TPCM - Test PCM

Syntax

```
tpcm sound.wav
```

playback the file "sound.wav" on the **PAS**

```
tpcm -v sound.wav
```

playback and **display** the file "sound.wav" on the **PAS**

```
tpcm -Tv sound.wav
```

playback and **display** the file "sound.wav" on the **SB**

```
tpcm [-T] [-dqmL] [-v] sound.wav ...
```

playback mode on **PAS/SB** using .wav files, optionally **display**

```
tpcm [-T] [-dqmL] [-v] [-r#l#s] [-iname]
```

playback mode on **PAS/SB** using raw sound data, optional **display**

```
tpcm -R[X][V#] [-dqmL] [-v] [-r#l#s] sound.wav
```

record mode on **PAS** into .wav file, optional **compression**, **voice-activated** and **display**, using specified **samplerate**, **samplesize**, **stereo/mono** parameters

Options

-d **debug** mode, output status

display DMA buffer information, playback or record parameters, dma buffer division processing numbers; during voice-activated record mode, displays the time at which recording was activated, and the current buffer offset, each time recording is voice-activated.

-T **playback** on **Thunderboard** or Sound Blaster

requires "`set BLASTER=...`" in environment, or previously loaded "**pcm.com**" using "`-tPort,Dma,Irq`" option. The Thunderboard, or Sound Blaster, can not be used as a recording device using this program.

-R[X] **record** mode, **X= compression** method; default: none

The program will **record** into a **compressed** format in **real-time**, using either the (I)ma, (A)law, or (M)ulaw formats.

-V# **voice activated** threshold value, when **recording** (`-R`)

Each DMA Buffer Division is processed, and if the calculation results in the threshold value being exceeded, then that data is saved to disk.

-t# #= **time** in seconds to **record**

During recording, this option specifies the amount of time before stopping; although fractional seconds are allowed, the precision is not great.

-L **Loop** sound continuously until ESCAPE

-v **visually** display waveform in MCGA mode (320x200x256)

`-r#`	#= **samplerate** (decimal) default: 11025
`-l#`	#= **samplesize** in **bits** (decimal) default: 8
`-s`	treat as **stereo** (default: mono)

These parameters are used for recording, or when the input file is of an unrecognized format. If the file specified is of a **recognized** format, then these values are ignored.

`sound.wav ...` = name(s) of **sound files** to playback

If the file specified is not recognized, then the default samplerate, datasize and channels are used, unless new values are specified using the options **`-r, -l,`** `and` **`-s`**.

`-iname` name= name of **raw** PCM file (no or unknown header)

Use the `-r`, `-l` and `-s` options to specify the samplerate, datasize and mono/stereo parameters. Note: "`-iname`" and "`sound.wav ...`" can not both be present on the command line.

`-w#` #= **delay** in clock ticks before timing-out; default: 4

This option prevents returning an errorlevel for slow disk drives.

`-q`	**quiet** mode (no sound or DMA transfers), allocate memory and run through file
`-m`	toggle use **Monochrome** screen memory for DMA

These were provided early in development as very low-level debug options; their usefulness has since diminished.

`-p#`	#= **palette** # (1-8) (default: 2 (6 if -v2 specified))
`-c#`	#= **cycle** palette with delay (default delay: 1)

These options pertain to modifying the background and the palette for the visual, `-v`, option.

Advanced Options

`-B#` #= DMA **Buffer size** (KB if <= 64, else Bytes)

Use this option to specify the size of the DMA Buffer. The default size is 32K, but 16K if using the `-m` option.

To specify a 64KB buffer, use `-B64`
To specify a 64000-byte buffer, use `-B64000`

The DMA Buffer Size and Divisions can be selected so as to provide a particular frequency of hardware interrupts, such as "15 interrupts per second", enabling syncronization capabilities with other multimedia events.

