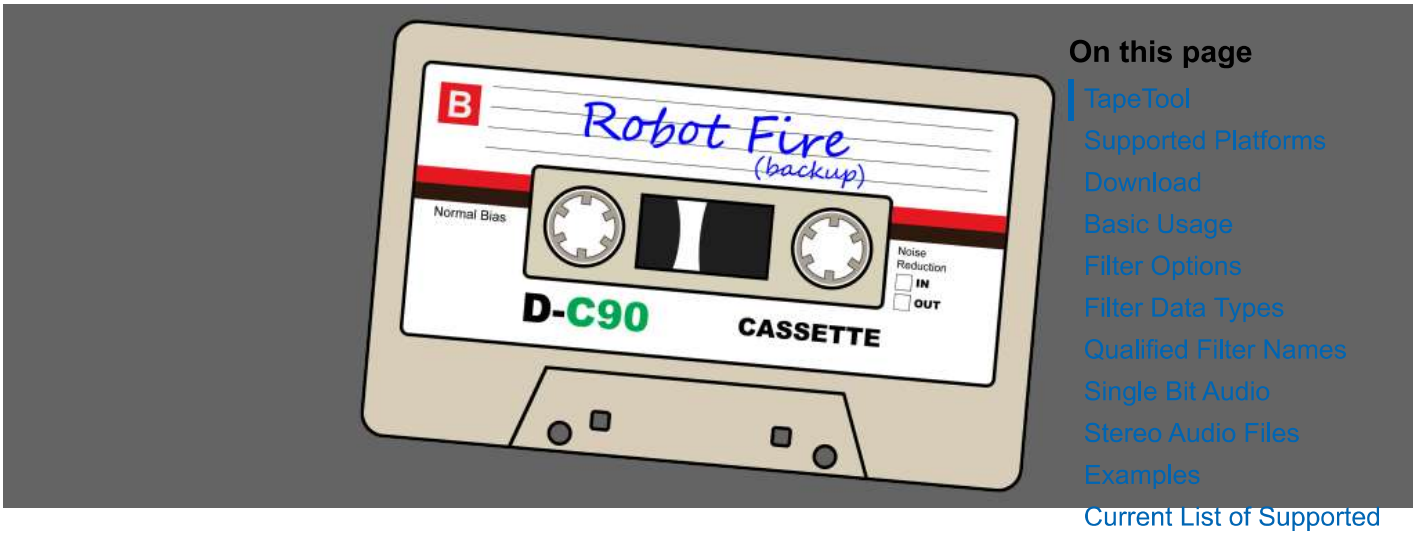

On this page

TapeTool

TapeTool (aka tapetool2) is a command line utility for converting and repairing Microbee, TRS-80 and Sorcerer audio tape recordings.

TapeTool2 might also be useful for other retro tape recordings - especially those that are similar to the Kansas City standard but hasn't been tested in these cases.

This is the second version of this program. Version 2 is a complete re-write and the command line arguments aren't compatibile with version 1. The original version can still be found here.

Version 2 improves over version 1 in the following ways:

- Simpler, easier to understand "filter" model
- Support for single bit audio files
- Various audio filters (eg: Low-pass, high-pass and band-pass filters)

Supported Platforms

TapeTool2 is written is C# and runs on Windows natively (assuming you have .NET installed) and will work on OS X and Linux if Mono is installed.

When using Mono, you need to invoke tapetool2 via mono. eg:

```
> mono tapetool2.exe --help
```

(or, you could write a simple shell script to invoke it)

Download

- tapetool2.exe

Basic Usage

Tape tool processes audio, binary and other data through a series of filters - starting with a file reader and typically ending with a file writer. Filenames and filters are specified on the command line and are processed in a left to right manner.

For certain known file extensions, the appropriate reader or writer filter is automatically created. For the first file on the command line a file reader is used, for all subsequent file names a file writer is used.

In between file reader and writers you can place filters which perform various manipulations on the data passed to them.

