronous. This interface automatically provides fallback support for a VSYNC based replayer if it cannot allocate a CIA timer.

The interface is 68000 compatible, optimised versions for the 68020 and better are also included.

NOTE

This interface was initially developed as an Amiga E module. With a little extra effort, it is also available as a runtime shared library. Therefore, it operates a simple mechanism in thxInit() and thxFree() to ensure only one task at any time is using the library. See the notes of thxInit() for more information about this.

Also note that all examples are given in Amiga E code.

Synopsis is given as 3 lines: the assembler/register synopsis, the C prototype, and the E synopsis.

EXAMPLE

More thorough examples are included with the distribution.
This is an example in Amiga E, using the E module thx-play.m.

```
MODULE 'tools/thx-play', 'tools/file'
PROC main()
  DEF mod
  IF mod := loadfile(arg, 0, MEMF_PUBLIC)
    IF thxInit(mod)=0
      thxPlay()
      REPEAT; WaitTOF(); UNTIL CtrlC() OR thxSongEnded()
      thxStop()

      thxFree()
    ENDIF
    freefile(mod)
  ENDIF
ENDPROC
```

Here is the same example, but using the shared thxplay.library.
Note that Amiga E forces library functions to have capitalised names, and that we must use OpenLibrary() and CloseLibrary().

```
MODULE 'thxplay', 'tools/file'
PROC main()
  DEF mod
  IF thxplaybase := OpenLibrary('thxplay.library', 5)
    IF mod := loadfile(arg, 0, MEMF_PUBLIC)
      IF ThxInit(mod) = 0
        ThxPlay()
        REPEAT; WaitTOF(); UNTIL CtrlC() OR ThxSongEnded()
```

```
            ThxStop()

            ThxFree()
          ENDIF
          freefile(mod)
        ENDIF
        CloseLibrary(thxplaybase)
      ENDIF
    ENDPROC
```

1.3 thxplay.library/thxFree

```
NAME
    thxFree -- free resources held by player.

SYNOPSIS
    void thxFree()

    void thxFree(void);

    thxFree()

FUNCTION
    Stops any THX module playing and frees resources used by the player.
    You can call this whether thxInit() suceeded or not.

SEE ALSO
    thxInit()
```

1.4 thxplay.library/thxGetNumSongs

```
NAME
    thxGetNumSongs -- get number of subsongs.

SYNOPSIS
    songs = thxGetNumSongs()
    D0

    UWORD thxGetNumSongs(void);

    songs := thxGetNumSongs()

FUNCTION
    Returns  the  number  of subsongs in the module, if any. You can use
    the  thxSetSong()  function  to  play one of the subsongs, if that's
    possible.

RESULT
    songs - 0 if there are no subsongs (only the main song), otherwise
            returns the number of subsongs.
```

```
SEE ALSO
    thxSetSong()
```

1.5 thxplay.library/thxGetVolume

```
NAME
    thxGetVolume -- get master volume.

SYNOPSIS
    volume = thxGetVolume()

    UBYTE thxGetVolume(void);

    volume := thxGetVolume()

FUNCTION
    Returns the current master volume value. Does not stop play.

RESULT
    volume - current volume setting from 0 (silent) to 64 (loudest)

SEE ALSO
    thxSetVolume()
```

1.6 thxplay.library/thxInit

```
NAME
    thxInit -- initialise player and module.

SYNOPSIS
    error = thxInit(module)
    D0              A0

    ULONG thxInit(APTR);

    error := thxInit(module)

FUNCTION
    Initialises  the  player (if needed) and initializes the module. You
    may  also  call  thxInit(NIL)  to  initialise the player but not the
    module.  Does not start to play the module until you call thxPlay().
    You  must  call  this each time you want to play a different module.
    The allocations made for the player are made only the first time you
    call  thxInit(), no matter how many modules you want. If allocations
    fail, they will be automatically freed.

INPUTS
    module - pointer to a THX module or NIL

RESULT
    error - zero means all went OK, any other value means failure.
```

```
NOTE
    In  the  library  version of this interface, thxInit() and thxFree()
    use  a task ownership system - to begin with, nobody 'owns' THX. The
    first  task  to  call  thxInit() will then 'own' THX, and successive
    calls  to  thxInit()  and thxFree() will only succeed for this task.
    When  this  task calls thxFree(), the owner goes back to nobody, and
    now other tasks are free to use THX.

    This  is the only arbitration used by the interface. All other calls
    may  be called from any task at all. Please respect this arbitration
    mechanism and avoid calling other THX functions unless thxInit() has
    succeeded  for  you. You must call thxFree() from the same task that
    called thxInit().

SEE ALSO
    thxFree(), thxPlay()
```

