Protracker

COLLABORATORS			
	TITLE : Protracker		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 31, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Protracker

1.1 Protracker

```
PureBasic - Protracker Module V1.00

  Protracker is a well known software allowing musicians
  to create complex music using the four Amiga stereo
  audio channels. This library allows the programmer to
  manipulate the Protracker modules very easily. The
  replay is of course OS compliant and perfect whenever
  the refresh rate. This library works toghether with the
  Audio library, so it's possible to play standard sounds
  while a module is playing, very useful for games !
  Main advantages of Protracker modules: compact, can be of
  any lengths, very fast to replay even on slow Amigas.
  This library has been fully rewritten and isn't based on
  third amiga shared libraries.

Commands summary:

   FreePTModule
   GetPTModulePos
   GetPTModuleRow
   InitPTModule
   LoadPTModule
   PausePTModule
   PlayPTModule
   ResumePTModule
   SetPTModulePos
   SetPTModuleRow
   SetPTModuleSpeed
   StopPTModule

   PTModule Demo
```

1.2 freeptmodule

```
  SYNTAX
```

```
FreePTModule(#Module.w)

  STATEMENT
Frees a module object earlier initialized with LoadPTModule().

Module
The module to free.
```

1.3 getptmodulepos

```
  SYNTAX
Result.b = GetPTModulePos()

  FUNCTION
Read the song position from the currently played module.

Result
The currently song position, which ranges from 0 to 127.
```

1.4 getptmodulerow

```
  SYNTAX
Result.b = GetPTModuleRow()

  FUNCTION
Read the row number from the currently played module.

Result
The actual row, range from 0 to 63.
```

1.5 initptmodule

```
  SYNTAX
Result.l = InitPTModule(Modules.l)

  FUNCTION
This is the initroutine and it must always be called
before any other PTModule functions.

Note:
====
The new Audio Lib serves both PTModule Lib and Sound
Lib with the allocation/freeing functions of audio
channels.  The little drawback is that InitAudio()
must always be called before InitPTModule().

Modules
The max module objects wanted.

Result
```

```
If this is TRUE the PTModule environment is correctly
initialized but if it's FALSE instead no other
function should be called.
```

1.6 loadptmodule

```
  SYNTAX
Result.l = LoadPTModule(#Module.w,FileName$)

  FUNCTION
Use this function to load a module from disk,
it must be the 31 samples 64 pattern type of
ProTracker module.

The module object must be free else the old
stuff will be gone.

Module
The module object to initialize.

FileName
An ordinary path to the module.

Result
If TRUE the module has been correctly loaded.
```

1.7 pauseptmodule

```
  SYNTAX
PausePTModule()

  STATEMENT
Call this statement to have the currently
played module paused.

The paused module could be resumed later
on even if one or more modules have been
played, and paused too, in between.
```

1.8 playptmodule

```
  SYNTAX
PlayPTModule(#Module.w)

  STATEMENT
Use this statement to start to play an module.

The currently played module will be stoped automaticly.

Module
```

```
The module to play.
```

1.9 resumeptmodule

```
  SYNTAX
ResumePTModule(#Module.w)

  STATEMENT
Use this statement to resume any paused module,
not just the currently played module that is
paused, at any time.

The currently played module will be stopped automatically.

Module
The module to resume.
```

1.10 setptmodulepos

```
  SYNTAX
SetPTModulePos(Pos.b)

  STATEMENT
Call this statement to set a new song position
for the currently played module.

Pos
The new song position.
```

1.11 setptmodulerow

```
  SYNTAX
SetPTModuleRow(Row.b)

  STATEMENT
Call this statement to set a new row number
for the currently played module.

Row
The new row number to set, must be in between 0 to 63.
```

1.12 setptmodulespeed

```
  SYNTAX
SetPTModuleSpeed(Speed.b)

  STATEMENT
```

```
This statement sets a new speed value for
the currently played module.

Speed
The new speed, could be somewhere in between 1 and 32.

As the value gets lower the module will go faster.
```

1.13 stopptmodule

```
  SYNTAX
StopPTModule()

  STATEMENT
Use this statement to stop the currently played module.
```