CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS MAY1988

Department of Music
Report No. STAN-M-47

# THE CONDUCTOR PROGRAM AND MECHANICAL BATON

M.V. MATHEWS, CCRMA
D.L. BARR, AT&T Bell Labs, Murray Hill N.J.

Research sponsored in part by
The System Development Foundation

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

# THE CONDUCTOR PROGRAM AND MECHANICAL BATON

M. V. Mathews, Music Department, Stanford University
and
D. L. Barr, AT&T Bell Labs, Murray Hill NJ

## Principals of Operation of the Conductor Program

The conductor program is intended to be used for the performanc of music on a digital synthesizer attached to a small computer. Normally synthesizers are played with keyboards. With the conductor program, a keyboard is not used. Instead the perform conducts with a mechanical baton called a daton. The score of the music in in the memory of the computer. The program can be viewed either as a sophisticated sequencer or as an intelligent musical instrument.

The program simulates a number of the functions of an orchestra following the baton of a conductor. The performer sets the tem by beating time on the daton. He can change the tempo from instant-to-instant and the computer will follow his tempi closely. The performer can also control other interpretable qualities of the music such as the loudness and the balance of the various voices. Control is exercised by where he hits the daton or how he moves a joy stick which is also available to hi

The score in the computer memory contains the pitches and durations of the notes to be played. Consequently the performe does not have to select the pitch of any note by a physical gesture. Elimination of pitch selection eliminates one of the traditional tasks of the performer, thus making part of his job easier. In much music, the performer has no interpretive choic about pitch in the sense that if he changes a pitch from that written by the composer it is considered to be a performer erro Since the performer has no real choice about pitch and since pitch selection can be a very demanding task, it seemed ideal t give this task to the computer. By contrast, duration and loudness are subject to much more performer interpretation and these tasks are left under his direct control in the conductor program.

## Hardware

A diagram of the equipment on which the conductor program operates is shown on figure 1. The central device is a small computer of the IBM PC type, specifically an AT&T 6300 which us an Intel 8086 processor. However, the program can easily be adapted to most IBM clones. The not completely standard featur required by the program is a millisecond clock which we have obtained by changing the speed of one of the existing computer clocks. The program can probably be converted to other popular personal computers such as are made by the Apple company. However, the IBM clones are particularly convenient since they have a backplane with slots into which special cards can be

inserted.

Two such special card are used by the conductor program. Sign
from the various sensors, the daton, the joy stick, and the kn
are a-to-d converted and put into the computer memory space by
data translation card. Signals to the synthesizer are sent vi
midi cable and a midi card, the Roland 401, generates these
signals. Although the midi card can be used to either send or
receive signals from the synthesizer, in this program it is on
used to transmit to the synthesizer.

The synthesizer currently used is a Yamaha 816, but as almost
current synthesizers can be controlled with midi, almost any c
be used without change in the program.

The data translation card is a 2801B (Data Translation company
It can be used in a number of modes, but the program connects
to one of the dma input channels of the computer. The 16
analogue inputs going to the card are sampled sequentially at
16KHz rate and each sample is stored in a location in the
computer memory space. Thus each analogue input is sampled onc
per millisecond. The a-to-d converter has an accuracy of 12
bits. Once started, the data input process proceeds without
further action by the computer central processor. The conduct
program can use the current value of any input simply by readi
an appropriate memory location.

Of the three input devices, the daton is the most novel. It
consists of a light rigid plate about 14 inches square and an
inch thick. In order to be both light and rigid, it is made a
a laminate with fiberglass reinforced plastic on upper and low
surfaces and a core of honeycomb paper material. This type of
construction is often used in airplanes.

The plate is supported at its four corners by four strain gaug
Each time the plate is struck, four electric pulses are genera
by the gauges. Circuitry to analyze the peak value of these
pulses computes both where and how hard the plate was hit. Fo
example, the strength of a hit is the sum of the four pulses.
The y position of the stroke is the sum of the pulses at corne
2 and 3 (see figure 1) divided by the sum of all four.

The accuracy of the daton is about 5% of the full range in any
dimension. This accuracy is sufficient for most conducting
purposes. The time constants of the plate are such that it ta
about 5ms to make a reading and perhaps 15ms more to stabilize
before a subsequent stroke occurs. The plate should be hit wi
a relatively soft drumstick in order work properly.

The daton only sends information to the computer at the instan
that it is struck. Such "percussive" information is appropria
for controlling time and for controlling the parameters of
percussive timbres, for example for the loudness of a piano-li
timbre. However, it is not sufficient to control continuous
factors such as the variation of the loudness or vibrato in a
violin-like tone within a single note. For this purpose, a joy

stick, which can send continuous information to the computer ha
been built. The joy stick is a simple mechanical linkage whicl
connects two potentiometers to the joy stick so that the voltag
from the potentiometers are proportional to the x and y positic
of the stick. The human engineering of the stick has been give
some consideration. It is about 10 inches long so as to move a
amount that is appropriate for human arm motion. The friction
and the stiction of the motion is carefully controlled by
adjustable teflon bearings so that it is easy to move the stick
but in addition so that the stick will stay in position if it i
released.

Finally, a set of knobs send ten signals to the computer. Thes
are used to set parameters in the computer program and to
individually control the loudness in the synthesizer channels.

## Software Design

The software is developed from a control philosophy proposed by
Miller Puckette (Puckette 1983). RTLET is based on passing
messages, or letter, between processes. RTLET consists of two
logically distinct sections: the letter delivery section and th
control processes. The letter delivery system passes the lette
to the control processes which perform some service. The
routine that controls passing of letters is the post office.
The actual delivery of the letters is performed by the postman.
Control processes are logically distinct from the post office
because letter delivery is a simple message passing system and
application independent. The post office could work well with
variety of real-time control applications.