`-D#` #= number of DMA **Divisions**

The DMA Buffer is split into divisions, which enables sound output to occur while loading data from the disk. The default is 16 if visually displaying with the `-v` option, else 8, and should not be less than 2.

Also, the number of divisions is important for the **Voice-Activated Recording**, as the calculation takes place for each division.

`-b###` ###= **Base Port** Address for PAS

This option allows the program to address **multiple** PAS boards, because the **pcm.com** driver can install multiple times for unique board addresses.

The default is for **tpcm.exe** to use the **last-loaded** pcm.com, for whichever board it addresses. When this option is specified, pcm.com is queried with the **board address,** loading a new pcm.com if the address is not serviced or proceeding with the resident copy of the driver.

If a PAS is located at one of the four **standard addresses** of 388h, 384h, 38Ch, or 288h, then the pcm.com driver will locate it automatically, in that order. Locating the board at a non-standard address **requires** that the Board Address be specified.

`-S####` ####= **Segment** Address of **DMA Buffer**
`-O####` ####= **Offset** Address of **DMA Buffer**

For advanced users: if specified, the program prompts to confirm this option. You can allocate memory outside of the program, in the Upper Memory Blocks, using special utilities, and then specify the location of this memory to tpcm.exe, thereby removing the need to allocate such from within the program.

You can also specify the location of video memory, which is normally safe, and useful for debugging DMA transfers, as in below, where the location of the DMA Buffer is specified to be at B000:8000 and the size to be 32K:

```
C:> tpcm -SB000 -O8000 -B32 sound.wav
```

If you were using this program from within another program, and had already allocated memory for DMA, then you would launch this program with that address.

`-Fseg:off` seg:off= Address of **Function** to process Hardware Interrupt

For advanced users: this option is useful when the program is spawned from within another program, which can supply a routine for the Hardware Interrupt, its address passed to **tpcm.exe** in this manner.

The trickiness that can be associated with this option, that is, a **completely external routine** getting called by a **hardware interrupt**, should not deter you from exploring its potentials; the source code to tpcm.exe includes the routines that are called during the HW IRQ (the waveform display routines), and so can serve as a guideline to developing your own.

Examples of Playback

```
C:> tpcm sound.wav
```

This simply **plays** the file "**sound.wav**".

```
C:> tpcm -T sound.wav
```

This **plays** the file "**sound.wav**" on the **ThunderBoard** (SoundBlaster) DAC.

The "pcm.com" driver looks for "BLASTER=" in the environment to configure for the SB.

```
C:> tpcm -v sound.wav
```

This **displays** and **plays** the file "**sound.wav**".

```
C:> tpcm -v *.wav
```

This displays and plays **all .wav** files.

```
C:> tpcm -vp7c600 *.wav
```

This displays and plays all .wav files using **palette 7** with a **delay** of "**600**" (computer speed dependent).

```
C:> tpcm -isound.snd -r22050 -s
```

This plays the **raw** PCM file "sound.snd" at **22050Hz** in **stereo**.

```
C:> tpcm -vm *.wav
```

This displays and plays all sound files, but uses the **monochrome screen memory** for the DMA buffer.

Examples of Recording

```
C:> tpcm -R x.wav
```

This **records** into the file "**x.wav**" at 11025Hz, 8-bit mono.

```
C:> tpcm -v -R -r44100 -s -l16 cdqualty.wav
```

This records into the file "cdqualty.wav" at **44100Hz**, **16-bit stereo**, while **displaying** the waveform.

```
C:> tpcm -RI -r44100sl16 cdqualty.cmp
```

This records into the file "cdqualty.cmp" at 44100Hz, 16-bit stereo, and **compresses** using the **IMA** method.

Playback Control

During playback of the waveform, several keys are processed:

`w`	**wait** for keypress, loops on DMA buffer

Not particularly glamorous, but can be quite annoying.

`< >`	**slow/speed** samplerate

Use this key to slowdown/speedup the playback or record rate using "factors".

