CONDUCTOR PROGRAM INSTRUCTION MANUAL

MAX MATHEWS

MARMAX
225 PRECITA AVE
SAN FRANCISCO CA 94110
415-821-4661
max.mathews@gmail.com

DEC 2007

PRELIMINARY EDITION

INTRODUCTION

One mode of using the radio-baton is the CONDUCTOR mode. This is a form of sequencer program in which a score is played by the baton as a sequence of predetermined notes. The conductor program differs from most other sequencers by providing the performer with a greater degree of expressive control. Triggers from the baton and trigger points written into the score provide control over timing of events in thescore. The XYZ positions of the sticks can be used to control overall dynamics, balance between voices, and timbre factors. The concept which led to the program is the performer as an orchestra conductor and the baton and conductor program providing him with a simulated orchestra.

The score of the piece to be played is on a file on the computer's disk. The score contains all the information necessary to conduct the music including the notes to be played, midi program changes to configure the synthesizer, the trigger points in the score, and the coupling of the xyz movements of the batons to control changes to be sent to the synthesizer.

The performer starts to play by hitting the baton with stick 1. This produces the first trigger. The baton processor sends noteon midi commands to the synthesizer for whatever notes are to be played by the first trigger. This general process is repeated for all the subsequent triggers.

MIDI CONNECTIONS OF CONDUCTOR PROGRAM, RADIO-BATON, AND SYNTHESIZER

The conductor program is written to work with the radio-baton and with a synthesizer. Only two midi connections are required, one from the radio-baton out to a midi input on the computer, a second from a midi output on the computer to the synthesizer midi input. The user must supply the computer midi device and device drivers–these ARE NOT included with the radio-baton or cond program.

The cond program will help choose the computer midi device numbers, provided appropriate device drivers have been installed in the computer. In addition to the radio-baton output and the synthesizer input, the cond program will also ask for three other midi connections which are not presently used. These are the radio-baton input and “other computer” input and output. They should all be given numbers -1 to indicate they are not used.

To avoid having to select device driver numbers each time the cond program is executed, a file midi.def can be added to the directory in which the conductor program is executed. This file contains the 5 device driver numbers requested by the cond program. A typical file would be:

```
baton  -1  synth  -1  -1
```

where baton and synth are the numbers of the drivers for the radio-baton output and synth input.

CONDUCTOR PROGRAM SCORE

The conductor program score can best be described with an example. Figure 1a shows the common music notation score and Figure 1b shows the conductor program notation score for a fragment of a violin-piano sonata.