Take the following command line as an example:

```
> tapetool2 robotf.wav smooth robotf_smoothed.wav
```

In this example a wave file reader will be used to read the file `robotf.wav`. The wave data will then be fed to a `smooth` filter which smoothes the audio signal before passing it to the wave writer filter that writes the smoothed audio data to the file `robotf_smoothed.wav`.

To get a full list of available filters, use `--help` (or see bottom of this page).

```
> tapetool2 --help
```

Filter Options

Most filters have options which can be set using command line switches after the filter. For example the smooth filter has a `period` setting which determines the smoothing period. eg:

```
> tapetool2 robotf.wav smooth --period:10 robotf_smoothed.wav
```

To determine the available options for a filter, use `--help` after the filter name. eg:

```
> tapetool2 smooth --help
```

will show:

On this page

```
Input Kind: audio stream
Output Kind: audio stream

Options:
  --period:val                    Smoothing period in samples (default=8)
```

Filter Data Types

For two filters to be connected the output of one filter needs to match the input of the next. The following example reads and audio file, parses it to a binary data stream using the `microbee.audioToBytes` filter and then writes it to the file output.bin

```
> tapetool2 robotf.wav microbee.audioToBytes output.bin
```

In some cases, tapetool can automatically construct an appropriate chain of filters to get the job done. eg:

```
> tapetool2 robotf.wav robotf.tap
```

will automatically use a sequence of filters consisting of:

```
robotf.wav
    -> waveReader
      -> microbee.audioToBytes
        -> microbee.bytesToBlocks
          -> microbee.blocksToBytes
            -> microbee.tapFileWriter
              -> robotf.tap
```

Qualified Filter Names

Some filters are qualified with a `microbee.`, `trs80.`, `sorcerer` or `kanasa` prefix indicating the filter is specific to that platform. To save having to type the prefix on every filter you can use the platform name switch at the start of the command. eg:

```
> tapetool2 --microbee robotf.wav audioToBytes bytesToBlocks blocksToBytes
robotf.tap
```

Tapetool2 supports a special audio file format called Single Bit Audio (aka ".sba" files). This format is a compact audio representation for tape audio data and is the tape file format used by FPGABee.

The file format consists of a 16-byte header in little endian format:

```
#define SBA_SIGNATURE  0x53425054       // TPBS aka "TaPe Bit Stream"

struct SINGLEBITAUDIOHEADER
{
    uint32_t signature;
    uint32_t sampleRate;
    uint32_t totalSamples;
    uint32_t reserved;
}
```

...followed by the audio data with 8 single bit samples packed into each byte. The samples are ordered from least significant bit to most significant bit. ie: bit 0 plays first.

Single bit audio files are single channel (ie: mono) only and typically have a sample rate of 22050Hz which is enough for 1200baud data.

Stereo Audio Files

Most filters in tapetool2 only work with mono signals and if fed a multi-channel signal will use the first channel (ie: left channel) and will ignore the other channels.

There are however filters that can be used to pick out a specific channel, or to multi-cast one channel on two many.

See the filters: `channelMultiCast`, `selectChannel`.

Examples

These examples have been split over multiple lines to clarify each filter. Normally you'd type all this on one line.

Converting a .wav file to a .tap file:

```
> tapetool2
    robotf.wav
    robotf.tap
```

On this page

```
> tapetool2
    robotf.wav
    bandpass
    robotf_filters.wav
```

Converting a .wav file to a .tap file with a 2.8Khz lowpass pass filter of on the input audio

```
> tapetool2
    robotf.wav
    lowpass --freq:2800
    robotf.tap
```

Converting a .wav file to a .sba file

```
> tapetool2
    robotf.wav
    robotf.sba
```

Converting a .tap file to a text representation

```
> tapetool2
    robotf.tap
    robotf.blocks.txt
```

Packing unchunked binary data and converting to a wav file.

```
> tapetool2 --microbee
    mygame.bin
    packData
    setHeader --autoStart:255
    blocksToBytes
    mygame.wav
```

Packing unchunked binary data and converting to a wav file with load address options:

```
> tapetool2 --microbee
    mygame.bin
```