`, .`	**slow/speed** samplerate

Use these keys to slowdown/speedup the samplerate in increments of 100Hz.

`R`	Enter new **Samplerate**

If "debug" mode is on, then a prompt is displayed. Then the program allows you to enter a new samplerate.

`m`	display "**mark**", ticks/bytes/samples

Use this key to display the current offset into the playback or record buffer.

`l`	toggle "**looping**"

Use this key to activate/deactivate looping back to start after end of sound file.

`x`	toggle "**debug**"
`p`	**Pause** Playback/Record

Press another key to resume playback/record.

`i`	**Display** current samplerate, channels, datasize
`!`	Get command **shell**
`ESCAPE`	**exit** current sound file

Display Control

During display of the waveform, several keys are processed:

`P`	**palette** select - the program waits for either 0= no palette, 1 through 8= palettes 1 through 8
`c C`	auto **cycle delay** (+1/10), increasing delay as you hold key
`r`	**reverse** cycle, c,C now decreases delay as you hold key
`SPACE`	stop cycling
`f F`	manually cycle one way (256/16 colors)
`b B`	manually cycle the other (256/16 colors)
`u d`	change RGB up/down 1
`U D`	change RGB up/down 10
`RETURN`	restore palette
`v`	Cycle Video Backgrounds

Operation

Use this program to **play** and **record .wav** files. It can **display** them during playback or record, and even with **compressed 16-bit** .wav files (see "**cmpwav.exe**").

Use this program to manually specify many parameters, such as the **location** and **size** of the **DMA buffer**, **samplerate** and **datasize**, **IRQ activity**, and others, during the playback or record.

This program does not require that the TSR **pcm.com** to be loaded in memory. Instead, the program will load "pcm.com" it if it is not already so found. This requires only that "pcm.com" be found in the "`PATH`" (tpcm.exe and pcm.com should be in the same directory: `\bin` or `\util`).

Options

As you can see, there are many options available to affect the startup conditions of the program.

`-T`

Use the `-T` option to output the sound on the **SB** part of the **PAS**, or on a **ThunderBoard**, or on a **Sound Blaster**.

You must set the variable "`BLASTER`" in the environment before pcm.com loads and before running **tpcm.exe**. The following defines the meaning of this variable:

```
C:> set BLASTER=A220 D1 I5 T3
        |       |    || || ||
        |     |   || ||  `---- Product Number "3"
        |       |    ||  `------ IRQ Number "5"
        |       |    `---------- DMA Channel "1"
        |       `------------- Port Address "220h"
        `-------------------- Variable Name "BLASTER"
```

`-d`

Use the `-d` option to observe the queueing of data and the waiting for DMA to finish.

Use this option when attempting to locate valid DMA/IRQ combinations, or otherwise attempt to debug your sound card's operation.

When using the voice-activated record mode, the output will be merely the current time and buffer offset when recording begins. Redirect this output to a file for later reference to sound clips of interest.

`-q`

Use the `-q` option to suppress the output of sound and have the program simply run through the file. Note that there will be no display (if the `-v` option was specified) as there is no DMA activity and so no User Function calls. Use the `-d` option to see the progress of loading the file and the associated simulated DMA transfers.

`-m`

Use the `-m` option to cause the program to use monochrome screen memory for the DMA buffer. Note: this memory may be used as DOS UMB (loadhigh), and overwriting this area with sound data is not suggested! However, if you do have a monochrome monitor, then you can watch the data slam by.

Why use the monochrome screen buffer for DMA? Well, nowadays, with VGA and multi-megabyte-memory machines, the B segment is a nice 32K block of memory commonly unused, and perfect for scratch memory (like sound, where overwrites are merely noisy, not fatal).

Unfortunately, this memory is too slow for use with 44,100 Hz sounds, so this is not a perfect solution. Also, some machines generate a high-frequency pitch when accessing this memory
during 16-bit DMA transfers.