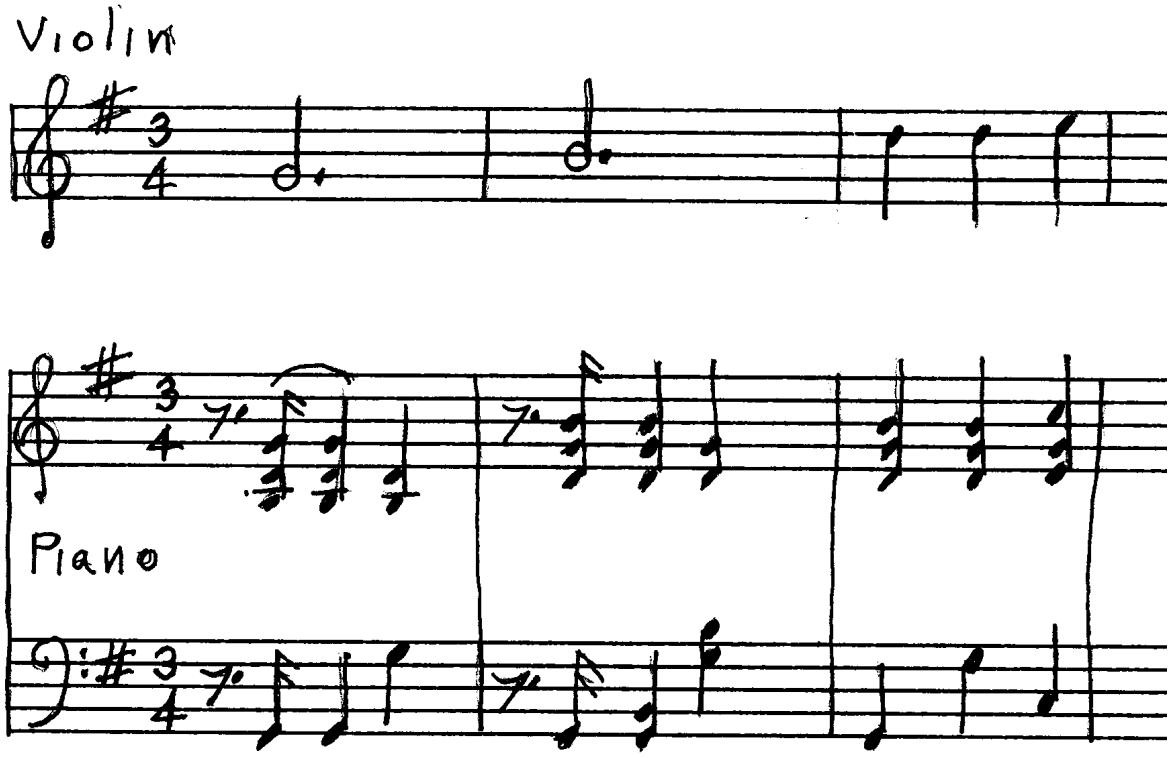

FIGURE 1a--SCORE IN COMMON MUSIC NOTATION

```
*-------------CONDUCTOR PROGRAM SCORE SETUP-----------------*
  I   K1#   v220                        *initialization & key & default tempo*
  K1#
  v220
  1 o60 h1 t40 kv100              *violin keyvelocity at constant value 100*
  2 o60 h2 t0  kx1               *piano rt hand x1->loudness via keyvelocity*
  3 o48 h3 t0  ky1               *piano lft hand y1->loudness via keyvelocity*
  q1 h1 y2 c7                   *continuous control of violin loudness with y2*
  q2 h1 x2 c1                    *continuous control of violin vibrato with x2*

  *-------------CONDUCTOR PROGRAM SCORE NOTES-----------------*
  *1* 0  /.... /.... /....                    *3 evenly spaced trigs 1st measure*
     1  G....  .... ...r.
     2  r...BDG...r.BD...r.                    *chords-- no dots between notes*
     3  r...!G,,,,,r,,!G..r.G...r.                          * 8 commas == 1 dot *
  *2* 0  ..././..../....                     *3 triggers playing 3 piano chords*
     1  b.... .... ...r.
     2  r...DGb,,,,,,r,,DGb...r.DG...r.
     3  r...!G,,,,,,r,,!GB...r.Gb...r.
  *3* 0  /.... /.... /....
     1  d...r.d....e...r.                              *2nd & 3rd notes slured*
     2  DGB..r..DGB..r..EGc..r..                          *3 stacato chords*
     3  !G...r.G..r..C..r..
     0                          *final 0 to play last measure & terminate piece*
```

FIGURE 1b--CONDUCTOR PROGRAM SCORE WITH HEAVY ANOTATION

The conductor program score is divided into two sections--setup and notes. Comments can be included in the score enclosed by asterisks. In the first line of the setup section,

```
I   K1#   v220
```

I initializes the baton processor, K1# sets the key signature, and v220 sets a default "inverse" tempo--ie, the larger the v number, the slower the tempo.

The second, third, and fourth lines initialize the three voices, 1,2, and 3. For voice 1 the:

```
1  o60  h1  t40  kv100
```

the o60 sets the pitch transposition so a "C" in the score means middle C. Middle C has keyno 60 in most synthesizers. The "o" setting can transpose a voice up or down by any number of half

steps. The h1 says this voice will be played on midi channel 1, the t40 sends a program change 40 on midi channel 1 to select the general midi timbre 40 which is a "violin" timbre, and the kv100 sets the key velocity of all notes played in voice 1 to a constant value of 100.

