How to decode a Vortex Tracker II "PT3" File

by Vince "Deater" Weaver, <vince@deater.net>
http://www.deater.net/weave/vmwprod/pt3_player/
10 September 2019

Background:

Vortex Tracker II (https://bulba.untergrund.net/vortex_e.htm)
is a music tracker (tool for writing music) that targets systems
with the AY-3-8910 sound chip. The music made is most popular
on various ZX Spectrum (z80) and Atari ST systems.

I wanted to make a decoder for a 6502-based Apple II with a
AY-3-8910 based Mockingboard sound card. The challenge was
all the low-level Vortex Tracker documentation is in Russian,
and the source code available is either uncommented z80 assembly
language or Russian-commented Pascal source code.

This document is based at least partly on the writeup here:
http://karoshi.auic.es/index.php?topic=397.msg4641#msg4641
as well as the AY_emul source code, and lastly just from reverse
engineering things while trying to get my code to match the output
of AY_emul exactly.

The PT3 Format

* File Header

Note: 16-bit values are little-endian

Offset	Size	Description	Contents
$00 - $0C	13 bytes	Magic	"ProTracker 3."
$0D	1 byte	Version	'5' for Vortex Tracker II
$0E - $1D	16 bytes	String	" compilation of "
$1E - $3E	32 bytes	Name	Name of the module
$3E - $41	4 bytes	String	" by "
$42 - $62	32 bytes	Author	Author of the module.
$63	1 byte	Frequency table (from 0 to 3)	
$64	1 byte	Speed/Delay	
$65	1 byte	Number of patterns+1 (Max Patterns)	
$66	1 byte	LPosPtr	Pattern Loop Pointer
$67 - $68	2 bytes	PatsPtrs	Pointers to the patterns
$69 - $A8	64 bytes	SamPtrs[32]	Pointers to the samples
$A9 - $C8	32 bytes	OrnPtrs[16]	Pointers to the ornaments
$C9 - ???		Patterns[]	$FF terminated, More on this below

Version: is at offset $0D, which is just the ASCII value of
the version number from the magic. Usually you
subtract off '0' to get the proper value.
If the value is not a valid number, '6' is assumed.

* Pattern list

The pattern list starts at $C9 and is terminated by $FF

The pattern is multiplied by 3.

A sample pattern list might look like
$03, $06, $09, $FF
which corresponds to playing in order patterns
1, 2, 3

```
* Samples

        The pt3 file allows for 32 samples.  These samples contain
        values that are applied to the music notes as they are playing.

        The 16-bit address of a sample X can be found by getting the
                16-bit little-endian address at offsets
                $6A+(X*2) and $6B+(X*2) in the header.

        Byte 0:  LOOP VALUE -- sample offset to return to once hit the end
        Byte 1:  LENGTH     -- number of 32-bit sample values
        Byte 2+: VALUES     -- list of 4-byte (32-bit samples)

        * Sample format

                + Byte 0 --- 7 6 5 4321 0
                               Bit 7 - Amplitude sliding
                               Bit 6 - 1/0 amplitude slide up/down
                               Bit 5 - 1/0 envelope slide up/down
                               Bits 4321 - sign extend, envelope slide value
                               Bit 0 - Sample has envelope

                + Byte 1 --- 7 6 5 4 3210
                               Bit 7     - Envelope sliding
                               Bit 6     - Accumulate Tone
                               Bit 5     - parameter to envelope sliding?
                               Bit 4     - Bit 6+4 used to set Noise/Channel
                                           mixer value
                               Bits 3210 - Amplitude

                + Byte 2 --- Freq Low -\
                + Byte 3 --- Freq High----- Used as base tone

* Ornaments

        The PT3 file format has 16 ornaments, which are patterns applied
        to the note.  This can be used for ?? effects.

        The 16-bit address of an ornament X can be found by getting the
                16-bit little-endian address at offsets
                $A9+(X*2) and $AA+(X*2) in the header.

        Byte 0:  LOOP VALUE -- ornament offset to return to once hit the end
        Byte 1:  LENGTH     -- number of ornament values
        Byte 2+: VALUES     -- list of single-byte values applied to the notes


* Pattern data

        For each of three channels (A,B,C) there is a nul-terminated
        stream of bytes that gives the pattern data.

        To find the pattern data for a pattern, look it up in

        A 6-byte chunk with the 16-bit addresses for A,B,C for pattern
        X can be found by
                a_addr_l = [address in ($68/$67)] + (X*6)+0
                a_addr_h = [address in ($68/$67)] + (X*6)+1
                b_addr_l = [address in ($68/$67)] + (X*6)+2
                b_addr_h = [address in ($68/$67)] + (X*6)+3
                c_addr_l = [address in ($68/$67)] + (X*6)+4
                c_addr_h = [address in ($68/$67)] + (X*6)+5

        Follow that address to find the nul-terminated pattern data.

        The data can be interpreted this way:

        + $00         - NUL termination, end of pattern
        + $01-$0f     - effects, see later.
```

```
                Note that parameters to the effect appear in the
                bytestream *after* the note to play
+ $10-$1f     - set envelope type
                + $10 means disable envelope
                        - sample number is next byte
                + $11-$1F
                        - envelope type is bottom 4 bits
                        - envelope period is next 2 bytes (big endian)
                        - envelope delay is next 1 byte
                        - sample number is next byte

+ $20-$3f     - set noise
                        - Noise set to the ($20...$3F) value - $20
+ $40-$4f     - set ornament
                        - ornament sent to bottom 4 bits
                        - possibly setting 0 counts as disabling
+ $50-$af     - play note

       0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f
  50: C-1 C#1 D-1 D#1 E-1 F-1 F#1 G-1 G#1 A-1 A#1 B-1 C-2 C#2 D-2 D#2
  60: E-2 F-2 F#2 G-2 G#2 A-2 A#2 B-2 C-3 C#3 D-3 D#3 E-3 F-3 F#3 G-3
  70: G#3 A-3 A#3 B-3 C-4 C#4 D-4 D#4 E-4 F-2 F#4 G-4 G#4 A-4 A#4 B-4
  80: C-5 C#5 D-5 D#5 E-5 F-5 F#5 G-5 G#5 A-5 A#5 B-5 C-6 C#6 D-6 D#6
  90: E-6 F-6 F#6 G-6 G#6 A-6 A#6 B-6 C-7 C#7 D-7 D#7 E-7 F-7 F#7 G-7
  a0: G#7 A-7 A#7 B-7 C-8 C#8 D-8 D#8 E-8 F-8 F#8 G-8 G#8 A-8 A#8 B-8

+ $b0         - disable envelope, reset ornament position
+ $b1         - set skip value
                        - next byte is how many lines to skip for
                          this note
+ $b2-$bf     - set envelope
                        - envelope type to (bottom 4 bits - 1)
                        - envelope period next 2 bytes (big endian)


+ $c0         - turn off note (volume=0 and disable everything)
+ $c1-$cf     - set volume
                        - volume set to bottom 4 bits

+ $d0         - done processing this note (end note)
+ $d1-$ef     - set sample
                        - sample set to value-$d0.
+ f0-$ff      - initialize ornament/sample.  Envelope is disabled
                        - ornament is bottom 4 bits
                        - next byte is sample*2

Note when parsing if you reach a note, a $D0 or a $C0 then you
are done parsing the note for this line and should move on
to parsing the effects.

Effects
~~~~~~~

