# FTAP: A Linux-based program for tapping and music experiments

## User's Guide

## Version 2.1.05



Steven A. Finney

November 1, 2001

# Contents

# 1 Introduction

FTAP (F[eedback] or F[lexible] Tap) is a Linux-based program (written in 'C') for running experiments involving sequential finger taps (on single or multiple keys), such as tapping or music experiments.[1] It uses MIDI devices for keystroke input and sound output, and collects and outputs MIDI data with millisecond resolution. FTAP provides highly configurable control of auditory feedback responses to input keystrokes on the basis of a simple ASCII-text parameter file, including many capabilities which would usually be done by low-level programming. These features include flexible control of delayed auditory feedback (DAF), pitch alterations, combination of delay and pitch alterations, controlled perturbations to either pacing tones or feedback responses, and other capabilities. In addition, FTAP includes a flexible internal metronome for pacing or synchronization signals and the ability to play fixed sequences from external files.

This User's Guide is intended to serve as an introduction to FTAP; more complete detail is included in an associated Reference Manual, which you should read thoroughly if you decide to use FTAP. In addition, Finney (2001a) and Finney (2001b) contain useful information. Finney (2001a) overlaps significantly with this User's Guide (and is better written); please cite Finney (2001a) when writing up any research which makes use of FTAP.

FTAP can run experiments such as synchronization/continuation tasks at various rates (Wing and Kristofferson, 1973), synchronization tasks in which the feedback from one tap (only) is delayed by a small number of milliseconds (Wing, 1977), delayed auditory feedback experiments with tapping (Chase et al., 1961; Finney, 1999), musical keyboard experiments in which pianists perform under conditions of delayed auditory feedback, pitch altered feedback, and silence (Finney, 1997), synchronization tapping with small perturbations in the pacing signal (Repp, 2000), synchronization tapping with delayed feedback (Aschersleben and Prinz, 1997), polyrhythmic tapping (Jagacinski et al., 1988), experiments combining synchronous and delayed feedback (Ruhm and Cooper, 1964),

[1] FTAP began as a program I wrote for an SGI Indigo which I used for my research as a graduate student at Brown University. I thank Jim Anderson, David Ascher, Peter Eimas, Mike Tarr, and Bill Warren for their assistance during this time. The port to Linux and many important enhancements were completed during a post-doctoral fellowship at Ohio State University; I thank Caroline Palmer, Pete Pfordresher, and Shane Ruland for their encouragement and assistance. I also thank David Huron and Doug Reeder for providing a web home for FTAP at http://csml.som.ohio-state.edu/ftap.

The contents of this document should be accurate and, in conjunction with the Reference Manual, relatively complete. However, the documentation is not as well-organized or well-written as it could be.

and many others. Sample parameter files for many of these are included in the FTAP distribution.

An FTAP trial is described by an ASCII-text parameter file, which specifies a subset of the full set of 47 parameters. A "parameter" is a description of some characteristic of the experimental situation, such as whether there is feedback to the user's tap, whether the feedback is delayed, the amount of the delay, the time between beats of the metronome, etc. One particular feature which contributes to FTAP's flexibility is the ability to define "trigger events", which change a parameter value in mid-experiment based on elapsed time, metronome count, or keystroke number. Thus, e.g., auditory feedback can be turned on, turned off, changed, or perturbed during a trial; the metronome can initially be on and then be turned off, experiments can be automatically terminated, etc. FTAP itself runs a single (possibly long and complex) experimental trial; i.e., "FTAP [paramfile]" will control auditory feedback for a series of subject taps, and store information about the taps and feedback in an ASCII file. Use of FTAP in a research setting would typically involve a wrapper program of some sort (e.g., a shell script or Python program) which would randomize trials, locate parameter files, etc.

The FTAP output file provides millisecond-precision timestamped information for all events (keystrokes, metronome, and feedback events) in a columnar text format, allowing easy access to the relationships between different types of events. FTAP also provides on-line self-diagnostics which can help verify adequate performance.