1.7 thxplay.library/thxNoteFX

```
NAME
    thxNoteFX -- perform FX command on user-specified channel.

SYNOPSIS
    void thxNoteFX(channel, command, parameter)
                   D0:2     D1:4     D2

    void thxNoteFX(UBYTE, UBYTE, UBYTE);

    thxNoteFX(channel, command, parameter)

FUNCTION
    Performs  an  effect command on the particular channel. You can call
    this  at  any  time,  even before you play the note, if you want the
    note  to  start  off  with an initial effect. See THX Sound System's
    documentation for the full list of commands and their parameters.

INPUTS
    channel   - The channel affected
    command   - the effect command, eg $C is the Set Volume command.
    parameter - the parameter to the command, eg $40 is full volume.

NOTE
    No  validation  of  the  command  or  its  parameter is done. Beware
    feeding wrong or out of range values. Range for command is $0 to $F,
    parameter is $00 to $FF.

SEE ALSO
    thxPlayNote()
```

1.8 thxplay.library/thxPause

```
NAME
    thxPause -- pause play of a song.

SYNOPSIS
    void thxPause()

    void thxPause(void);

    thxPause()

FUNCTION
    Pauses the playing module. Call thxPlay() to continue play again.

SEE ALSO
    thxPlay()
```

1.9 thxplay.library/thxPlay

```
NAME
    thxPlay -- start playing the song.

SYNOPSIS
    void thxPlay()

    void thxPlay(void);

    thxPlay()

FUNCTION
    Starts  playing  the module. If the module has just been initialised
    or  stopped,  or  the  subsong has just been changed, then play will
    start  at  the  beginning  of  the  song/subsong. Otherwise, it will
    continue from where it was paused.

SEE ALSO
    thxStop(), thxPause()
```

1.10 thxplay.library/thxPlayNote

```
NAME
    thxPlayNote -- start playing a user-specified note.

SYNOPSIS
    void thxPlayNote(channel, note, instrument)
                     D0:2     D1    D2

    void thxPlayNote(UBYTE, UBYTE, UBYTE);

    thxPlayNote(channel, note, instrument)

FUNCTION
```

```
    Plays  one of the instruments in the THX module at a particular note
    on  a particular channel. It is up to you to ensure that the channel
    you  play  the  note  on is empty and so will not interfere with the
    note  being  played.  This function is to allow you to play your own
    notes  during  THX  play,  for  example  as  part of a game as sound
    effects.  The  note  played is subject to the same conditions as the
    song itself, such as the global volume control. In addition, you can
    apply  'FX'  commands to the note. In effect, what is happening when
    you  call  thxPlayNote()  is  that  the  'track data' for the chosen
    channel being played is overwritten (not the module itself, just the
    sound output). It is overwritten on the first line by your specified
    instrument  with  note  and  FX,  then  on  consecutive lines by the
    'blank'  note and instrument. This 'overwriting' stops only when you
    call thxStopNote(), or stop the module naturaly.

INPUTS
    channel    - The channel on which the note is played, from 0 to 3.
    note       - The halfnote (pitch) at which the instrument is to be
                 played, from 1 (C-1) to 60 (B-5).
    instrument - an instrument from the song, from 1-63. You should know
                 which instrument you want to play!

EXAMPLE
    thxPlayNote(2, 8, 12) is equivalent to this in THX Sound System's
    tracker view:

    ---00000 | ---00000 | G-112000 | ---00000
    ---00000 | ---00000 | ---00000 | ---00000
    ---00000 | ---00000 | ---00000 | ---00000
    [...]

SEE ALSO
    thxStopNote(), thxNoteFX()
```

1.11 thxplay.library/thxPlaytime

```
NAME
    thxPlaytime -- get current playtime of song.

SYNOPSIS
    seconds [, ticks, tickspd] = thxPlaytime()
    D0         D1     D2

    ULONG thxGetSyncByte(void);

    seconds, ticks, tickspd := thxPlaytime()

FUNCTION
    Gets the current playtime into the play of a currently playing song.

RESULT
    seconds - the number of seconds elapsed since the start of the song.
              it is calculated from the following two results
    ticks   - number of internal clock ticks.
    tickspd - the speed of internal clock ticks in Hz.
```

```
BUGS
     Will wrap at 65536 seconds. Also, due to a bug in the replayer,
     will wrap at 65536 _ticks_ first. This will hopefully be fixed,
     but the seconds limit probably will not be.

NOTE
     Most C compilers will be unable to get the ticks and tickspd
     results. Too bad. They're not that important.
```

1.12 thxplay.library/thxSetSong

```
NAME
    thxSetSong -- set song to be played.

SYNOPSIS
    thxSetSong(song)
               D0

    void thxSetSong(UWORD);

    thxSetSong(song)

FUNCTION
    Sets  which  song  to play, if a module contains more than one song.
    Most  modules  only  contain  one  song,  but  some  modules contain
    sub-songs  as  well  as  the  main one. You can use this function to
    specify  which  one  should be played. If you call this function and
    there is already a song playing, it will be stopped first.

INPUTS
    song - 0 to set the main song to be played, any other number will
           change to that subsong, if it exists. Otherwise, no change
           will be made (other than the stoppage).

NOTE
    It is up to you to start playing the module again.

SEE ALSO
    thxGetNumSongs()
```