AY_emul supports having up to one of each effect for each line, and
tracks the order in which to apply them.  However in the wild
I have not seen more than one effect applied per line/channel
(I don't think Vortex Tracker lets you add multiple?)

The bytes for the effects follow after the byte that ended the note.

The effect value on disk is not the same as that displayed when
viewed in the tracker.

On      In
Disk  Tracker
~~~~  ~~~~~~~

$01     $01:    Glissando (slide through discrete notes) / Tone Down
```

```
                        First byte: delay
                        Next 2 bytes: frequency to add (is negative)

        $01     $02:    Glissando (slide) / Tone Up

                        This appears differently in tracker, but is just
                        the same as above but with a positive frequency.

        $02     $03:    Tone portamento (continuous slide)

                        First byte: delay
                        Next 2 bytes: ignored?
                        Next 2 bytes: slide step (little-endian)

        $03     $04:    Sample Offset
                        First byte: value to set the sample position offset

        $04     $05:    Ornament offset
                        First byte: value to set the ornament offset

        $05     $06:    Vibrato
                        Periodic sound off/on in that channel
                        First byte: OffOn delay (frames to stay off)
                        Second byte: OnOff delay (frames to stay on)

        $08     $09:    Envelope Glissando -- Frequency decreasing
                        First byte: delay
                        Next two bytes: Slide add (little-endian)

        $08     $0A:    Envelope Glissando -- Frequency increasing.
                        Like previous but with sign switched?

        $09     $0B:    Set playing speed (new Delay).
                        First byte: new delay (global number of frames per line)


  * Frequency Tables

        Various versions of the tracker use different frequency tables.
        Players lookup notes in these tables to get the proper frequency.

        The value in the frequency table field specifies up to 4 (0...3)
        tables, but there is an alternate set of tables if the tracker version
        PT3_VERSION is 3 or less.

        Presumably for space reasons, some players only support tables
        one and two from the newer set.

        These tables in theory can be calculated at runtime, but usually
        they aren't unless you are extremely space constrained.
        The z80 player has code to do this in z80 assembly.

        The tables are a set of 96 16-bit values (8 octaves of 12 notes)

        The frequency tables supported by Ay_Emul:

                PT3Version#     Table#          Frequency Table
                -----------     ------          ======================
                <=3.3           0               PT3NoteTable_PT_33_34r
                all             1               PT3NoteTable_ST
                <=3.3           2               PT3NoteTable_ASM_34r
                <=3.3           3               PT3NoteTable_REAL_34r

                3.4+            0               PT3NoteTable_PT_34_35
                all             1               PT3NoteTable_ST
                3.4+            2               PT3NoteTable_ASM_34_35
                3.4+            3               PT3NoteTable_REAL_34_35

  * Volume Tables
```

There are multiple 256-byte amplitude/volume lookup tables
These seem to have changed with different versions of Vortex Tracker
so to be complete you need to contain them all and use the proper
one at run time

* Corner cases seen in the wild:
+ What to do if the delay on an effect is set to zero?

* 6-channel PT3 Files

In the Protracker 3.7 format you can look for a turbosound header
at the very end of the file.

Type1 (4 bytes), Size (16-bits, little-endian)
Type2 (4 bytes), Size (16-bits, little endian)
TSID: (4 bytes) 02TS

If the type of these files is PT3! then these are describing
PT3 subfiles, and can be read as two 3-channel files which can
be interpreted together as one 6-channel file.