FTAP typically uses a MIDI keyboard as an input device, and one or more MIDI tone generators (ideally polytimbral) as output devices. FTAP's input and output to these devices is in the form of MIDI messages which specify note (which key is being pressed or released) and key velocity (force); all timing is taken care of by FTAP itself. The use of MIDI provides a standard protocol and readily available input devices, such as most electronic keyboards (which can be used for either tapping or music experiments). It also allows transparent application to other MIDI devices, such as electronic drumpads.

The basic design of the FTAP program is shown in Figure 1. MIDI data comes from a MIDI input device (typically a keyboard). The data is possibly mapped or transformed (e.g., delayed or altered in pitch), and is then sent to a MIDI output device (a tone generator, which may or may not be the same as the input device). In addition, FTAP can generate its own MIDI output data (e.g., metronome or pacing tones).

The FTAP distribution includes binaries, full C source code, documentation, sample parameter

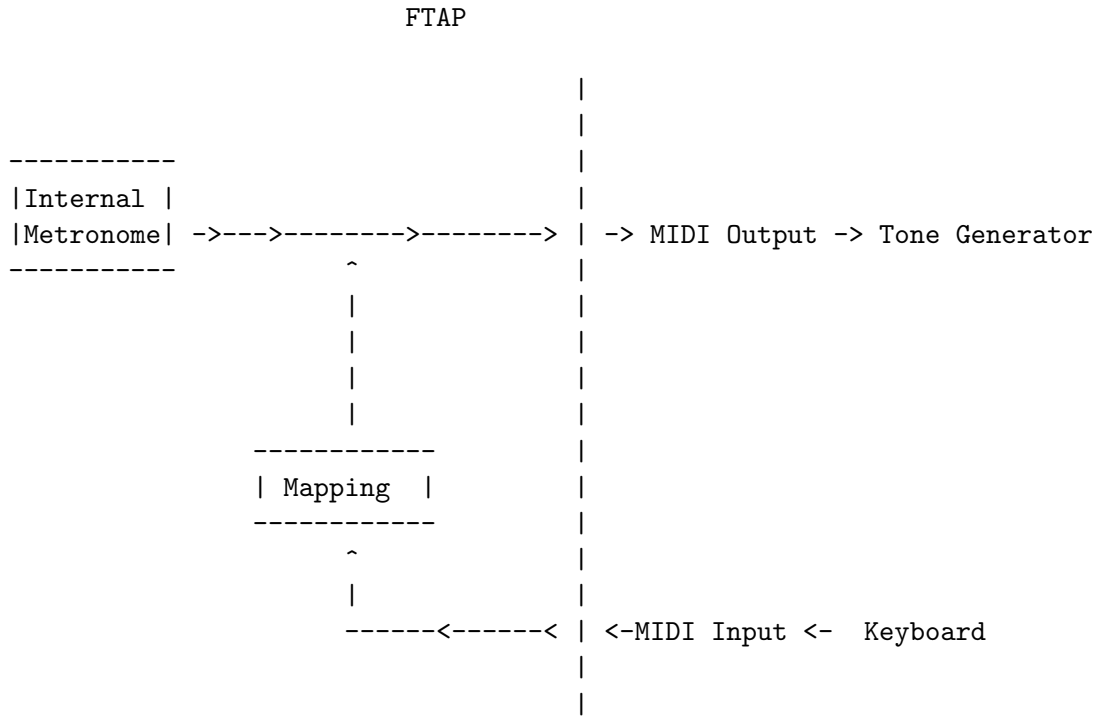


Figure 1: Architecture of FTAP

files, and supporting utilities. It is provided at no cost (but without warranty) under the GNU Public License from the FTAP web site at http://csml.som.ohio-state.edu/ftap, or by contacting the author. Send requests, comments, or suggestions to sf@csml.som.ohio-state.edu.

## 2 Configuration and Usage

My configuration for running FTAP is a a 200 MHz Pentium with a Creative Soundblaster 16 sound card, running the RedHat 6.2 Linux distribution. See the Reference Manual for some configuration issues, and let me know about any problems you encounter.

