081280e0-4

Crossbones

COLLABORATORS			
	TITLE : 081280e0-4		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Crossbones	August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

081280e0-4

1.1 Protracker Library Documentation

```
                              CIA Protracker Library
                                  Blitz Basic II

                             Version 1.02b (14/11/95)

                                   Updated by:
                             Crossbones/Binary Assault
```

1.2 Getting Started/Installation

```
                                 Getting Started:


What you'll need:

 o Blitz Basic II version 1.8 or greater.
 o An Amiga*
 o Any music file in protracker format.


Machine Requirements:

 o Kickstart 2.0x+
 o At least 1 meg of chip ram.
 o A kickin' stereo.
```

```
Installing the library:

 First, remove any older version of this library you might have. Look for
names such as "NeilsTrackerLib.obj" or "CiaTrackerLib.obj." Don't delete them,
but move them out of the way for now, keeping them until you're sure you don't
need them, or that you've found the correct library to remove.

 Now copy the included library into your blitzlibs:userlibs directory. Once
copied, you'll need to recompile your DEFLIBS file to use it. (NOTE: Users of
version 2.1 need not do this, according to Acid.)

* - I sincerely hope you have one of these. :)
```

1.3 Introduction

```
                              Introduction



The standard soundtracker replay routines supplied with Blitz Basic 2 have
many faults, which this library attempts to overcome.  Some of the features
are:

  - Plays all PT songs that utilise either the VBLANK timing or the
    more recent CIA based timings
  - Plays back correctly on 50/60Hz systems, running either PAL or NTSC
  - Contains more specialised functions for advanced programmers
  - Enables the programmer to syncronise graphics with their music


Credits:

Original ProTracker playroutine by Amiga Freelancers, converted and enhanced
for Blitz by Neil O'Rourke.  Naggings from Roy, Jeff and Richard. Newer
revisions and further enhancements by Crossbones/Binary Assault.


NOTE:

 Neil stopped work on the library at version 1.6. Unfortunately, all the
source code I've located of this library is only at version 1.3. This means
that if you are fimiliar with the last version of his library, a few of the
commands might not contain the same sytax or might not even be in this library
any more.

 If you find that you simply can't live without a certain feature, you might
try to contact me and coax me into providing the update you need.  I say
*MIGHT* because there is no guarentee that I'll work on it.
```

1.4 Commands (Contents)

```
                        Commands

                   ~LoadTrackerModule~
                   ~~~StartTracker~~~~
                   ~~~~StopTracker~~~~
                   ~~~DecodeModule~~~~
                   ~~GetTrackerSize~~~
                   GetTrackerLocation~
                   ~FreeTrackerModule~
                   ~~GetTrackerEvent~~
                   ~CheckTrackerEvent~
                   ~WaitTrackerEvent~~
                   ~~CheckTrackerID~~~
                   ~~~ModuleToChip~~~~
                   ~~GetModuleName$~~~
                   ModulePositionJump~
                   ~GetModulePosition~
                   ~~~~PauseModule~~~~
                   ChangeTrackerVBLank
                   ~~~~PlayTracker~~~~
                   ~~~TrackerVolume~~~
                   ~ChangeTrackerMask~
                   ~~~ModulePatterns~~
```

1.5 Squish squish!

```
                            Bugs
                  (And How To Squash Them!)


Things I know about

 o Because of some unknown problem, the routine actually reserves 3 bytes of
   memory (added to the actual module size.) Keep this in mind when calling
   the function to find out the byte-size of a module.

Things I don’t know about

 Well!?! If I don’t know about them, I can’t really put them here can I? ;)
```

1.6 Contacting The Author

```
                  How to Contact the Author:
                      (This part’s easy!)

                   Crossbones/Binary Assault
                        aka Steve Flock

                         Via Internet
                    sflock@comtch.iea.com
```

```
                              Via Snail-Mail
                                Steve Flock
                            2421 west LaCrosse
                            Spokane, Washington
                                   99205
```

(Note: Please be sure that if you're contacting me via snailmail that you put the correct postage on the letter. If I have to pay to get the mail, chances are you won't get a reply.)