On this page

```
filename:"MYGAME"
    blocksToBytes
    mygame.wav
```

Packing unchunked binary data from a file with non-".bin" file extension:

```
> tapetool2 --microbee
    binReader --filename:mygame.rom
    packData
    setHeader --autoStart:255
    blocksToBytes
    mygame.wav
```


Current List of Supported Filters

Here's the current output from `tapetool2 --help`.

```
tapetool2 v1.2.1200 - Microbee/TRS-80/Sorcerer Tape Diagnotic Utility
Copyright (C) 2017-2018 Topten Software.

Usage: tapetool2 [filters...]

Supported Filters:
  analyse                       Analyses an audio stream for tape related
characteristics
  audioToCycleLengths           Generates audio cycle lengths from an audio
stream
  bandPass                      Applies a band-pass filter to an audio stream
  binReader                     Binary file reader (*.bin reader)
  binWriter                     Binary file writer (*.bin writer)
  changeRate                    Changes the sample rate of an audio stream
without resampling it
  channelMultiCast              Multicasts a single audio channel to multiple
identical channels
  cycleLengthsToCycleKinds      Generates cycle-kinds from cycle-lengths
  dcOffset                      Adjusts the DC offset of an audio stream
  gain                          Adjusts the volume level of an audio stream
  highPass                      Applies a high-pass filter to an audio stream
  kansas.audioToBytes           Parses an Kansas City tape audio stream into a
```

```
audio stream
  kansas.audioToHalfCycleKinds   Generates Kansas City audio half-cycles from an
audio stream
  kansas.bitsToBytes             Decodes a Kansas City bit stream into byte
stream
  kansas.bitsToCycleKinds        Generates Kansas City cycle kinds from a bit
stream
  kansas.bitsToHalfCycleKinds    Generates Kansas City half-cycle kinds from a
bit stream
  kansas.bytesToBits             Encodes a byte stream into Kansas City bit
stream
  kansas.cycleKindsToAudio       Generates Kansas City audio cycles from a cycle
kind stream
  kansas.cycleKindsToBits        Parses Kansas City cycle kinds into a bit
stream
  kansas.halfCycleKindsToAudio   Generates Kansas City audio from a half-cycle
kind stream
  lowPass                        Applies a low-pass filter to an audio stream
  microbee.audioToBytes          Parses an Microbee tape audio stream into a
bytes
  microbee.audioToCycleKinds     Generates Microbee audio cycles from an audio
stream
  microbee.audioToHalfCycleKinds Generates Microbee audio half-cycles from an
audio stream
  microbee.bitsToBytes           Decodes a Microbee bit stream into byte stream
  microbee.bitsToCycleKinds      Generates Microbee cycle kinds from a bit
stream
  microbee.blocksToBytes         Encodes a Microbee block stream into bytes
  microbee.bytesToBits           Encodes a byte stream into Microbee bit stream
  microbee.bytesToBlocks         Decodes a Microbee byte stream into blocks
  microbee.bytesToTap            Encodes a byte stream into a Microbee tap
stream
  microbee.cycleKindsToAudio     Generates Microbee audio cycles from a cycle
kind stream
  microbee.cycleKindsToBits      Parses Microbee cycle kinds into a bit stream
  microbee.halfCycleKindsToAudio Generates Microbee audio from a half-`cycle
kind stream
  microbee.packData              Packs binary data into Microbee block format
  microbee.parseAudio            Parses a Microbee audio stream into block
stream
  microbee.renderAudio           Parses a Microbee block stream into an audio
```