The operation of the post office may be seen in figure 2. A
letter is delivered to the post office by some control process.
An initial letter, delivered by the initialization routine,
starts the process. The posted letter is placed in the letter b
in order sorted by the delivery time stamped on the letter. Th
postman reads the destination and delivery-time of the top lett
in the bin. When real-time equals delivery-time it send the
letter to the control process, and transfers control to the
control process. The control process runs to completion and
returns control to the postman. The postman reads the delivery
time of the next letter in the bin and waits for that delivery-
time. Delivery-times are specified in beats, an arbitrary unit
whose time may be adjusted to vary the tempo of the musical
piece. One of the timers in the computer is used as a real-tim
clock. An interrupt handler keeps a variable, t_time, equal to
the current time value in milliseconds. Various tempo algorith
are used to relate t_time and beats. Control processes may sen
letters to other control processes or to themselves to
continuously cycle the delivery process.

Control processes are written specifically to play music. A
control process reads the notes from an intermediate score. The
intermediate score is compiled from an alphanumeric score which
can be prepared on any word processor. The score gives the

pitches and durations in beats of each note to be played and in addition it gives the times in beats between all baton strokes.

The intermediate score consist of a list of operation codes each followed by information appropriate to that particular cod The main operation codes are listed in figure 3.

Before starting to play, the intermediate score is loaded into array, score[], that is read and parsed by the control process. Playing a note is accomplished as shown in figure 4. A pointer into score[] is maintained by control process CP1. CP1 reads t operation code and, based on the type of operation, the control process performs the operation. In the case that the operation code is PLAYN, five pieces of information are then read from th score array. The first is the midi channel to play on. The mi standard allows up to 16 devices to be addressed on the bus at any given time. The second item needed is whether or not the note is part of a chord. This is used to bypass the timing aspects of RTLET when certain notes should happen simultaneousl The control process keeps reading from the score array and playing the notes until all notes of a chord are played. The third item is the duration of the note, specified in terms of a legato factor. This factor allows a certain amount of control over whether a note is played staccato or legato. This variabl can have three values: one corresponds to staccato, in which th note is played for 50ms, another corresponds to normal, in whic the note is on until the start of the next note, and the final is for legato, in which the note is on 50ms past the start of t next note. The manner of note playing can also be written out in the score by separating the note into a shorter note followe by a rest. The fourth piece of information is the key number o the note to be played, which determines the frequency. The mid standard allows for 128 possible pitch values with middle C as value 60. The final item is the delay, in beats, before the ne note starts.

The final thing that CP1 does before returning to the postman i to send a letter to itself with a delivery time equal to the current beat plus the delay. The assures that the process will continue. When a termination opcode is encountered, the final letter is not sent, and post office terminates.

Other opcodes are used to control the execution and flow of the interpreter. For instance, multitrack sequencing may be done i RTLET using the START opcode. The START opcode forces two lett to be sent from CP1. The first letter is the letter that is required to make CP1 continue to interpret the score array for the track containing the START opcode. The second letter start CP1 again, but at a different location in the score file, thus starting a second track.

Multitrack sequencing is essential in RTLET. Consider the exce of the Kreutzer Sonata on figure 5. The beginning of the sixth measure contains 2 quarter notes, 2 dotted quarter notes, and a half note, all starting simultaneously. RTLET cannot play thes as a chord in one track, since all notes in a chord must both

begin and end together. The solution, as given in figure 6, is to split the piano part into three tracks, V3, V4, and V5. There is no limit to the number of tracks in RTLET.

Control Processes as Concurrent, Re-entrant, Pure Procedures

Although control processes can be written in a variety of ways, it is often useful to write them as pure procedures. The opcod interpreter procedure CP1 which we have described is such a procedure. The score[] array which is read by CP1, and which contains a description of the music to be played, can be looked upon as a program which is interpreted by CP1. Each time CP1 i activated by the postman, another opcode in score[] is interpreted and executed.

Only one copy of CP1 exists, even though from the user's standpoint as many "copies" of CP1 as desired may be "simultaneously" playing notes. At any given moment in real-time, the pseudo-copies of CP1 can be executing opcodes at the same or different places in score[]. Commands exist for starting new copies and for terminating copies. Opcodes also exist for both absolute and conditional jumps of the CP1 execution to other locations in score[].

In order to be useful, pure procedures must be able to store local variables somewhere while they are not being used. In languages such as C, local variables are pushed onto a single stack. Such storage requires that the procedures be called and return in a hierarchical order and does not work with the RTLET control processes which can be called in very arbitrary orders the postman according to the delivery times on letters.

The solution to the storage problem used in RTLET is to pass local variables to a procedure via the letter that activates th procedure. The principal use of a local variable in CP1 is to store the location in score[] of the next opcode to be interpreted. Obviously, other local variables can be handled i the same way. For example, if one wanted to set up a loop in score[] and to execute a block of opcodes n times, the counter for the loop could be set up as a local variable.

Another application of local variables is in passing channel dependent information along. For instance, if a user wants to play 2 tracks simultaneously at different intensities, the simplest way to handle this is to set the intensities of each track and pass this information as part of the letter. Each channel is set differently, so each is propagated at a differen value.

Although we have not attempted to do so, we believe the letter principal would be a good way to handle local variables in a general purpose real-time compiler.

Servicing of Real-Time Devices

In addition to sending letters to control processes, the postma services any real-time devices that are being used. At present these consist of the daton, the joy stick, the knobs, the computer keyboard, and the synthesizer. Servicing is done by a wait loop in postm(). The wait loop also looks at the milliseco clock (t_time) and updates the current beat as appropriate.