1.7 History

```
                   Revision History : CiaTrackerLibrary

Pre 1.01B:

 -Original version by Neil updated to use a decent method of CIA timing. No
  longer needs the crappy "SetDMA" command. :)

1.01B:

 -Fixed problem with slots. (Was supposed to be able to allow 0-8, but was
  only set only to keep track through 7.
 -Added various commands in an attempt to get up to date with the commands
  from Neil's last update. (Version 1.6)
 -Added better run-time error checking (Limited edition version of 1.01B.)

1.02 (Current):

 -Hoo boy! Did I ever screw up 1.01B. Good thing I didn't send it too much.
  Seems that the run-time errors section REALLY made a mess of things.
  Deleted them and rewrote the entire error checking system. MUCH better.
 -I am now using the TrackerTester program on all versions of the library
  ready for release. Since I have problems with beta testers offering to
  help, then never saying a damn thing once they get the library, I figure
  this is the next best thing. Checks all major commands for failure.
  (NOTE: This program is now included in releases. Use it if you have
  problems and report the errors/bugs to me for fixing.)
 -Fixed small problem with LoadTrackerModule command. (1.02a)
 -DecodeModule command changed to also include function from ModuleToChip.
  ModuleToChip still tokenises, but performs no function. (1.02a)
 -Repaired section that grabs the cia timer. This fixed the following
  problems:
  - Used to lose characters over the serial port when the player was active.
  - Failed to play if one of the two cia timers was being used. (It only
    was able to allocate one timer successfully, and failed on the other.)
  These two should fix any problems the player had on specific machines.
  (1.02a)
```

1.8 Thanks!! :)

The Thankyou Page:

This is the page where I get to thank all the people who have helped this library advance to the stage it’s currently at.

Ted Bailey
Andrew (Defender)
Richard Elmore

I know there are more, but I’ve forgotten your names. If you’re seeing this and saying "Geez, and he forgot me!" then get ahold of me and let me know.

1.9 LoadTrackerModule (Command)

LoadTrackerModule

Usage:

success=LoadTrackerModule(TrackerModule#,FileName$)

Comments:

Loads the named module into chip ram, ready for playing. This command can only be called in Amiga mode. success is a boolean return code (true). If the load fails for any reason, success returns the AmigaDOS error code.

Note that there is an implicit call to FreeTrackerModule for whatever module you are trying to load. However, if you want to load another module, don’t try to load it on top of the existing one that is playing. Use another TrackerModule# (you have from 0 to 8). The results are unpredictable, and range from nothing to a system crash. We can’t call StopTracker, because this will stop everything.

1.10 StartTracker (Command)

StartTracker

Usage:

success=StartTracker(TrackerModule#)

Comments:

Starts to play the requested module, stopping any modules already playing, or restarts the current module, and returns true. Returns false if the module couldn’t be started for some reason (like it isn’t loaded).

1.11 StopTracker (Command)

StopTracker

Usage:

```
StopTracker
```

Comments:

Stops the current module. Not much to it, really.

1.12 DecodeModule (Command)

DecodeModule

Usage:

```
DecodeModule TrackerModule#,ModuleAddress
```

Comments:

This sets an arbitary area of memory as a tracker module, useful if you have BLoaded/INCBIN'd a file and want to hear if it is a module. Caution: a non-module may crash the Amiga when you try to play it.

NOTE: As of release 1.02a, the ModuleToChip command has been changed. DecodeModule now does this for you automatically if the module that is being decoded is in fast ram. The ModuleToChip command will still tokenise though, to prevent problems that might occur to previous users of this library.

1.13 GetTrackerSize (Command)

GetTrackerSize

Usage:

```
trackerlength=GetTrackerSize(TrackerModule#)
```

Comments:

Not really much of a useful command. Simply returns the size (in bytes) that the module is using in memory. There should be no need to use this information, and these commands are only included because they served a purpose in debugging a long time ago, and to remove them would cause problems with the Blitz tokens.

1.14 GetTrackerLocation (Command)

GetTrackerLocation

Usage:

```
trackerlength=GetTrackerLocation(TrackerModule#)
```

Comments:

Not really much of a useful command. Simply returns the memory location that the module is occupying. There should be no need to use this information, and these commands are only included because they served a purpose in debugging a long time ago, and to remove them would cause problems with the Blitz tokens.

1.15 FreeTrackerModule (Command)

FreeTrackerModule

Usage:

```
FreeTrackerModule TrackerModule#
```

Comments:

This frees a module loaded with LoadTrackerModule. You cannot free a module that has been set up with DecodeModule (see below), but there is nothing to stop you trying.

1.16 GetTrackerEvent (Command)

GetTrackerEvent

Usage:

```
trackerevent=GetTrackerEvent
```

Comments:

This command is a customised extension to the ProTracker replay routine. A "TrackerEvent" occurs when the replay routine comes across a $8xx command. This command is not defined in the command list, and many demos (eg Jesus on E’s) use it to trigger effects. This command gets the most recent TrackerEvent, so any program looking at this will have to compare the current value to the value that triggered the current effect.