The setup for voices 2 and 3 is similar except that the key velocity for voice 2, the piano right hand is controlled by the x position of stick 1 when it makes a trigger. The key velocity for the piano left hand is controlled by the y position of stick 1. Thus we have independent control over the loudness of the left and right hands. By the nature of key velocity in synthesizers, this control can only be set at the beginning of each note, but such control is appropriate for a piano.

The last two lines in the setup use "patch cords" q1 and q2 to connect x2 and y2 to synthesizer control changes. There are 16 patch cords, q0--q15, in the nc program. They can only be used for one function at a time. The statement:

```
q1  h1  y2  c7
```

uses patch cord q1 to connect the y position of stick2 to control change 7 on midi channel 1. In this way, the loudness of the violin is continuously controlled by the y position of stick 2. Similarly the x position of stick 2 control is assumed to control the vibrato strength. The control of loudness with c7 is standard in most synthesizers. c11 can also be used for this purpose. Control of vibrato is not standard and usually the specific desired control must be programmed into the synthesizer.

In the notes section of the score, pitches are designated by the letter names of the notes. Upper and lower case letters are used to span a two octave range. Although the "o" transposition control can associate any pitch with a given letter, o60 is usually used to write treble clef passages and o48 for bass clef passages. For these setting, the pitch- letter equivalences are as shown on Figure 2.

The key signature and five accidentals modify the letter pitches. The accidentals are #-sharp, @-flat, $-natural, ^up-one octave, and !-down one octave. Accidentals precede the pitch letter. Several-as many as desired--accidentals can be applied to a pitch letter. Accidentals apply only to one note--they do not carry to the end of the measure.

Durations are designated by the dots and commas between the pitch letters. The time between two note, in baton score time units, is the number of dots + 1/8 the number of commas between the pitch letters for the notes. Thus a dot by default equals 8 commas. The equivalence between dots and commas can be set to other values. (See the complete list of score symbols in the appendix.)

The relation between dots and note values is arbitrary and can be selected by the composer. In the example we have chosen 4 dots to equal one quarter note.

In the rest of this text, we will usually talk about dots as the time specifier, but in all cases we mean dots and commas.