All waiting in RTLET is done in the postm() wait loop and all other procedures return to this loop as rapidly as possible. N real time devices may be added to the program by rewriting the loop and thus can be guaranteed equal priority in service.

The input devices all put signals into a 16 variable array via the dma input; each millisecond new samples are read into the array. The wait loop examines this array and acts accordingly. For example, for the drum, when the total force signal exceeds threshold, the program watches the other drum signals, waits un they have reached their peak values, computes x and y, and sets flag indicating that the drum has been hit. Signals from the jo stick are usually transmitted to the mod wheel and footpedal controls of the synthesizer. The wait loop watches the joy sti signals and sends appropriate midi signals to the synthesizer only when the joy stick has been moved.

Postm() also reads t_time when a drum stroke occurs and compute the implied tempo, btim, as the time difference between the current stroke and the previous stroke divided by the number of beats written into the score between the strokes. Actually bti is the inverse tempo and is the number of milliseconds per beat

In addition to tempo, some special cases in time control must b treated. The expected beat for the next daton stroke is writte in score[] (for example as V6 on figure 6) and the notes played depend on whether the computer arrives at the next expected bat beat before or after the daton has been hit. If the computer arrives at the beat first, the program halts execution and wait until the daton has been hit before playing the next note. If t daton stroke occurs before the computer halts, the computer jum immediately to the note in the score which immediately follows the baton stroke; intervening notes are not played.

This algorithm, which involves stopping and skipping notes in t score, is not the only way that tempo can be regulated. It causes the computer to follow the conductor very closely, and i preferred by most conductors who tried the program. However, i also demands that the conductor beat very evenly in sections in which he wishes to maintain a constant tempo. One can design other algorithms which smooth the tempo variations produced by the daton.

## A Simple Example of a Conductor Program Score

The use of the conductor program is best understood by discussi a simple example. Figure 5 shows a short score written in the appropriate form to be put into the conductor program. The sco

is entirely alphanumeric and can be written with any word processor. The equivalent score in normal musical notation is also shown.

The first two lines of the score set the key signature to two sharps and set some other initial parameters in the program whi we will not discuss.

The next two lines of the score tell the computer that there wi be two voices in the score, the first being the baton and the second being an instrumental voice. For example, the line "v2 h0 t3" says that the second voice will be played in the 6th octave on synthesizer channel 0 with timbre number 3.

The fifth and sixth lines of the score specify the baton stroke and the notes to be played. The baton has a line in the score like any other voice in which s slash, "/", indicates where bat strokes are to be made. In line 6, the pitches of the notes ar written simply as the letter names of the notes AB...Gab...g ov a two octave range where capital letters are used for the lower octave. The actual octave in which the notes are played is als determined by the octave constant o6, which can be changed at a time. The letter r indicates a rest.

Durations in the score are written as dots. In this case the value of a dot is arbitrarily chosen to be an eighth note. Thu the first note, is written d.. and is a quarter note. The note DFad cause a chord to be played because there are no dots betwe the letters.

Baton stokes, indicated in the fifth line are also separated wi dots to locate their position in the score. A stroke can serve several musical purposes. It can start a note or start a group of notes or start a chord or start a rest which is equivalent t stopping a note. Computationally, each baton stroke after the first stoke sets the tempo which will be used until the next stroke. The tempo computation is shown on figure 5 and has already been discussed The tempo can be changed at each stroke, thus the conductor can accelerate or retard the music. The time of the strokes are measured to an accuracy of 1 millisecond by clock in the computer. This accuracy is more than sufficient f human gestures.

We will now discuss playing the score stroke-by-stroke. The first stroke is an upbeat which produces no sound (it plays a rest) and together with the second stroke, sets the initial tempo.

The second and third strokes play the the first two d's in the score. These are legato and each note lasts until the next not starts.

The next stroke starts a staccato d which last's only half of it quarter note duration. The next stroke starts an f with a fermata which will sustain till it is cutoff by the following stroke.

The next two strokes begin and end an a which is moved up an octave by the accidental mark, " ^ ". The second of these strok starts the playing of three eighth notes ^a.g.e. whose tempo is determined by the time between the two strokes.

The next three strokes play three more eighth notes e.d.d. whos are individually determined by the times of these strokes.

The next to last stroke starts a chord, DFad, which continues till it is cutoff by the final stroke.

This score illustrates most of the basic functions of the baton The function of a particular stroke is determined by where it appears in the score in relation to the notes in the playing voices. Depending on its position, the stroke can start a note or stop a note or start a group of notes. As far as the comput program is concerned, all strokes are treated in the same way. Each stroke specifies a tempo and starts the computer playing t group of notes at the specified tempo. It may be musically usef in many case to have only one note in the group so that the stroke is a trigger for this individual note. But this is only special case of the general baton function of tempo setting anc triggering a group of notes.

## The Kreutzer Sonata--a More Realistic Example

Figures 6 and 7 respectively show the traditional and the alphanumeric score for a short section from the beginning of Bethoven's Kreutzer sonata. Six tracks V1 through V6 are defin by lines 3 through 8 in this score. All of these tracks will b interpreted by CP1 "simultaneously" and in parallel. Tracks 1 a 2 are the violin part, tracks 3 through 5 the piano part, and track 6 the daton.

Track six is of particular significance because it it the baton track. Only baton strokes are present here. In the first bloc of tracks one through six, only the baton stroke is executed. The other tracks all have rests. This is equivalent to the conductor's upbeat. The time between the first and second batc strokes is used by RTLET to set the initial tempo of the piece. The first group of four dots (up to the next baton stroke) is played at this tempo. At the next baton stroke, the program recalculates the tempo.