**IMPORTANT NOTE**: FTAP currently will not work with the OSS MIDI drivers distributed with standard Linux, as FTAP makes use of the select() system call, which appears not to be supported (the stock drivers have real-time timing issues in any case; see Finney (2001b) or the Refernce Manual). I use the 4front MIDI drivers (http://www.4front-tech.com); the ALSA drivers may also work, but I haven't tried them. In addition, FTAP requires that real-time clock support be loaded; this is the default on RedHat, but may require some additional command (e.g., "insmod

rtc") on other distributions. Ideally, evaluation use could be done with the stock drivers and without RTC; this may (or may not) be incorporated into a future release.

The FTAP distribution includes a binary for Intel Pentium machines, as well as source code. To run FTAP, you must have a working external MIDI interface (e.g., a sound card with a MIDI In and Out port, and an installed Linux driver). To run the demonstration and sample files in the "params" directory in the distribution, you will need to attach a MIDI keyboard which transmits on MIDI channel 1, and a tone generator which receives on MIDI channel 1.[2] Computer-internal MIDI sound generation has not been tested (but might work); all of FTAP's MIDI I/O is done through "/dev/midi". If you plan to collect real millisecond-precision data, I recommend (1) testing the throughput of your MIDI configuration by using a loop cable, and (2) running FTAP with root privileges (e.g., configured as setuid 'root'). See Section 7.

## 2.1 Usage

FTAP takes a parameter file (ASCII text) as an argument, and runs a long trial based on those parameters. Simple usage is the following:

```
ftap <parameter file>
```

On completion of a trial, some timing diagnostics will be printed out. These will be explained below, and will give you some idea of how efficiently FTAP is running on your particular system.

## 2.2 Sample Parameter Files

If you want to try FTAP out before reading the rest of this manual, go to the "params/demo" or "params/sample_experiments" directory of the distribution and pick a file that looks interesting to you (all have short explanatory header comments which start with the "#" character). For example, in the "params/demo" directory, you could type

```
../../bin/ftap metronpat3
```

If your system is properly configured (MIDI interface, driver, attached MIDI and amplification equipment), and you have muttered the right magic incantations, you should hear a patterned

[2] A single keyboard can be used for both input and output if internal sound generation can be turned off so that there is no uncontrolled synchronous feedback (e.g., by disabling MIDI LOCAL mode.)

sequence, and there will be an output file named "metronpat3.sub.block.trial.abs" in the current directory. Otherwise, see the Reference Manual (particularly the troubleshooting section).

Experimenting with, modifying, and running the supplied parameter files is probably the best way to explore FTAP's capabilities. In most of these files the metronome and feedback use the same tone generator voice (MIDI channel), though for some of the experiment files it is assumed that there is a white noise voice on MIDI channel 2. In the "params" directory, the "demo" directory contains parameter files which demonstrate some of FTAP's more interesting capabilities. Among them are the following:

1. The 'metronpat*' files produce various metronome sequences.
2. The '*pitch' files demonstrate pitch manipulations.
3. 'twochan' demonstrates use of the two feedback channels (that is, both synchronous and delayed feedback to a single keystroke)
4. 'trigger' and 'mettrigger' demonstrate various triggering capabilities.

The "sample_experiments" directory contains parameter files implementing experiments similar to ones in the literature (Aschersleben and Prinz, 1997; Finney, 1997; Finney, 1999; Wing, 1977; Repp, 2000).

# 3 FTAP Basics

"Auditory" output from FTAP is in the form of MIDI messages, which are sent to a MIDI tone generator. FTAP allows configuring the MIDI channel, note number, and MIDI velocity of all output signals, but programming the tone generator to respond appropriately to these is the user's responsibility; that is, the external tone generator itself must be set up to respond appropriately for the desired sound. E.g., FTAP will allow specifying the MIDI velocity for an output message, but the actual resulting sound level depends on how the tone generator is set up.

## 3.1 Parameters: General

Parameter specifications in a text file specify FTAP's behavior. There are 3 types of parameters (integer, string, and array), as well as trigger events which can change integer parameter values during

the course of a trial. Any parameters not explicitly specified in the input file will default to rational (and documented) inactive values. Each parameter controls one specific aspect of the experiment; in general, these may be flexibly combined in interesting ways. The ordering of parameters within an input file is of no significance.