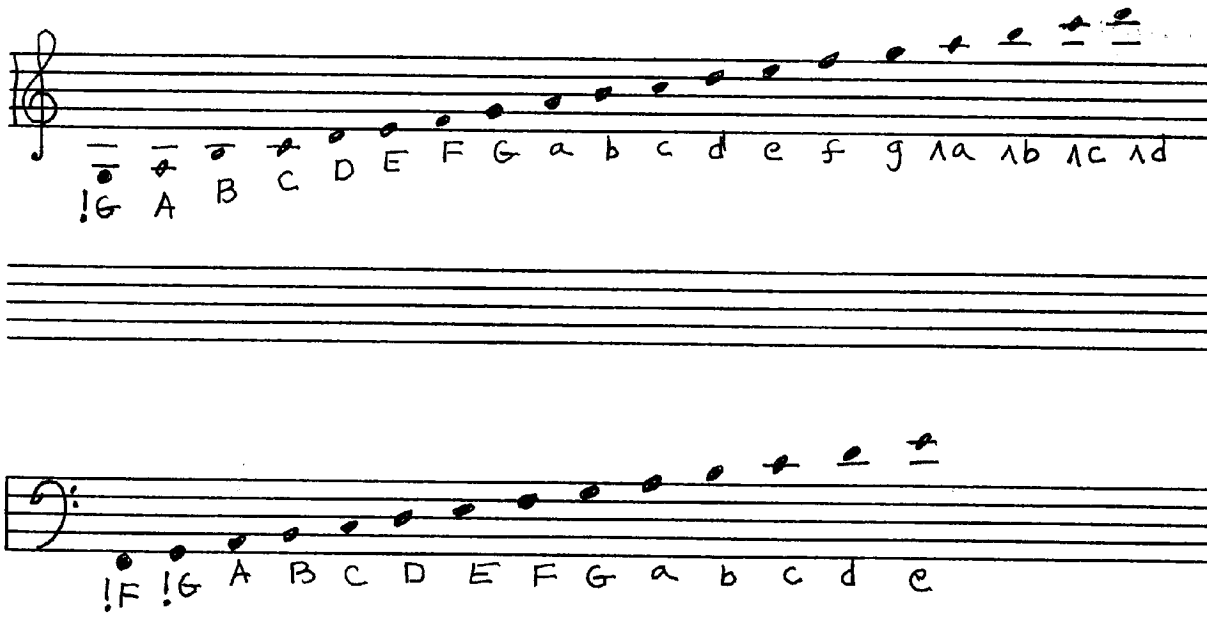

FIGURE 2--USUAL NOTATION FOR WRITING
TREBLE AND BASS CLEF NOTES

Each measure has 4 voices. Voice 0 is always the baton triggers from stick 1 In this voice a "/" indicates the location of a trigger. Voices 2,3, and 4 are as defined in the setup.

The duration of each measure is dominated by the number of dots in voice 0. If some other voice has more dots than voice zero, an ERROR comment will occur. If some other voice has fewer dots than voice zero, additional dots will be automatically added to the end of this voice .

Simultaneity between the events in different voices (including the baton voice) is specified entirely by dots. Thus, any notes that are written following the same number of dots after the beginning of the measure will be played simultaneously. In a given voice, pitches with no intervening dots and commas will be played as a chord.

Notes in a given voice are sustained until a new note is written or until a rest "r" is written. Thus to play staccato notes, rests must be used.

All notes of a chord in a given voice end at the same time. If it is desired to have the notes of a chord end at different times, the +/- notation described in the appendix can be used.

TEMPO CONTROL

Four methods of tempo control are available. The primary control is the trigger marks ("/") and dots in voice 0. After the first trigger, the performance starts at the default tempo (v220 in the example) specified in the setup. For each subsequent trigger, the computer recomputes the tempo by dividing the time (in milliseconds) taken by the performer between the last two triggers by the number of dots in the score between the corresponding two trigger marks. The section FOLLOWING the second trigger mark is played at the new tempo. Thus the performer automatically resets the tempo with each trigger. However, the tempo always lags one trigger mark behind the performer's actions. Thus the tempo between any two trigger marks is determined by the time he took one mark previously. If this lag causes problems in a particular section, the usual solution is to add more trigger marks thus giving finer tempo control at the expense of more gestures on the part of the performer.

If the performer changes tempo rapidly, an irregular performance may result. If the performer slows the tempo the baton will reach the next trigger mark before the performer gives the next trigger. In this case (unlike a real orchestra) the baton will simply stop playing and wait for the performer to make the next trigger. If the performer speeds up and makes the next trigger before the baton has played all the notes leading up to this trigger, the baton will jump to the next trigger point, omitting any unplayed notes.

The second method of tempo control is to write default tempos into the score. A statement, vn, where n is any integer, can be put just ahead of any trigger mark. At that trigger mark the baton with start playing at the default temp. After subsequent triggers it will recompute the tempo using the primary method of tempo control.

The third method of tempo control is to designate a section of the score by writing two w's in the baton voice. In this section, trigger marks will be ignored and the tempo will be continuously modified by y1. This method is useful to make smooth accelerando and retards.

The last method of tempo control is to designate a section of the score by writing two W's in the baton voice. In this section, trigger marks will be ignored and the tempo will be constant at whatever value it had at the beginning of the section.

THE COMPILE PROGRAM

The raw score must be compiled before it can be performed. This is done from a command line with the command:

compile score

where score is the name of the score file. The score can either be a conductor program score as described in the preceding section, or a conductor program score (sometimes called a "header" file) plus a midi score.

The compile program will generate a file score.p which has an extention .p which is the actual and only file read by the cond program for a performance.

THE COND PROGRAM

The cond program in the computer manages score and voicing files and downloads information in the baton processor for performance. The program is executed from a command line with the command

cond score

where score is the name of the compiled score file (without the .p extension). New scores can also be selected inside the cond program with the "s" command.