The first played notes are in the second block of tracks one through six. The first track is the only one playing here. It starts with a chord, "AEc^a" which has a quarter note duration (four dots). The chord is legato and will continue until the next baton stroke starts the next chord.

At the end of the first measure, we have an example of a slight staccato chord. The score is:

```
1  ...bd...r.
```

```
2 r...
3 r...
4 r...
5 r...
6  .../... ./
```

The chord "bd" terminates to a rest before the baton stroke is written. A more staccato chord could be written by moving the rest to the left.

Another function of the baton can be to cutoff a fermata with a separate stroke. This is used in measure 4 where the final "c" plus the chord "AE" are allowed to continue as long as desired the performer according to the score:

```
1  d....  c.. r.. ....
2 AE....   .. r.. ....
3  r....   ..  .. ....
4  r....   ..  .. ....
5  r....   ..  .. ....
6  /....  /.. /.. ....
```

Methods of Putting Scores into the Conductor Program

If the conductor program becomes popular, then there will be a need to put much music into a form which can be conducted. The alphanumeric score shown on figure 7 can be prepared with any word processor. Although this is an entirely practical way of working, faster and more accurate procedures can probably be developed.

As a masters thesis at MIT Ruth Shyu (Shyu 1988) developed a simple program for preparing scores in which the notes are displayed in a simplified common practice music notation drawn the computer on its terminal. The display provides a fast accurate way to proof read the music. The program was designed to minimize the number of keystrokes required to put in the music.

Leland Smith (Smith, 1977-1987) has developed an excellent program for high quality music printing in which the music is written into the computer on a standard asci keyboard using a special language which he has invented. Parts of his program could be adapted to prepare scores for the conductor program. Smith's program also displays a good quality common practice music notation on the computer terminal and these displays are helpful for proofreading.

Another possibility is to "play in" the score on a midi keyboar The midi interface on the computer can be used either for input or output so that no additional hardware would be needed to implement this option. The procedure would be to play in eithe some or all the voices of a score and then to edit the computer

record to add conducting marks, to correct mistakes, and to properly quantize the note durations. With midi, the pitches a automatically quantized by the keyboard, but durations are almc continuous variables. Hence, they would have to be fitted intc some appropriate set of standard note lengths. Although making useful and practical system to "play in" a score is undoubtedly more difficult than it appears to be, it can probably be done. It is an interesting question for research in programming and human engineering.

A final possibility would be a music scanner that would read nc printed music and automatically convert it to machine readable form. We believe this is a very difficult task and may be the last of the methods we have discussed to be usefully achieved.

Uses of the Conductor Program

Just as for any other new musical medium, the conductor program can be used by contemporary composers to perform new compositic (Rocco 1985, Radunskaya 1986, Boulanger 1987, Chafe 1988) especially written for this program. However, in addition to i use in new music, the program may have important uses playing traditional music. Two possibilities which seem promising and which may be economically important are accompaniment and activ listening.

Accompaniment of soloists, which is often done with a piano, is both difficult and uninspiring for the accompanist. Very good reading and ensemble abilities are required from the pianist. seldom gets a big share of the glory for a good performance. Consequently, it is both difficult and expensive to obtain accompanists. An alternative is to put the accompaniment intc the conductor program and to conduct the accompaniment with the daton. Much less reading skill is required by the accompanist and he can focus his attention on ensemble. Also, certain soloists such as singers who have their hands free can accompar themselves. We have done demonstrations with a soprano (Maurer Chowning 1987) showing that this is indeed a promising method.

Active listening is a way of experiencing music which is made possible by the conducting program. Instead of listening to recorded music, the music appreciator would purchase a computer score and conduct his own interpretation of the piece. In addition to allowing an individual style of interpretation, the listener could take the piece apart to focus on individual voic or sections of the music in ways that are not possible with normal recordings and thus quickly gain a deeper understanding the music. We believe this may become a very popular way of appreciating music.

Active listening could also lead to an entire new music industry--that of preparing scores for the conductor program.

References

Boulanger, Richard, 1986, Shaddows, a piece for the conductor program and electronic violin, premiered MIT Nov 1986

Chafe, Chris, 1987, Virga, a piece for the conductor program and harp, premiered Stanford April 1987

Chowning, John, 1988, The name "active listening" was suggested by John Chowning.

Chowning, Maureen, 1987, soprano, Mozart recitative and aria, Deh, vieni, non tardar, from the Marriage of Figaro, accompanie by the conductor program

Data Translation Co, 100 Locke Dr. Marlboro, Massachusetts

Puckette, Miller, 1985, informal discussions durning the summer

Radunskaya, Ami, 1986, Fruit Salid on Pi, a piece for the conductor program, three cellos, and a bass mandolin, premiered at Stanford, October 1986

Rocco, Robert, 1985, AT&T Bell Labs, Murray Hill, New Jersey, many short pieces and demonstrations composed for the conductor program.