The following is an example of a integer and an string parameter as they would appear in the input file; the parameter name is on the left and its value is on the right. The first example specifies that the metronome should be turned on; the second specifies the file from which a sequence of pitches will be read:

```
METRON_ON                   1
PITCHSEQ_FILE               pitchfiles/scale
```

Different conventions are used for different integer parameters. For some integer parameters (e.g., FEED_ON, which specifies whether feedback is sounded in response to keystrokes), values of 0 and 1 are used to signal "off" or "on", respectively. In other cases (e.g., FEED_LEN), a 0 value signals that feedback should use the keystroke times from the subject to determine the length of the sounded feedback, while a non-zero value specifies a fixed-duration sound regardless of the time the key is held down. Finally, values of an integer parameter may signal choices between a number of different possibilities (e.g., delay modes). Most integer parameters can be changed during the course of the experiments by the use of trigger events (see below).

### 3.2 Metronome

For synchronization experiments, or to provide a reference point for musical tempos, FTAP provides a metronome (or pacing signal). A metronome pulse is of fixed pitch, loudness, and length, and occurs at a fixed ISI (here defined as MSPB: milliseconds per beat). However, any of these characteristics may be changed by a trigger event while the trial is running. It is also possible to impose a rhythmic pattern on the metronome.

In the simplest case, the MIDI channel, (fixed) note, (fixed) volume, and (fixed) sound length are programmable, as well as number of milliseconds per beat (MSPB) of the metronome. The following parameters specify a 2 beats per second (500 ms/beat) metronome on MIDI channel 1, with a fixed MIDI velocity of 90, MIDI note number of 86; and with metronome pulses lasting for 30 ms (i.e., there is 30 ms between the MIDI Note On and the MIDI Note Off).

```
METRON_ON          1
MSPB               500
MET_CHAN           1
MET_NOTE           86
MET_VEL            90
MET_LEN            30
```

The metronome can also be patterned. The array parameter specification below will cause the metronome to occur in a repeating 4 beat pattern of 3 sounds and a pause. (Array parameters are specified by a count field giving the number of elements, followed by the elements themselves. Here, 1 indicates a sounded beat and 0 indicates a silent beat).

```
MET_PATTERN_ARRAY  4 1 1 1 0
```

The next specification would cause the metronome to cause a louder sound (higher MIDI velocity) on the first beat of each pair of beats:

```
MET_VEL_ARRAY  2 110 90
```

Pitch, duration, and MIDI channel of each metronome beat in a pattern can be similarly specified.

### 3.3 Feedback

The characteristics of the feedback event to a user's keystroke are highly controllable. At any given time, feedback may be on or off (the FEED_ON param set to 1 or 0, respectively). The pitch may be the same as the input keystroke value, or may be one of a number of pitch alterations. The velocity (loudness) may follow the keystroke, or may be fixed. The feedback timing may be synchronous or delayed.

In a simple case, the output just mirrors the keyboard input, as in normal electronic keyboard performance. The parameters below output the user's keystrokes as feedback on MIDI channel 1 (as one might want to do for a music experiment); the value of 0 for FEED_PMODE, FEED_VMODE, and FEED_LEN means "use input values" for pitch (note), velocity (loudness), and note length.

```
FEED_CHAN       1
FEED_PMODE      0
FEED_VMODE      0
FEED_LEN        0
```

For a tapping experiment, a fixed volume, fixed note, and fixed length pulse tone might be preferred. The following parameters will send out out a fixed length (100 ms), fixed pitch (E5, or midi note 76), and fixed volume (MIDI velocity 90) tone in response to a user's keystroke. The value of 1 for FEED_PMODE and FEED_VMODE indicates a fixed output value for pitch and velocity, respectively; the specific fixed values are provided by FEED_NOTE and FEED_VEL.

```
FEED_CHAN       1
FEED_PMODE      1
FEED_NOTE       76
FEED_VMODE      1
FEED_VEL        90
FEED_LEN        100
```

More interesting are the delay and altered pitch mappings. The FEED_DMODE parameter controls the delay feedback mode. A value of 0 will give synchronous feedback, while 1 will delay feedback for the keystroke by a fixed interval (determined by the FEED_DVAL parameter). The following parameters would delay auditory feedback to a user's keystrokes by 250 ms.

```
FEED_DMODE     1
FEED_DVAL      250
```

The pitches which occur in response to keystrokes are controlled by the FEED_PMODE parameter. If FEED_PMODE = 0, the pitch from the user's keystroke will be sounded. FEED_PMODE = 1 will give the fixed value FEED_NOTE in response to *any* keystroke, while FEED_PMODE = 4 gives quasi-random pitch output. Other capabilities (such as playing an arbitrary sequence of notes from an input file) are also provided.