When the program is executed, the command menu shown in Figure 3 appears on the computer screen.

```
Midi configuration read from ".\midi.def"
Baton MIDI input set to device 0 = SB Live! MIDI In [3000]
Baton MIDI output set to device -1
Synthesizer output set to device 5 = SB Live! MIDI Out [3000]
Other Computer MIDI input set to device -1
Other Computer MIDI output set to device -1

 UPPERCASE COMMANDS
   D = print a description of this program given by the programmer
   ? = print this command list          M = print more commands
   Q = quit the program
   S = silence synthesizer
   P = position display toggle          B = buf display toggle

CONDUCTOR COMMANDS s--select score   z--play   n--no baton mode on/off
score file is beethoven
type COMMAND (type z to start playing)
```

Figure 3--COMMAND SCREEN FOR THE COND PROGRAM

Commands are executed by typing the indicated letter. Most of the commands do not require additional explanation. We will make a few comments about some of them.

1. The "s" command will select a compiled score. A compiled score has the same name as a source score, but with the extension ".p".

2. The z command will start playing the score. A source score must be compiled with the compile.exe program. The source score can either be a conductor program score or a type 1 midi file plus a header file. The header file is a conductor program score that sets the expressive controls and makes certain initializations. The compile program will generate a file with extension .p which is the file the cond program reads during a performance

3. Normally the score will start playing at measure 1. The "m" command allows the score to start at the beginning of any measure. The measure will remain in effect until a new "m" command is typed.

4. There are several useful test programs. The B program tests all 16 a-d inputs. The P test program displays the xyz data for each baton and the signals from the four pots.

MIDI FILE PROGRAMS

The cond program can also play scores that are written on midi files. However, although the notes in the score can be written in a midi file, the expressive controls have no standard midi notation. In addition the beats (trigger points) must be added to the midi file notes.

To provide these necessary augmentations to the midi file, a header file accompanies the midi file and the first track and the first channel of the midi file are used exclusively to indicate the beats and certain other control information. The header file and the midi file are read alternately by the compile program to generate the .p file.

The midi file must be a type 1 midi file. The midi file is called score.mid and the header file is called score (no extension) where score is the name of the file. The resulting .p file generated by the compile program is score.p.

A simple example of a midi file will be discussed next. More complex midi files and header files are on the disk which accompanies the baton.

The header file is shown in Figure 4

```
*demo--HEADER FILE FOR demo.mid midifile*

I                   *initializers cond program*
v60                  *sets initial default inverse tempo to 60*
Z127 &2ky1          *replaces channel 2 midi key velocity control with y1 control*
q0 h2 v0 c10        *puts channel 2 voice in left channel*
q2 h2 y2 c11        *controls channel 2 continuous loudness with y2*
q0 h2 v100 c7       *measure 1-> initial  channel 2 c7 volume to 100*
M                   *transfers compile program to read midi file*
q0 h2 v120 c7       *measure 2-> increases channel 2 c7 volume to 120*
M                   *transfers compile program to read midi file*
0                   *final 0 (zero) to terminate score*
```

Figure 4-- HEADER FILE FOR A MIDI FILE SCORE

The compiler starts by reading the header file. A “M” in the header file transfers the compiler to read the midi file. A C#5 (where C5 is middle C) in channel 1 of the midi file causes the compiler to resume reading the header file. A subsequent M in the header file sends the compiler back to the midi file, etc. Thus the compiler assembles the .p file by alternately reading the header file and the midi file. In this was, the expressive control can be changed at any point the score.

The header file must not contain any notes to be played or any dots or commas specifying time intervals. All time information must be in the midi file.

The staff notation for the midi file demo.mid is shown in Figure 5. The track 1, channel 1 contains trigger points and certain notes which control the compile program.

The beginning of a C5 (middle C) in channel 1 denotes a trigger point. This note IS NOT played by the synthesizer. The end of the C5 is also ignored, but the C5 MUST be short enough so successive C5's do not overlap.

In addition to C5's, track 1 contains C#5's and D5's. The C#5 causes the compile to return to the header file.

A D5 increments a counter which keeps track of the measure numbers.