Smith, Leland, 1977-1987, Score (tm), Computer Music Typograph System, Passport Designs, Inc

Shyu, Ruth, 1988, MIT masters thesis in computer science, The Maestro Program

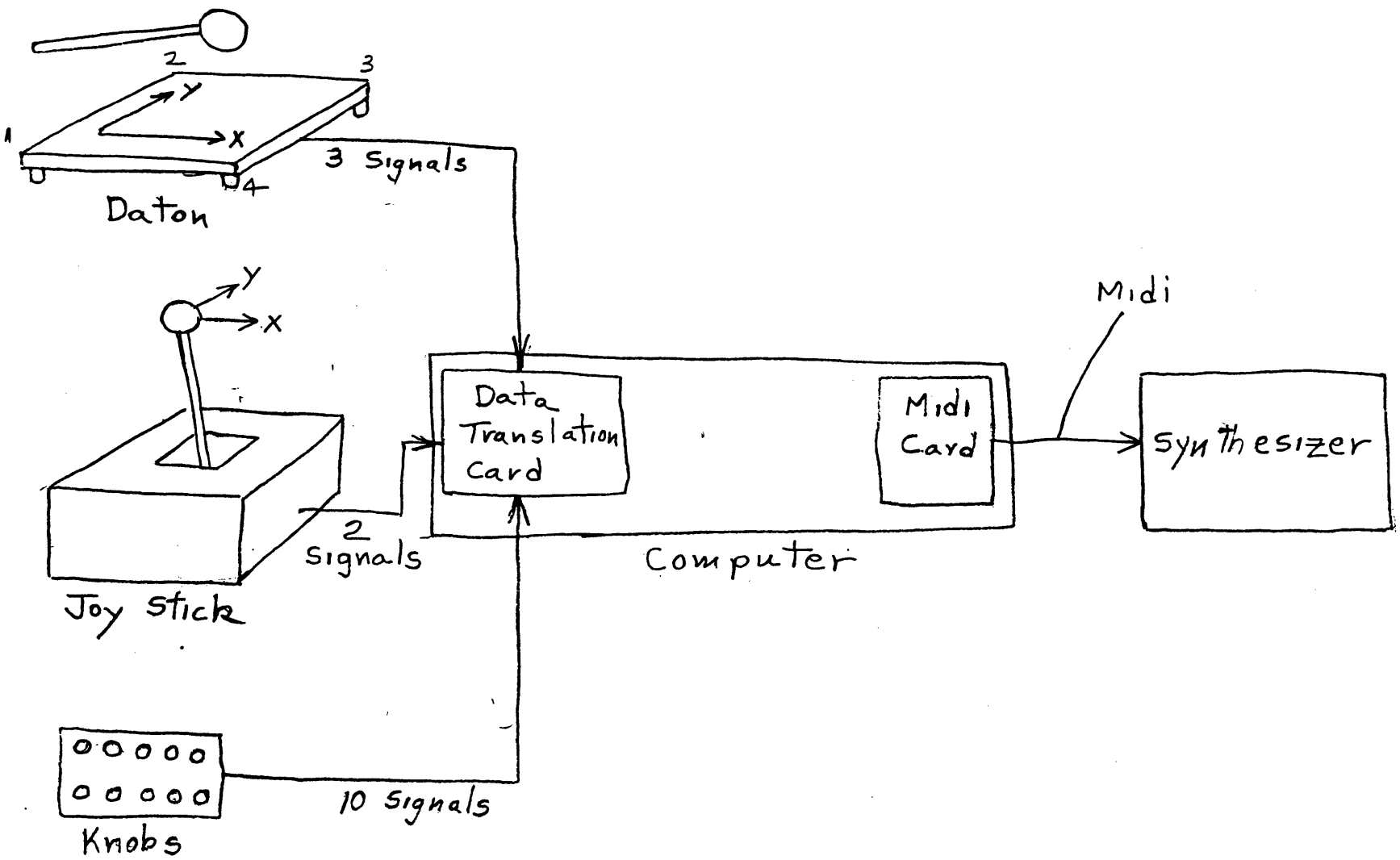



Fig 1 Diagram of Conductor Program Hardware

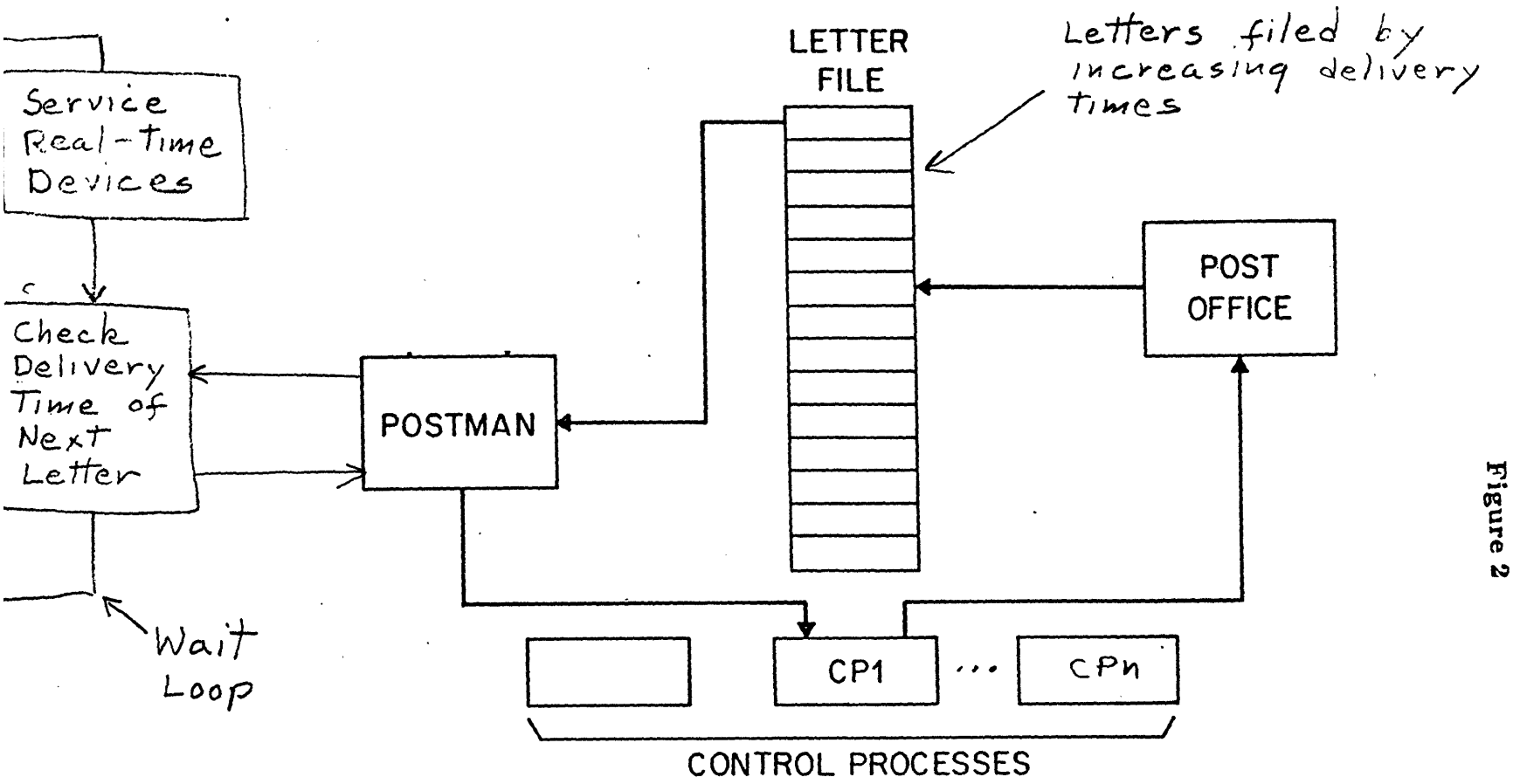



Figure 2

LETTERS:

DESTINATION
DELIVERY TIME
CONTROL PARAMETER 1

| mneumonic | |
|---|---|
| **PLAY OPERATIONS** | |
| PLAYN: | play a note with beat ti |
| **WAIT OPERATIONS:** | |
| WAITR: | wait for x beats |
| **MIDI DEVICE CONTROL:** | |
| TIMBRE: | change timbre immediate |
| **REAL TIME CONTROL:** | |
| BAT: | baton stroke |
| **TEST OPERATIONS:** | |
| COMPA: | compare register and val |
| COMPR: | compare two registers |
| **BRANCHING INSTRUCTIONS:** | |
| JMP: | unconditional branch |
| START: | start parsing at locatio (fork) |
| SUB: | jump to sequence at loca |
| **REGISTER CONTROL:** | |
| SETR: | set register to a value |
| INCR: | increment a register |
| DECR: | decrement a register |
| **TERMINATION CONTROL:** | |
| TERM: | sequence termination |
| TERMA: | master termination |