On this page

```
  microbee.tapFileReader         Microbee tape file reader (*.tap reader)
  microbee.tapFileWriter         Mirobee tape file writer (*.tap writer)
  microbee.tapToBytes            Decodes a Microbee tap byte stream
  microbee.textBlockStreamReader Text block-stream file reader (*.blocks.txt
reader)
  microbee.textBlockStreamWriter Text block-stream file writer (*.blocks.txt
writer)
  microbee.unpackData            Unpacks binary data from Microbee block format
  mono                           Mixes a multi-channel audio stream to mono
  resample                       Resamples an audio stream to a new sample rate
  selectChannel                  Selects one channel from a multi-channel audio
stream
  singleBitAudioReader           Single bit audio file reader (*.sba reader)
  singleBitAudioWriter           Single bit audio file writer (*.sba writer)
  smooth                         Smooths audio using a moving average
  sorcerer.audioToBytes          Parses an Exidy Sorcerer tape audio stream into
a bytes
  sorcerer.audioToHalfCycleKinds Generates Exidy Sorcerer audio half-cycles from
an audio stream
  sorcerer.bitsToBytes           Decodes a Exidy Sorcerer bit stream into byte
stream
  sorcerer.bitsToHalfCycleKinds  Generates Kansas City half-cycle kinds from a
bit stream
  sorcerer.blocksToBytes         Encodes a Exidy Sorcerer block stream into
bytes
  sorcerer.bytesToBits           Encodes a byte stream into Exidy Sorcerer bit
stream
  sorcerer.bytesToBlocks         Decodes a Exidy Sorcerer byte stream into
blocks
  sorcerer.bytesToTap            Encodes a byte stream into a Exidy Sorcerer tap
stream
  sorcerer.halfCycleKindsToAudio Generates Exidy Sorcerer audio from a half-
cycle kind stream
  sorcerer.packData              Packs binary data into Exidy Sorcerer block
format
  sorcerer.parseAudio            Parses a Exidy Sorcerer audio stream into block
stream
  sorcerer.renderAudio           Parses a Exidy Sorcerer block stream into an
audio stream
  sorcerer.setHeader             Updates the header in a Exidy Sorcerer block
stream
```

On this page

```
  sorcerer.textBlockStreamWriter Text block-stream file writer (*.blocks.txt
writer)
  sorcerer.unpackData            Unpacks binary data from Exidy Sorcerer block
format
  textBitStreamReader            Text bit-stream file reader (*.bits.txt reader)
  textBitStreamWriter            Text bit-stream file writer (*.bits.txt writer)
  textByteStreamReader           Text byte-stream file reader (*.bytes.txt
reader)
  textByteStreamWriter           Text byte-stream file writer (*.bytes.txt
writer)
  textCycleKindReader            Text cycle-kind file reader (*.cycles.txt
reader)
  textCycleKindWriter            Text cycle-kind file writer (*.cycles.txt
writer)
  textCycleLengthReader          Text cycle-length file reader (*.cyclelen.txt
reader)
  textCycleLengthWriter          Text cycle-length file writer (*.cyclelen.txt
writer)
  textHalfCycleKindReader        Text half-cycle-kind file reader
(*.halfcycles.txt reader)
  textHalfCycleKindWriter        Text half-cycle-kind file writer
(*.halfcycles.txt writer)
  trs80.bitsToBytes              Decodes a TRS-80 bit stream into byte stream
  trs80.bitsToCycleKinds         Generates TRS-80 cycle kinds from a bit stream
  trs80.blocksToBytes            Encodes a TRS-80 block stream into bytes
  trs80.bytesToBits              Encodes a byte stream into TRS-80 bit stream
  trs80.bytesToBlocks            Decodes a TRS-80 byte stream into blocks
  trs80.cycleKindsToAudio        Generates TRS-80 audio from a cycle kind stream
  trs80.cycleKindsToBits         Converts TRS80 cycle kinds into a bit stream
  trs80.parseAudio               Parses a TRS-80 audio stream into block stream
  trs80.renderAudio              Parses a TRS-80 block stream into an audio
stream
  trs80.textBlockStreamReader    Text block-stream file reader (*.blocks.txt
reader)
  trs80.textBlockStreamWriter    Text block-stream file writer (*.blocks.txt
writer)
  waveReader                     Wave file reader (*.wav reader)
  waveWriter                     Wave file writer (*.wav writer)

Options:
  -h | --help                    Show these usage instructions, or use after
```

On this page

```
--kansas                    Use filters 'kansas.*'
--microbee                  Use filters 'microbee.*'
--sorcerer                  Use filters 'sorcerer.*'
--trs80                     Use filters 'trs80.*'
```

On this page