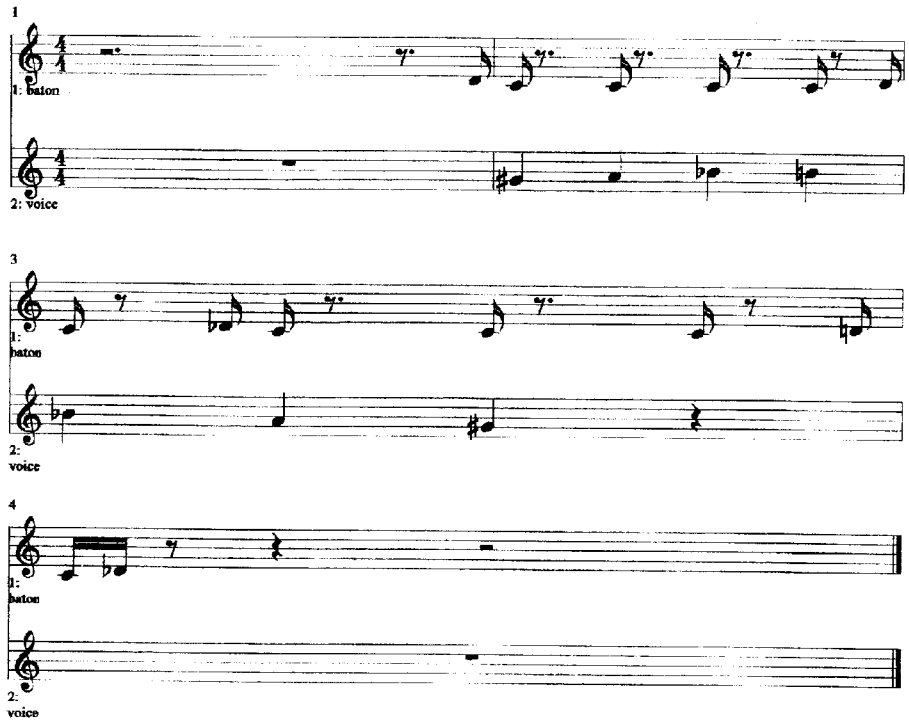

Figure 5-MIDI FILE, demo.mid, TRACK 1 IS BATON TRACK, TRACK 2 IS VOICE

The first measure of the midi file should contain only a D5 which will start the measure counter. All C5's and all playable notes (in channels 2 through 16) should start in measure 2 of the midi file. Measure 2 will be labeled measure 1 during the cond performance.

The D5 flat in measure 3 will return the compile program to the header file where it will reset the c7 control change for channel 2 to increase the loudness of voice 2. The "M" in the header file will then return the action to the midi file where the rest of the notes and triggers will be played. The final D5 flat in measure 4 will return to final 0 (zero) in the header file to terminate the score.

.

APPENDIX 1-CONDUCTOR PROGRAM SCORE--SYMBOLS & REAL-TIME CONTROLS

0 through 24	Select voice
A through G, a through g	Specify note pitch
An,Bn,etc	Pitch letter can be followed immediately (no space) by an accent number, n, which will be added to the key velocity. Accent on a note of chord will be applied to subsequent notes to end of chord or until a new accent is written
+A,+B, etc	Start note using midi noteon convention. Midi notes must be explicitedly turned off
-A,-B, etc	Midi noteoffs
RESET button	Resets baton to power-up state
B14- button	Advances to next measure in score
B15- button	Can be used as sustain pedal (see qn hm s15 c64 command below)
.	Specify one dot beat-time duration
	One dot normally equals 8 commas (see j command below)
,	Specity one comma beat-time duration
/	Trigger point
#	Sharp accidental
@	Flat accidental
!	Octave down accidental
^	Octave up accidental
$	Natural accidental
* xxxxxx *	Comment (max length--240 charactors)
hn	Set midi channel of voice to n
I	Initialize ns score compiler program
jn	Set one "dot" equal to n "commas"
	note--jn must be at the beginning of a voice number 0 line
Kn# Kn@	Set key signature n is number of flats or sharps
kun	Set key velocity control
	un can be:
	x1,y1,or z1--#1 stick control
	x2,y2,or z2--#2 stick control
	p1,p2, or p3--pot control