Fig 3 RTLET Operation Codes

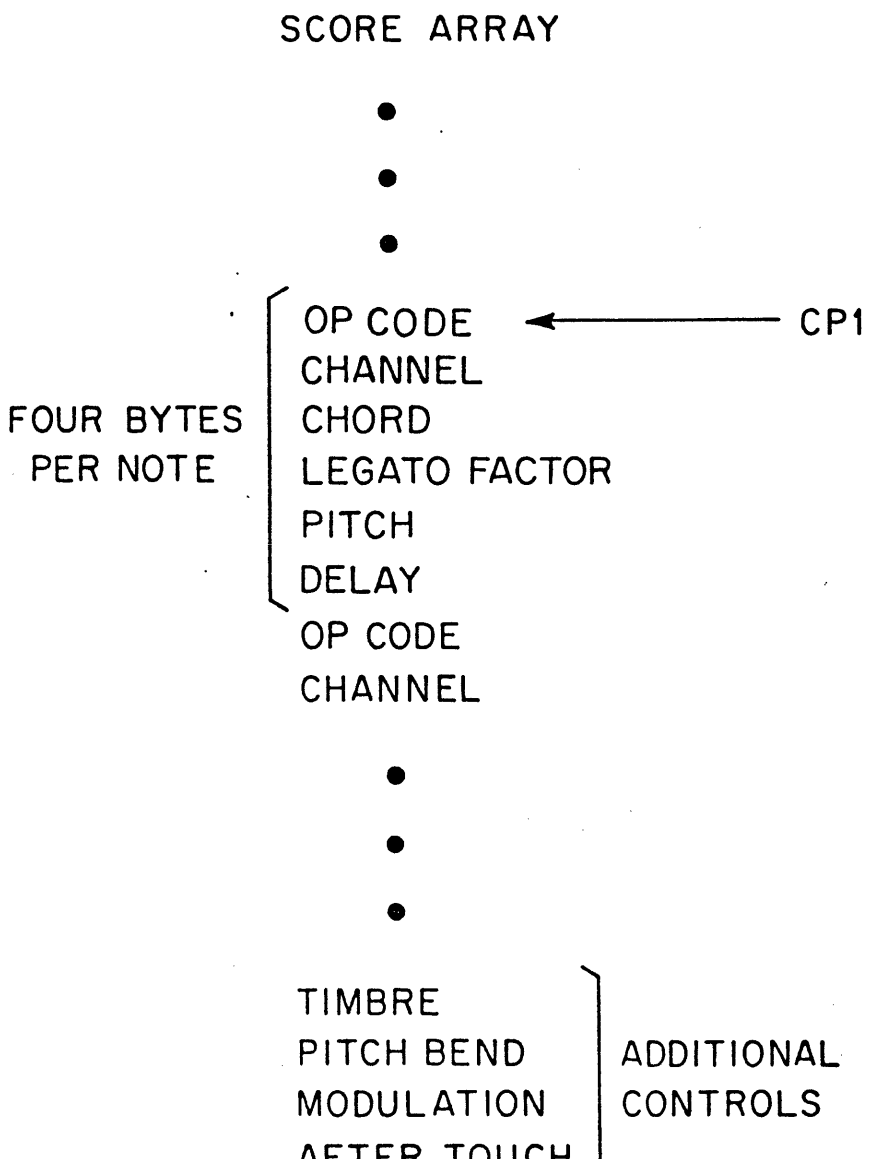
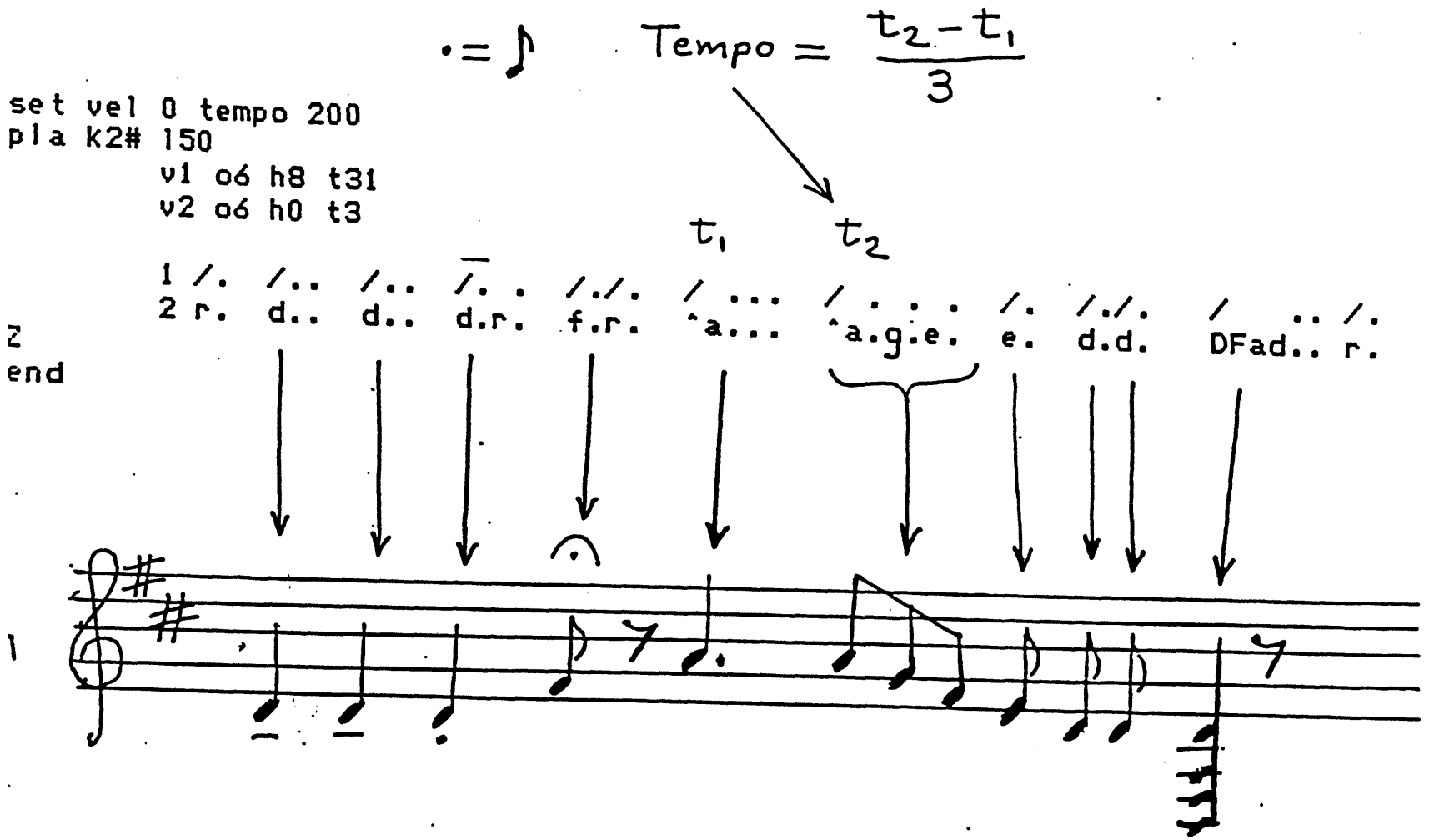

SCORE ARRAY
OP CODE
CP1
CHANNEL
FOUR BYTES
PER NOTE
CHORD
LEGATO FACTOR
PITCH
DELAY
OP CODE
CHANNEL
TIMBRE
PITCH BEND
ADDITIONAL
MODULATION
CONTROLS

Figure 4


• = ♪
Tempo = $\frac{t_2 - t_1}{3}$
set vel 0 tempo 200
pla k2# 150
v1 o6 h8 t31
v2 o6 h0 t3
$t_1$
$t_2$
1 /. /.. /.. /. . /.// / ... / . . . /. /.// / .. /.
2 r. d.. d.. d.r. f.r. a... a.g.e. e. d.d. DFad.. r.
2
end
1

# SONATE

(komponiert 1803)