An interesting capability is that pitch alterations and delay can be freely combined, simply by setting FEED_PMODE and FEED_DMODE appropriately. E.g., setting FEED_DMODE to 1 and FEED_PMODE to 4 would give random pitch output that is also delayed (a manipulation used in Finney, 1997).

### 3.4 MIDI Channels

The MIDI channel parameters (e.g., FEED_CHAN, MET_CHAN) allow specification of different MIDI channels for the feedback and the metronome. The use of this feature, in conjunction with a tone-generator which allows simultaneous production of voices with different timbres (most modern synthesizers and tone generators, and some common older ones, e.g., a Yamaha TX-81Z), allows

sounding a different timbre for feedback and metronome, providing a clear separation of the two. E.g., MET_CHAN could be set to 1 and FEED_CHAN could be set to 2 for a tone generator which generates a short percussive sound on MIDI channel 1 and a piano sound on MIDI channel 2.

### 3.5 Trigger Events

An important part of FTAP's flexibility is the ability to alter FTAP's behavior during the course of a trial. One simple example occurs with the standard continuation paradigm, in which the metronome sounds for a specified number of beats and then stops, while the subject continues tapping. A more complex case is exemplified by Wing (1977), who investigated the effect of delaying the feedback for a single keystroke in the midst of otherwise uniformly delayed feedback. FTAP allows for changing the value of an integer parameter based on (1) a specific keystroke number, (2) a specific metronome beat, or (3) a specific time (in milliseconds) since trial start.

In the parameter file, triggers are specified by the keyword "TRIGGER". The first field following the word "TRIGGER" is a unique trigger ID (for identification in the output file); this is followed by the trigger type: K(eystroke), T(ime), or M(etronome). The fourth field is the count for the trigger (keystroke number, number of milliseconds, or metronome count), and the last two fields are the name of the parameter to change and the new value. The following specification would turn the metronome off on the 16th beat(e.g., for a continuation paradigm), while simultanously turning on feedback to the subject's keystrokes. Here, multiple parameter changes occur in response to a single event:

```
TRIGGER 2 M  16 METRON_ON 0
TRIGGER 3 M  16 FEED_ON  1
```

A special pseudo-parameter "END_EXP" allows terminating a trial. The first trigger below would terminate the experiment on the subject's 56th keystroke; the second would end the experiment after 30 seconds:

```
TRIGGER 1 K  56    END_EXP 0
TRIGGER 2 T  30000 END_EXP 0
```

The next pair of events (with proper setup) would cause feedback to the user's 40th keystroke to have a delay of 75 millisecond, with all following keystrokes delayed by 40 ms (Wing, 1977):

```
TRIGGER 2 K 40 FEED_DVAL        75
TRIGGER 3 K 41 FEED_DVAL        40
```

There are some subtleties involving triggers. E.g., although the keystroke counting mechanism is useful, it is not robust to subject errors. If you wish to do an FTAP manipulation on a certain note in a musical piece (say, the 20th note, which you expect to be E4), you can set a trigger for keystroke 20, but if the subject misses or adds a note, the effect may not occur at the desired place: the trigger occurs on keystroke 20, not the note E4.

# 4 Sample Experiment

Figure 2 shows a complete parameter file for a simple synchronization/continuation tapping experiment. The trial starts with fifteen metronome pulses at a 250 ms rate, and ends after 20 seconds. Keystroke data is collected throughout, but synchronous feedback to the user's keystrokes (fixed pitch, volume, and length) occurs only during the continuation phase; it is off during synchronization. The VEL, NOTE, LEN and CHAN parameters must be set to reasonable values for your tone generator setup. To turn synchronous feedback on throughout, set FEED_ON to 1 (trigger #1 can then be removed). To change this to an experiment with delayed auditory feedback, set FEED_DMODE to 1, and add a FEED_DVAL parameter for the desired delay.