vi--where -1<i<128--setting
constant
M Transfer score reading to Midi file
on (o is lower case "oh") Set transpose for one voice.
Keyno of note "C" is n
p Turn sustain of one voice on
P Turn sustain of one voice off

qn hm si [wj ek rl tp] ck Setup control change
qn specifies patchcord n (0...15)
hm specifies midi channel m
si can be:
x1,y1,or z1--#1 stick control
x2,y2,or z2--#2 stick control
p1, p2, or p3--pot control
vi where -1<i<128--setting constant
ck where k is control change #
k=0 will remove patchord
[wj ek rl tp] optional linear
transform of the x,y,or z
stick variables
j,k,l,p are integers
in the range 0 to 127
qn hm s15 c64 make B15- a sustain pedal for
midi channel m
&mky1, &mky2, etc sets key velocity in midi channel m
to value specified by baton
x1 or y1 or z1 or p1 or vi etc
useful which playing a midi file
&mtn sets preset timbre in midi channel m to
value n
&CbT MSB LSB patch & bank select
C=channel (1-16), T=preset # (0-127),
MSB=bank # most significant byte, LSB=bank # least significant byte
example–for roland jv-1010 in “perform” mode, to select
Solo Fr.Hrn1 in channel 2 write in score &2b81 84 2
this command will cause the following three midi commands
to be sent:
b1 0 84, b1 32 2, c1 81

r Rest
tn Send midi program change n

Un or U-n	Set transpose up or down in midi key numbers
vn	Set tempo n = 240,000/tempo (0<n<256) where tempo is commas/minute
Vn	Set tempo, n = dots/min
w	Set or reset continuous tempo control
W	Set or reset fixed tempo, ignore triggers
(vvvvv)z	Define macro z where z is a letter a,b,...z which names the macro v may be any charactor. the charactor X is a dummy variable which will be repaced by a substituted variable when the macro is evaluated
'zn	Evoke macro z n times. If n is omitted the macro will be evoked once
[u v w]'z	Define sequence of substitutions for dummy variables in a macro. u v and w are substitution variables which are successively substituted for dummy variables in the macro. Substitution variables are one or more charactors. A A blank is the spearation charactor between substitiution variables.
s (key on computer)	Quiet synthesizer (works only when batan is in COMMAND mode)
Zn	Change keyvelocity in midi noteon's by subtracting n from the keyvelocity that would otherwise be used
0	0 (zero) required at end of score

APPENDIX 2–RECENT ADDITIONS TO THE CONDUCTOR PROGRAM

LINEAR TRANSFORMATION OF CONTROL CHANGES

An optional feature has been added to the qn command which allows the linear transformation of control change functions.

The control change command can now be written:

qn hm si [wj ek rl tp] ck

where the [parameters] are optional and specify two points on a linear transformation of the x or y or z data from either baton before it is sent to the synthesizer as a control change.

j,e,l, and p are integers in the range 0...127 which specify the two points.

j=X1
k=X2 (X2 > X1)
l=X1t
p=X2t

if X1=X2, no transformation will be made.

The linear transform can be used to change the range of the control change or even to change the slope of the control change function from positive to negative.

Example 1;

q2 h3 x2 w5 e120 r64 t127 c7

will compress the range of control change 7 in channel 3 to go from 64 to 127 as the x2 baton variable goes from 5 to 120.

Example 2:

q2 h3 y2 w5 e120 r64 t64 c7

will completely compress the range of control change 7 in channel 3 to be constant at 64 independent of y2 (this is a stupid thing to do);

Example 3:

q2 h3 z1 w40 e127 r127 t30 c7

will make control change 7 in channel 3 go from 30 to 127 as baton 1 is lifted and z1 goes from 127 to 40.

Example 4:

q2 h3 x2 w0 e127 r127 t0 c7

will make control change 7 in channel 3 go from 127 to 0 as y2 from baton 2 goes from 0 to 127.