`-L`

Use the `-L` option to loop on the sound file continuously. Press `ESCAPE` to exit the loop and proceed to the next file (if any).

Use this option with the '`m`' key to display the current timeticks/bytes/ samples of playback or record. This gives you the ability to jot notes on paper of where interesting clips occur in a sound file.

While looping, you can press the "`< >`" keys (or "`, .`") to slowdown or speedup the samplerate, and so achieve a higher degree of resolution when specifying "marks".

`-i`

Use the `-i` option to specify a filename (no DOS wildcards allowed). Use the `-s`, `-l` and `-r` options to specify stereo, the sample size in bits, and the sample rate.

Use these options to process "raw" data.

This option will also process "raw compressed" formats, using the following syntax:

```
-ifile.ext,F,OFFSET,LENGTH
```

`F`= format: `p`= PCM, `i`= IMA, `a`= ALAW, `m`= MULAW
`OFFSET`= point within file to begin decompression
`LENGTH`= amount of data in bytes to process

For instance, suppose you get some "sound file" of an unknown type. You could use this program to attempt to listen to the file:

```
C:> tpcm -isound.dmp -l8
```
... try 8-bit PCM ...

```
C:> tpcm -isound.dmp -l16
```
... try 16-bit PCM ...

```
C:> tpcm -isound.dmp,i
```
... try 4-bit IMA ...

```
C:> tpcm -isound.dmp,a
```
... try 8-bit ALAW ...

```
C:> tpcm -isound.dmp,m
```
... try 8-bit MULAW ...

Additionally, using a "dump" program, you might see 38hex bytes of "header" information. You could skip over this data and decompress the remaining using IMA like so:

```
C:> tpcm -isound.dmp,i,38h
```

Similarly, if, while playing some "unknown" file type, you hear sound, you can "mark" the beginning and end of the cut (using the 'm' key during playback), then use the `-i` option to specify what format, how far in and for how long to play:

```
C:> tpcm -isomefile.dat,p,14469,44100 -r11025 -l16 -s
```

Admittedly, only very few persons will ever use this option to its fullest extent, but, here you go.

`-r`

Use the `-r` option to specify the samplerate for .wav files specified with the `-i` option, and for non .wav files, and for recording. This value can range from 4000 to 88200 for mono files, and 8000 to 44100 for stereo files.

`-l`

Use the **`-l`** option to specify the bit-size of the ram data samples. Use this option when recording sound files or by using the "**`-i`**" option.

`-s`

Use the **`-s`** option to cause the program to treat the **`-i`** specified file or raw PCM files as stereo. This option has no effect on files whose formats are known, unless the file is specified using the `-i` option.

`-B`

Use the `-B` option to specify the size of the DMA buffer. If this number is <= 64, then the value is inferred as KB. Otherwise, it is assumed to be the size of the DMA buffer in Bytes.

> "`-B64`" specifies a 64x1024, or 64KB size buffer
> "`-B64000`" indicates 64000Byte size buffer

To get a 1-second DMA buffer with a 22050Hz, 16-bit mono sound, use "`-B44100`".

`-D`

Use the `-D` option to specify the number of paritITions for the DMA buffer.

The **pcm.com** driver handles number-of-divisions of the DMA buffer that are **not** powers of 2. Thus, `-D5` is valid.

Note that the size of the DMA buffer should be a multiple of the number of divisions, otherwise the DMA controller will transfer the remaining data (which will be, presumably, uninitialized and so noisy).

`-v`

Use the `-v` option to select a picture to display:

`-v1`	tpcm.scr	256-color gradient
`-v2`	tpcm2.scr	me
`-v3`	tpcm3.scr	lame-looking "tape deck" with useless knobs

These must be simple "memory dump" files, 320x200x8bit.

Defaults

Mono vs. **stereo** display is selected **automatically** according to the channels for recognized .wav files.