# 5 Output File Format

FTAP's output file names have a suffix of ".abs", and obligatorily encode the parameter file name, as well as the subject, block, and trial number (these are usually provided as parameters when FTAP is run). An output file name might be "250p.s1.b1.t7.abs". All of this information is also contained in the file itself.

An FTAP output file starts with a listing of the parameters, as well as some diagnostic information; all these lines are identified by a '#' character in the first column. The data lines follow this header. Only note events (keystroke, feedback, and metronome) will be described here; the data lines also include MIDI controller events and indications of FTAP trigger actions. Data lines always have 8 columns, and the type of data can identified by the character in column 8. The following is a sample output data line; fields are separated by tabs:

```
9630    D       1       86      D6      75      12      K
```

```
FEED_ON       0
FEED_CHAN     1
FEED_PMODE    1
FEED_NOTE     76
FEED_VMODE    1
FEED_VEL      90
FEED_LEN      100
FEED_DMODE    0


METRON_ON     1
MSPB          250
MET_CHAN      1
MET_NOTE      86
MET_VEL       100
MET_LEN       30

TRIGGER 1 M  16 FEED_ON 1
TRIGGER 2 M  16 METRON_ON 0
TRIGGER 3 T  20000 END_EXP 0
```

Figure 2: Sample Parameter File: Synchronization/Continuation Paradigm

The first field is the elapsed millisecond time of the recorded event, relative to trial start. The second field is "D" or "U", depending upon whether the event is a key down or key up (that is, MIDI NoteOn or NoteOff). The 3rd field is the MIDI channel of the event. The 4th field is the MIDI note number, while the 5th field is a letter representation of that note. The 6th field is the MIDI velocity, while the 7th is the sequence number for input keystrokes. The 8th field defines the type of event: "K" for a user keystroke, "M" for metronome events, and "F" for feedback events. This allows filtering the data file on the basis of event type; e.g., if the only data of interest are the keystroke down times, it's simple to isolate such events by extracting the lines with a 'K' in column 8 and a 'D' in column 2.

Figure 3 displays an editted version of the output file for the experiment specified in Figure 2; only the first second of the synchronization phase is shown. Note the characteristic negative asynchrony found in such experiments (e.g., the key press at time 966 precedes the pacing signal at time 1000).

```
# TIME                Wed Sep  6 12:07:50 2000
# VERSION_NUMBER      2.1.02
# SCHED_AV            0.49
# SCHED_MAX           1
# FEED_ON             0
# FEED_CHAN           1
....
# MET_VEL             100
# MET_LEN             30
# METRON_ON           1
# MSPB                250
....
# Stamp   UpDown   CH    Note#   NoteNm   Vel    Seq   Type
250       D        1     86      D6       100    0     M
280       U        1     86      D6       0      0     M
447       D        1     60      C4       117    1     K
500       D        1     86      D6       100    0     M
530       U        1     86      D6       0      0     M
561       U        1     60      C4       0      0     K
721       D        1     60      C4       127    2     K
750       D        1     86      D6       100    0     M
780       U        1     86      D6       0      0     M
856       U        1     60      C4       0      0     K
966       D        1     60      C4       124    3     K
1000      D        1     86      D6       100    0     M
....
```

Figure 3: Sample output file: Synchronization paradigm

# 6 More Specialized Capabilities

This section briefly describes some further FTAP capabilities; some of these are likely to be of use primarily for tapping experiments (e.g., involving a single finger on a single key), while others are oriented more towards music experiments (multiple fingers, multiple keys).

## 6.1 Random Delays

Additional values for FEED_DMODE (in conjunction with other parameters) allow for delaying each keystroke on on a random basis. The following parameters will cause the delay for each keystroke to be randomly selected from the set 100, 200, and 300. Note that such random delays can reorder output events relative to their corresponding inputs; these can be detected by inspection of the output file.