1.17 CheckTrackerEvent (Command)

CheckTrackerEvent

Usage:

```
success=CheckTrackerEvent
```

Comments:

This routine checks to see if a TrackerEvent has occured since the last time the routine was called, and returns True if it has. Use GetTrackerEvent to determine what data the $8xx command had.

1.18 WaitTrackerEvent (Command)

WaitTrackerEvent

Usage:

Unknown

Comments:

** V1.6: DO NOT USE THIS COMMAND! **

** V1.0b: I haven't checked this. **

1.19 CheckTrackerID (Command)

CheckTrackerID

Usage:

```
success=CheckTrackerID(TrackerModule#)
```

Comments:

This checks the module for the standard Pro/Noise/SoundTracker ID string "M.K." (or "M!K!" in the case of a 100 pattern PT module), and returns True if one of them is found. This means that you can safely call StartTracker.

Note that there is no 100% guarenteed way of determining what is a module and what isn't. Bit Arts, for example, remove the M.K. identifier to make it harder to rip modules, so if you're writing a module ripping program, you have to take this result with a grain of salt.

1.20 ModuleToChip (Command)

ModuleToChip

NOTE: This command is outdated as of this release. Even though the command
      will tokenise, the command has no function. Please see the

```
      DecodeModule command for more details.
```

1.21 GetModuleName$ (Command)

```
                              GetModuleName$

Usage:

name$=GetModuleName$(TrackerModule#)

Comments:

Returns the name of the module in name$. Not too useful, but I made a little
interface for workbench using the library, and needed a command like this.
```

1.22 ModulePositionJump (Command)

```
                            ModulePositionJump

Usage:

ModulePositionJump(Position#)

Comments:

 This command tells the play routine to jump to the pattern requested in
Position#.

NOTE: There is no error checking done at this time. It would be wise to know
      where you're going.
```

1.23 GetModulePosition (Command)

```
                             GetModulePosition

Usage:

position=GetModulePosition

Comments:

This returns the current pattern the replay routine is playing.
```

1.24 PauseModule (Command)

PauseModule

Usage:

PauseModule

Comments:

Stops the current module from playing, effectively pausing it. Use the command again to unpause it.

1.25 ChangeTrackerVBlank

ChangeTrackerVBlank

Usage:

ChangeTrackerVBlank

Comments:

This command seems pretty useless, but there sure are alot of module players that offer vblank timing. Call the command before playing the module, then call the StartTracker command, so it knows which module to use. Then simply call the PlayModule command on every vblank.

1.26 PlayModule (Command)

PlayModule

Usage:

PlayModule

Comments:

This command is to be used if you use the ChangeTrackerVBlank command.

To use this command, call ChangeTrackerVBlank first. Secondly, call the StartTracker command. At each vblank, you must then call PlayMoudule, which will keep the music playing.

1.27 TrackerVolume (Command)

TrackerVolume

Usage:

TrackerVolume [Volume Level]

Comments:

What can be said? This command changes the volume level of the module. Note: this effects all the channels currently masked in. (Refer to the command ChangeTrackerMask for more information on this).

1.28 ChangeTrackerMask

ChangeTrackerMask

Usage:

```
ChangeTrackerMask [NewMask]
```

Comments:

For all intents and purposes, you might never need this command. This command allows you to tell the replay routine that it is not supposed to use a certain channel, or certain channels. This is useful if you want, for instance, a two channel module playing and sounds effects at the same time. The command wants to know what channels the replay routine CAN use.

How to figure out what channels to mask:

This is fairly simple, and there are a couple of different ways to do this.

1. ChangeTrackerMask %0000ABCD - Where ABCD represents channels 0-3.

   Channels to be used by the replay routine would be represented as 1's, and channels you wish to use for sound effects, etc., would be represented as 0's.

   Example: ChangeTrackerMask %00001001 would play channels 0 and 3, while channels 1 and 2 would be free for you to use.

2. ChangeTrackerMask [DecimalValue] - Where decimal value represents:

   1 - Channel 0
   2 - Channel 1
   4 - Channel 2
   8 - Channel 3

   Example: ChangeTrackerMask 9 would play channels 0 and 3, while channels 1 and 2 would be free for you to use.

1.29 ModulePositions

ModulePatterns

Usage:

```
patt=ModulePatterns(Module#)

Comments:

 This command is kind of useless, but I wanted one so it’s included. ;) All
this command does is return the number of patterns used in the module.
```