You can specify multiple files to process, if the -i option is not also used, by naming them at the end of the command line. You can use DOS wildcards: "tpcm *.wav". These files are examined to determine their type, and .wav files have their format information used to specify the number of channels and samplerate, but the command line parameters or default values are used for non-.wav files.

Unfortunately, the `-v` visual display parameters are not reset between .wav files, meaning that if the first file is mono, then the display for all subsequent files is single-channel.

Record Mode

`-R`

This option will put the program into "**record**" mode.

Most options are valid in this mode, with exception of `-l` (looping) and `-iname` ("input file name"), and that only one filename is processed.

The program will attempt to compress in real-time using the format specified with the -R option:

`-RA`	- use ALAW
`-RM`	- use MULAW
`-RI`	- use IMA
`-RW`	- use Microsoft ADPCM (not real-time - but implemented).
`-RP`	- no compression

This output format is compatible with the input, so you can always use **tpcm.exe** to play/record these files, and "**cmpwav.exe**" (from MediaVision) to control their format.

`-V#`

This option will set **Voice**-**Activated** Mode, when used with the "`-R`" record option.

The "`#`" indicates a "**threshold**" value. DMA buffer divisions **not** above the threshold will **not** be stored on disk. The algorithm is this: "if **1/# samples** in the DMA Buffer Division are **greater** than #, then **store** the data onto disk". A value of 2 would mean: "if 1/2 the samples are greater than 2, save it"; 10 would mean: "if 1/10th the samples were greater than 10, save it".

Return Value

Playback

`errorlevel 0`	if the file played with no error.
`errorlevel 1`	if the file played but not properly (IRQ mismatch)
`errorlevel 255`	if it failed to find or load the pcm.com TSR.
other `errorlevel`	syntax error, bad memory, etc.

Record

`errorlevel 0`	if the file recorded with no error.
other `errorlevel`	syntax error, bad memory, etc.

Requirements

Media Vision **Pro AudioSpectrum** (any model).

"**pcm.com**" (or "pcmfun.com").

This program requires a **graphics monitor** with the ability to display 320x200x256 colors in order to **display** the waveform.

Secrets

DMA Buffer Location

It is possible to specify the location of the DMA buffer manually using two options not documented above, however, this can be very dangerous. Use the -S# and -O# options to specify the segment and offset (in hex) of the DMA buffer (use the -B# and -D# option to specify the size and divisions). You can then have the program play or record and direct the DMA buffer to A000:0, thus seeing the PCM data appear on the screen in graphics mode. Interesting? Maybe, but since we can, why not?

If you have the "setdma.exe" and "freedma.exe" programs, you can use them to allocate and deallocate a 16K block of memory from the very top (thereby eliminating any waste). The "setdma.exe" program displays the segment to specify with the tpcm -S option (and to "freedma.exe").

The -S and -O options ask you to confirm that you are manually specifying these values to prevent accidental confusion with the -s and -o options, unless you insert a "!" between the "S" or "O", as in "-S!####" or "-O!####". Using screen memory as the DMA buffer can be "noisy".

Multiple PAS Boards

Also, you can specify a "Board Address" to gain access to multiple PAS boards by using "-b###". Each board requires an instance of "pcm.com" or "pcmfun.com" loaded with the same board address.

"tpcm.exe" will use the first available pcm.com (pcmfun.com) if no particular board address is specified. If an address is given, then the first pcm.com (pcmfun.com) that controls that board will be used. If no suitable handler is found, then the program will install "pcm.com" for that particular board address. The default address is 388h, but the TSR will locate at the first found address.