FTAP retains the note length (time between key down and key up) for each input/output pairing, even when the delay is random.

```
FEED_DMODE  2
NRDELAYS  3  100  200   300
```

### 6.2 Masking Noise

It is possible to define a masking signal which goes on at the start of the experiment, and off at the end. I've used this with a white noise voice on a MIDI tone generator to mask physical key noise (where the white noise in the example is programmed on MIDI channel 2):

```
MASK_CHAN           2
MASK_NOTE           64
MASK_VEL            35
MASK_ON             0
```

### 6.3 Second Feedback Channel

The above examples have referenced a set of "FEED_" parameters, which implements what I have chosen to call a feedback channel (a possibly unfortunate term, as it has nothing to do with MIDI channels). FTAP also implements a second FEED2 feedback channel, which can be configured largely independently of the first one. This can be used to either provide 2 feedback events for one keystroke (for instance, giving both synchronous and delayed feedback to a single keystroke), or to provide different feedback for different parts of the keyboard under control of a specified split point. The FEED*_ON parameters controls whether a particular feedback channel is used: set to 0, the channel is inactive; set to 1 the channel is active. Setting both FEED_ON and FEED2_ON to 1 (and configuring the parameters for each channel differently, e.g., one delayed, one not) will cause two output events for each keystroke. Setting the FEED*_ON parameters to 2 or 3 will cause them to sound only above or below a MIDI note value specified by the SPLIT_POINT parameter, allowing different feedback effects for different parts of the keyboard.

### 6.4 Pitch Changes

Many pitch mappings (mappings of input keystroke to output note) are available through use of the FEED_PMODE parameter:

1. Correct pitch mapping (same as input note value). [FEED_PMODE = 0]

2. Fixed note pitch mapping (all input events are mapped to the same output note, specified by FEED_NOTE). [FEED_PMODE = 1]

3. Reversed pitch mapping: the keyboard is effectively reversed around middle C, with low notes on the right and high notes on the left. [FEED_PMODE = 2]

4. Semi-random pitch mappings, both consistent (the same key always gives the same note) and inconsistent (two successive presses of a key will give different notes). [FEED_PMODE = 3 or 4 ]

5. Pitch sequences: the notes that will be played for each succeeding keystroke are read in from a user-specified file (specified by the PITCHSEQ_FILE parameter) [FEED_PMODE = 5]

### 6.5 Controllers

For some music experiments, recording pedal data is necessary. Pedal information is encoded as a MIDI controller event, and many other controllers exist, such as aftertouch, pitch bend, and breath control. Although FTAP records and outputs such controller information, it is less clear that sophisticated feedback manipulations of such controllers serves a useful purpose (in addition, it potentially gets complicated). The working compromise is that such controller events are output and delayed in accordance with the FEED_ON and FEED_DMODE parameters, but are otherwise unmodified. Output file lines for controller events contain a 'C' in column 8 for controller input, and 'G' for controller output.

### 6.6 Click Files

'Click files' provide a way of playing a fixed, prepared stimulus sequence (this is *not* FTAP's strength). The stimulus files are in the same format as FTAP's output files (that is, 8 columns of ASCII data), and the easiest way to prepare them may be to get FTAP to record a performance or generate an approximation using the metronome features, and then edit the resulting file by hand.

Click files were originally implemented to play simple rhythmic patterns, and were thus conceived of as being simple "click" sequences. However, complex musical pieces could also be presented in this way, allowing the collection of on-line keyboard responses while a musical piece is playing, with precise recording of the timing relations between notes in the piece and the subject responses.

However, there is currently no utility for converting standard MIDI files into FTAP format files, nor any way of playing MIDI files themselves as sequences.

# 7 Real-time Configuration and Diagnostics

The stock Linux MIDI drivers do not support millisecond-resolution output (at least with cards that do not provide interrupts, such as the SB-16); in addition, the stock drivers do not support the select() system call on "/dev/midi", which is FTAP's mechanism for doing non-blocking reads (a Reference Manual Appendix goes into more detail about the story here). Thus, unfortunately, the stock Linux drivers will not work with FTAP, even for evaluation purposes (this may change in the future). I have been using the 4Front MIDI driver; the ALSA drivers may also work, but I haven't tried them. For better timing precision for actual data collection, run FTAP with Linux superuser privileges; see the FTAP Reference Manual and Finney (2001b) When configured this way, FTAP provides millisecond precision data collection on a Linux operating system; this may surprise those who are aware of the potential problems of real-time data collection on a complex multi-user, multi-tasking operating system. However, the combination of modern fast hardware (e.g., 500 Mhz Pentium based computers), real-time support provided by standard Linux, and careful coding makes such precision possible. For actual data collection, I recommend running on a system with as many services as possible disabled, and with no other users logged on, and configuring FTAP as setuid 'root' so that it can take advantage of the "real-time" capabilities of Linux. In addition, because the performance of the MIDI interface itself may vary with the MIDI hardware and driver being used, I recommend running a MIDI throughput test using a loop cable. Both of these issues are covered in the Reference Manual.