1.13 thxplay.library/thxSetVolume

```
NAME
    thxSetVolume -- set master volume.

SYNOPSIS
    void thxSetVolume(volume)
                      D0

    void thxSetVolume(UBYTE);
```

```
    thxSetVolume(volume)

FUNCTION
    Sets the master volume. Does not stop play.

INPUTS
    volume - from 0 (silent) to 64 (loudest)

NOTE
    This  function  can take up to two frames to take an audible effect.
    If the song is paused, will not take effect until unpaused.

SEE ALSO
    thxGetVolume()
```

1.14 thxplay.library/thxSignalEnd

```
NAME
    thxSignalEnd -- Signal() when song ends.

SYNOPSIS
    thxSignalEnd(task, signalset)
                 A0    D0

    void thxSignalEnd(struct Task *, ULONG);

    thxSignalEnd(task, signalset)

FUNCTION
    Asks  THX  to send the signalset to the specified task when the song
    ends.  If songend occurs and the signal is sent, it will not be sent
    again  unless  you  call thxSignalEnd() again to reload the trigger.
    The signal will also be cancelled if you call thxStop() directly, or
    indirectly through thxSetSong() or thxFree().

NOTE
    The detection of songend is crap (sorry Dexter :^)

INPUTS
    task       - pointer  to  a  task  or  process structure, simply use
                 FindTask(NIL) to send to yourself.
    signalset  - a 32bit set of signals, to be sent to task when songend
                 occurs.

EXAMPLE
    thxSignalEnd(FindTask(NIL), SIGBREAKF_CTRL_C) will send you a CTRL-C
    when the song ends.

SEE ALSO
    thxSongEnded(), exec.library/Signal()
```

1.15 thxplay.library/thxSongEnded

```
    thxSongEnded -- detect if song has ended.

SYNOPSIS
    songended = thxSongEnded()
    D0

    BOOL thxSongEnded(void);

    songended := thxSongEnded()

FUNCTION
    Returns  nonzero  value if the player has detected the end of a song
    and is now looping.

NOTE
    The detection of songend is crap (sorry Dexter :^)

RESULT
    songended - nonzero if song is now looping, zero otherwise.

SEE ALSO
    thxSignalEnd()
```

1.16 thxplay.library/thxStop

```
NAME
    thxStop -- stop playing a song/module.

SYNOPSIS
    void thxStop()

    void thxStop(void);

    thxStop()

FUNCTION
    Stops  the  module.  Can  be restarted from the beginning again with
    thxPlay().

SEE ALSO
    thxPlay(), thxFree()
```

1.17 thxplay.library/thxStopNote

```
NAME
    thxStopNote -- stop playing user-specified note.

SYNOPSIS
    void thxStopNote(channel)
                     D0:2
```

```
    void thxStopNote(UBYTE);

    thxStopNote(channel)

FUNCTION
    Stops  anything you started with thxPlayNote(). Please be aware that
    notes  which  don't  fade  away  on  their own will first need to be
    silenced with thxNoteFX($C, $00), or such

SEE ALSO
    thxPlayNote()
```

1.18 thxplay.library/thxSyncByte

```
NAME
    thxSyncByte -- get sync byte value.

SYNOPSIS
    syncvalue = thxGetSyncByte()
    D0

    UBYTE thxGetSyncByte(void);

    syncvalue := thxGetSyncByte()

FUNCTION
    Gets the current setting of the 'external timing' byte, which can be
    set  to any byte value at any moment in time during play of the song
    BY  the  song  itself,  using  the '8' command  in the tracker. This
    function  is  here to allow you to mark specific events in the music
    with   the  '8'  command  and  a  value,  then  wait  until  calling
    thxSyncByte()  returns that value. The returned value doesn't change
    until another '8' command in the song changes it.

NOTE
    Be  very  careful  not  to  busy-wait on a new value if there is the
    possibility the song is paused or not playing.

RESULT
    syncvalue - current value of the sync byte.
```

1.19 thxplay.library/thxWind

```
NAME
    thxWind -- wind the song forward or back.

SYNOPSIS
    void thxWind(direction)
                 D0

    void thxWind(UBYTE);
```

```
    thxWind(direction)

FUNCTION
    Advances forward or backwards through the song by a specified number
    of  positions. Please use the value 1 to skip forward and -1 to skip
    back, for future compatibility.

INPUTS
    direction - if  1, winds on to the next position.
                if -1, winds back to the previous position,
                if  0, ignored.

NOTE
    Be  wary  of  stepping  beyond  the  end  of  a song. Also note this
    function only takes effect only once a frame.
```