The following sequence will demonstrate the logic implemented:

```
C:> proas -b388 d:5 q:15 s:1,220,1,5 v:70
C:> proas -b384 d:6 q:11 s:1,230,3,7 v:70
```

These two lines initialize two PAS16 boards, one at 384h, one at 388h

```
C:> tpcm sound.wav        # no handler, load pcm.com at 388h
```

With no handler installed, the program loads pcm.com, and with no board address specified, the default is 388h.

```
C:> tpcm -b384h    sound.wav      # no handler, load pcm.com at 384h
```

With no handler installed for the particular address, pcm.com is loaded for that particular board.

```
C:> pcm -b384 -i          # install handler for board 384h
```

When a handler is installed, that board is the default board when no address is specified.

```
C:> tpcm sound.wav        # is handler, plays on board 384h
```

With a handler installed, and no board address specified, the sound is played on that handler's board.

```
C:> tpcm -b388 sound.wav  # no handler, load pcm.com at 388h
```

A handler is installed, but not for the specified board, so pcm.com is loaded and the sound plays on that board.

```
C:> pcm -b384 -u          # uninstall handler for board 384h
C:> pcm -b388 -i          # install handler for board 388h
```

The default board to play on when no address is specified was 384h, and is now 388h.

```
C:> tpcm sound.wav        # is handler, plays on board 388h
```

No address specified, plays on first handler's board.

```
C:> tpcm -b384 sound.wav  # no handler, load pcm.com at 384h
```

No handler for that board installed, so pcm.com is loaded and the sound is output on that board.

```
C:> pcm -b384 -i          # install handler for board 384h
```

Now there are two handlers installed, one for each board.

```
C:> tpcm sound.wav        # play on first found (384h PAS)
```

The first handler found (is the last one loaded) is at at 384h, so since no address is specified, output comes out of the board at 384h.

```
C:> tpcm -b384 -T sblast.wav      # play sound on board 384h TB
```

Address specified, and handler exists.

```
C:> proas -b380 d:7 q:12 s:1,220,0,3
C:> pcm -b380 -i
```

Uh, oh, another board is initialized, at 380h, and a handler is installed for it.

Since this is the last handler, it is the first found, and so is the default board when no address is specified.

```
C:> tpcm sound.wav
```

No address specified, output on 380h.

Programmer Output

dasj;l;kare-0423m saldf, 3129;as,.4pdfp

I did that!

Whew! If you are reading this, then you are my kind of user!

I feel like I can tell you things... things I haven't told anyone else!
Except for everyone who will listen, that is!

Seriously (why do I keep backspacing and undoing, instead of just getting on with it!). (Okay, so I just finished the "copyright" stuff below... talk about conjuctive phrasing!)

Okay, so what I really want to say is, if you have gotten this far, don't stop now!

First, a little about the programmer of this stuff. Me. SRS.EXE, PCM.COM and PCMFUN.COM, TPCM.EXE, proas.exe, WAVCMP.LIB... all I can say is: "there you go".

I use'm all the time.

My only regret is that I only use them to build them! I wish, instead, I was in your place (maybe), and had cause to use them! I can imagine being a recording engineer, or database programmer, and having a real need for this stuff. Or, as an independent developer writing a game or other application.

Tell me about some other program that will record at 88,200 Hz, please! Or will stat all .wav or .voc files, please! Or (un)compress!

Now that I can play the SB and record it and all other inputs on the PAS, in the background(!), well, my head is five feet wide!

Here's a little joke I played for awhile: I recorded random FM at 88,200, then played it back at 12,000-4,000 (nearly so-to-truly subsonic) whenever people came into my cube regarding bugs in "-------.exe", in the background, so no one knew! (Maybe I make people uncomfortable, anyway.)

The following text is merely for informational purposes, and really only for those that use "tail" on documents to see what is at the end, and I don't want those of you who read all of the above to be cheated by those who skip to the last screen!

If you would like to become a registered user of this and its associated software, please post some mail that includes your first name and your age to

Media Vision
3185 Laurelview Court
Fremont, CA 94538
Attn: Software Engineering, DOS

or fax the same to

(510) 226-2582

Acknowledgements

Programmer: Bart Crane