In addition, the timing of a given run of FTAP is indicated in output diagnostics printed both to the screen and to the output file. Proper real-time functioning of FTAP requires that the internal scheduling routines be called at least once a millisecond (this is usually assumed to be sufficient precision for human behavioral experiments). The diagnostics indicate whether there were any instances in which this was not the case by tallying the number of scheduling intervals $> 1$ ms, $> 5$ ms, and $> 10$ ms; the following is a sample printout:

```
Experiment ended
Mean time between sched()'s (ms): 0.49, schedcnt: 2205
> 1 ms: 0, > 5 ms: 0, > 10 ms: 0, max: 1 ms
```

Since the the maximum time between scheduler calls on this trial was 1 ms, FTAP ran with millisecond-level precision. This is usually the case on a properly configured system, but having the information displayed allows the user to immediately detect any problems.

# 8 Concluding Remarks

FTAP is provided as Open Source, in hopes that people will find it useful. If enough people find it useful, hopefully some of them will also help improve it.

- If you find any bugs in FTAP, let me know. If you've fixed a bug, send me the fix and I'll incorporate it into the standard source.
- If you have problems getting FTAP to work on your system, let me know. If you have gotten it to work, and you're using a different Linux release, hardware architecture, or MIDI card than I've listed in the Appendix of the Reference Manual, let me know so I can add it.
- If there are things FTAP can't do which you'd like it to do, let me know. If you code a useful enhancement, let me know and I'll consider incorporating it into the standard source. However, I am concerned about keeping the core of FTAP relatively simple, so eventually there may be a standardized mechanism for special-purpose "extensions".
- If you have created parameter files for an experiment you think other people might be interested in and want to make them public, let me know and I'll set up a "Current Experiments" section on the web page.

Steve Finney
sf@joplin.psy.ohio-state.edu or sf@csml.som.ohio-state.edu
http://csml.som.ohio-state.edu/ftap

# References

Aschersleben, G. and Prinz, W. (1997). Delayed auditory feedback in synchronization. *Journal of Motor Behavior*, 29:35–46.

Chase, R., Rapin, I., Gilden, L., Sutton, S., and Guilfoyle, G. (1961). Studies on sensory feedback II: Sensory feedback influences on keytapping motor tasks. *Quarterly Journal of Experimental Psychology*, 13:153–167.

Finney, S. A. (1997). Auditory feedback and musical keyboard performance. *Music Perception*, 15:153–174.

Finney, S. A. (1999). *Disruptive Effects of Delayed Auditory Feedback on Motor Sequencing*. PhD thesis, Brown University.

Finney, S. A. (2001a). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods, Instruments, and Computers*, 33:63–72.

Finney, S. A. (2001b). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, and Computers*, 33:167–173.

Jagacinski, R. J., Marshburn, E., Klapp, S. T., and Jones, M. R. (1988). Tests of parallel versus integrated structure in polyrhythmic tapping. *Journal of Motor Behavior*, 20:416–442.

Repp, B. H. (2000). Compensation for subliminal timing perturbations in perceptual-motor synchronization. *Psychological Research*, 63:106–128.

Ruhm, H. and Cooper, W. (1964). Influence on motor performance of simultaneous delayed and synchronous pure-tone auditory feedback. *Journal of Speech and Hearing Research*, 7:175–182.

Wing, A. (1977). Perturbations of auditory feedback delay and the timing of movement. *Journal of Experimental Psychology: Human Perception and Performance*, 3:175–186.

Wing, A. and Kristofferson, A. (1973). The timing of interresponse intervals. *Perception and Psychophysics*, 13:455–460.