Op.
(Kreutzer

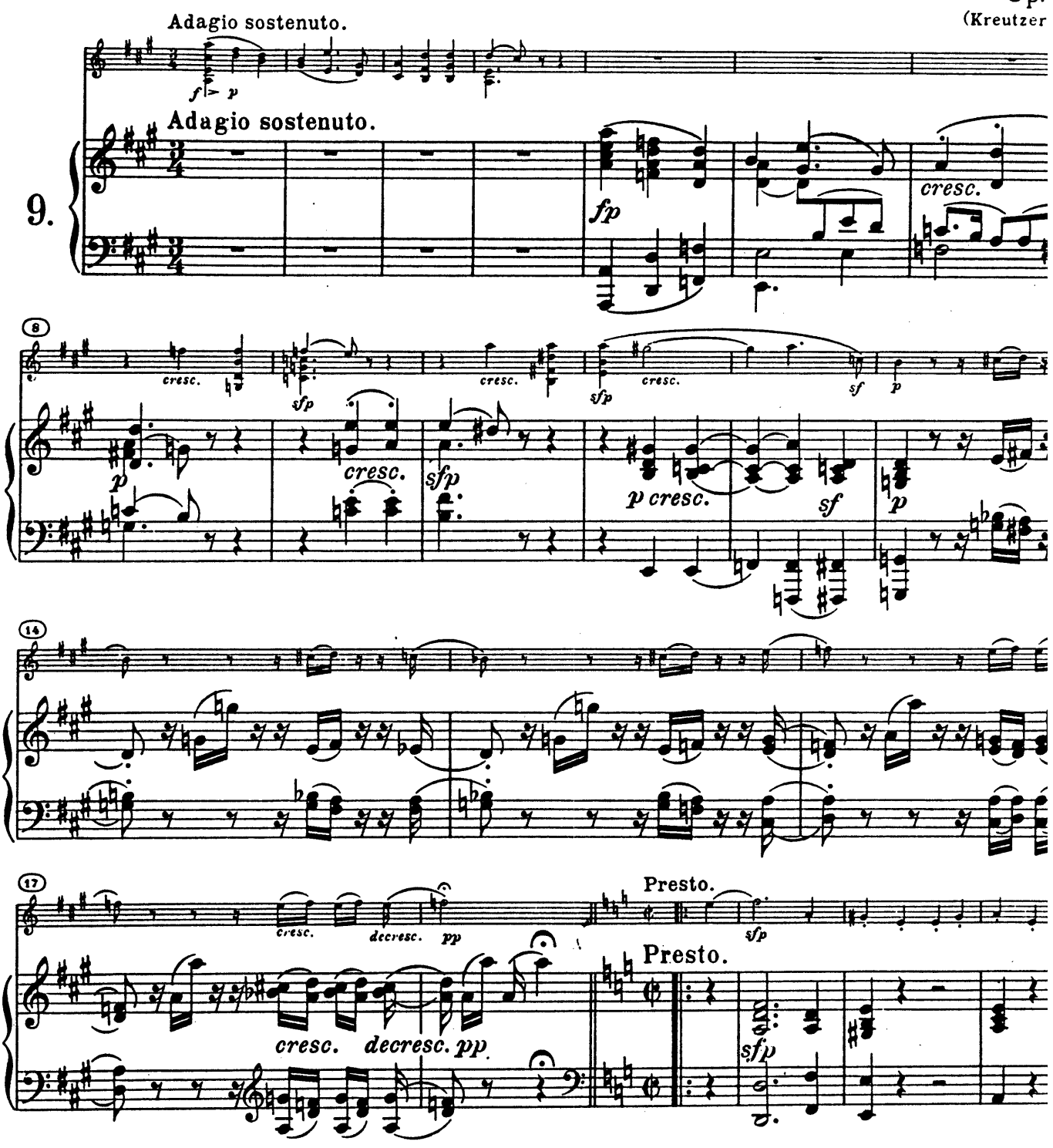



Fig 6 Kreutzer Sonata in Normal Music Notation

## Figure 6

```
set vel 50 tempo 200
pla k3# l50
        v1 o7 h0 t9
        v2 o7 h1 t9
        v3 o6 h2 t0
        v4 o4 h3 t0
        v5 o4 h4 t0
        v6 o4 h5 t0

1 r....
2 r....
3 r....
4 r....
5 r....
6 /....

1 AEc^a.... df.... bd...r.          Gb.... Ee.... ..DG.r.
2 .... .... ....                    .... .... ....
3 .... .... ....                    .... .... ....
4 .... .... ....                    .... .... ....
5 .... .... ....                    .... .... ....
6 /.... /.... /....                 /.... /.... ../..

1 Ca...r. BFd...r. BGd...r.                 d.... c..r.. ....
2 r.... .... ....                   AE.... ..r.. ....
3 r.... .... ....                   .... .... ....
4 r.... .... ....                   .... .... ....
5 r.... .... ....                   .... .... ....
6 /.... /.... /....                 /.... /../.. ....

1 r.... .... ....                   .... .... ....
2 r.... .... ....                   .... .... ....
3 ace^a.... $Fad$f.... Dad...r.             ab...r. Ge.... ..G.r.
4 o4 Aa.... Dd.... $F$f...r.                o5 d!E.... .r.b.. e..d.r.
5 .... .... ....                    e.... ...r. e...r.
6 /.... /.... /....                 /.... /...../..

1 r.... .... ....
2 r.... .... ....
3 a...r.Dd...r.Dd...r.
4 $c...b. a.r.a.. b..$c.r.
5 $f.... ...r. f...r.
6 /.... /.... /....

1 .... $f...r. !$GDb$f...r.         $C$G$c.... ..r.. ....
2 r.... .... ....                   $f.... e..r.. ....
3 Fa$C.... B$G..r.. ....            .... $Ge...r. ae...r.
4 o6 Dd$G.... ..r.. ....            o5 .... $ce...r. $ce...r.
5 r.... .... ....                   .... .... ....
6 /.... /.... /....                 /.... /.... /....

1 .... ^a...r. BF# d^a...r.                 Eb^a.... g.... ....
```

Fig 7 Kreutzer Sonata Alphanumeric No

```
2 r.... .... ....                    .... .... ....
3 e.... # d..r.. ....                          .... BDG...r. B$CG....
4 o5 bf^ a.... ..r.. ....             .... o3 e...r. e....
5 r.... .... ....                    .... .... ....
6 /.... /.... /....                  /.... /.... /....

Z
end
```