

DECUS

PROGRAM LIBRARY

DECUS NO.	8-804
TITLE	MUSIC: PDP-8 MUSIC PLAYING PROGRAM
AUTHOR	Richard Wilson and Others
COMPANY	Digital Equipment Corporation Maynard, Massachusetts
DATE	11 February 1976
SOURCE LANGUAGE	PAL8

ATTENTION

This is a USER program. Other than requiring that it conform to submittal and review standards, no quality control has been imposed upon this program by DECUS.

The DECUS Program Library is a clearing house only; it does not generate or test programs. No warranty, express or implied, is made by the contributor, Digital Equipment Computer Users Society or Digital Equipment Corporation as to the accuracy or functioning of the program or related material, and no responsibility is assumed by these parties in connection therewith.

GENERAL INFORMATION

Object Computer(s) PDP-8 (Except 8/s or PDP-12) Source Computer (if different) _____
File Name MUSIC Version No. _____
Title PDP-8 Music Playing Program
Author Richard Wilson and others
Submitter (if other than author) _____
Affiliation _____
Address Digital Equipment Corporation
Maynard, Massachusetts Country _____
Monitor/Operating System Paper tape or OS/8 DEC No. _____
Core Storage Required 4K or more Starting Address _____
Peripherals Required a radio and the device of distribution
Other Software Required Binary Loader or OS/8 DEC or DECUS No. _____
Source Language PAL8 Category _____
Restrictions, Deficiencies, Problems _____
Date of Planned or Possible Future Revisions _____

TAPES AVAILABLE

Paper Tapes Object Binary Object ASCII Source Other _____
DECtape LINCtape Format PDP-8 Magtape: 7 Track 9 Track BPI _____
Object Files Source Files Documentation Files Other _____

Also available on two floppy diskettes

MUSIC is a program which will play music in four part harmony on any PDP-8 family core memory computer, except the 8/S or PDP-12. The music to be played is input to the program as a standard OS/8 ASCII file. The music may be picked up by the use of an AM radio, or by a simple interface. The OS/8 distribution media include the source of the player, which can be customized for various configurations, along with approximately 45 minutes of music, such as Joplin, Bach, Beethoven, movie tunes, etc.

The binary paper tape is intended for any 1.5 microsecond PDP-8, and runs in 4K, but will only play short tunes. Several short tunes are available on paper tape.

1. INTRODUCTION

This section gives a general overview of the music player, and how it works.

The music player is based upon a certain computer instruction. For the purposes of this discussion, assume that the instruction is the CAF instruction of the PDP-8/E, M, F, or A. Assume that a program were to execute the CAF instruction 440 times a second. The CAF instruction is one which is used to clear out various peripherals (its function is the same as that of the "CLEAR" key). Thus, when this instruction is executed, logic all over the computer goes "zap", clearing out various registers. Now, if a radio is held close to the computer, it will pick up some of this energy, and 440 times a second, will deliver a pulse to the speaker. The result is that you will hear a tone from the radio -- as a matter of fact, you will hear an A.

By very careful programming, it is possible to play any of several notes from the radio, and in fact, up to four at once. This is done by simply overlaying the patterns for each of the notes it is desired to play. Various refinements can be made to improve the reception of the signal, and sometimes an instruction other than CAF must be used, but the principle is still essentially the same.

Playing music is a two-part operation. When the player is first started, it reads the ASCII input, and as it reads it, it "compiles" it and places the results in a core buffer. When the end of the input is found, the music is played. At the end, the music can be played again, or another piece can be "compiled". While the music is being compiled, if any errors in syntax are detected, an error message is printed. More about all that later.

1.1 THE VARIOUS VERSIONS OF THE MUSIC PLAYER AND WHY THEY EXIST

There are several factors which make it difficult for one player to operate on all systems. They are:

COMPUTER SPEED. The frequency and speed of the music depend directly upon how long it takes to execute certain instructions. For this reason, MUSIC will not play on a semiconductor 8/A. An 8/F is noticeably faster than a core 8/A.

INSTRUCTION SFT. Some computers have the BSW instruction, others do not. Some have an MQ. Some don't have the CAF instruction.

CONFIGURATION. Some computers have more memory than others, which is nice to take advantage of. Some computers have peripherals which do something mechanical as the result of the CAF instruction (mostly RX8E, the floppy drive), so that the CAF instruction should not be used. Some configurations don't run OS/8.

1.2 WHAT ARE THE VARIOUS VERSIONS?

While this is being written, it is planned to distribute three versions of the player. The first two are found on the OS/8 distribution media, and are called MUSIC.SV and MUSIC1.SV. MUSIC is configured to run on a machine with a cycle time of 1.5 microseconds, without the use of the CAF, BSW, or MQ instructions. It needs only 8K, so it will run on almost any OS/8 system, although somewhat too fast on an 8/E, M, or F. MUSIC1 will run properly only on a 12K 8/E, M, or F. It uses the CAF instruction, so if the system includes an RX01, it will go crazy. Although MUSIC1 will operate on an 8K 8/E, it will not be able to play a very long tune.

The third version is to be distributed as a binary paper tape, and uses only 4K of memory. It accepts input only from low-speed or high-speed paper tape. It is configured to run on any 1.5 microsecond PDP-8. Other versions of the player can be assembled from the PAL source of MUSIC, found on the OS/8 media. Instructions for this follow in a later section.

1.3 HOW TO RUN MUSIC ON AN OS/8 SYSTEM

First, the proper version of MUSIC should be stored on the system device. If the computer is a 12K or larger 8/E, M, or F, without an RX01, copy the file MUSIC1.SV to the system device, changing its name to MUSIC.SV. Otherwise, copy over MUSIC.SV.

Now, the music source files (all those with extensions of .MU) may be copied to the desired device. For convenience, place the desired files on the device DSK:.

Now, run MUSIC by typing:

```
R MUSIC
```

MUSIC will call the command decoder, which will respond by

printing its asterisk (*). In response to the asterisk, type the names of up to nine tunes to be played, separated by commas. Any standard command decoder input (without wild cards) may be used. For example:

```
GMINOR,LTNSLD,ENTER
```

If you're lucky, MUSIC will play all three tunes, and then return to the OS/8 monitor. Typing CTRL/O will stop the printing of error messages. Typing CTRL/Q will cause the player to be Quiet--it will act as if it had reached the end of the tune. Setting switch 0 on will cause the current tune to be repeated. Setting switches 6 through 11 allows the volume to be altered. Note: For technical reasons, a character typed at the keyboard while music is playing may be misinterpreted by the player. If a random key is hit, don't be surprised if the player treats it as a CTRL/Q or a CTRL/C.

1.4 HOW TO RUN MUSIC ON A PAPER TAPE SYSTEM

The binary paper tape of MUSIC should be loaded into core using standard loading procedures. Load address 200, and start the computer. Now, place the paper tape of one of the tunes in the high speed or low speed reader, set the switch register to all zeroes (for high speed), and wait for the tape to load. If using the high speed reader, it may be necessary to depress the feed switch momentarily. If you want to keep the high speed reader from starting, perhaps to keep your finger from getting chewed up, turn on any bit in the switch register until you have loaded the tape in the reader.

After the tape has read in, the tune will be played. If it is desired to repeat the tune, simply set switch register bit zero, or do a manual restart from address 0. To interrupt the tune, type a CTRL/C or a CTRL/Q (for Quiet). For new tunes, make your own paper tape using the instructions later in this document.

If a tune has been interrupted with a CTRL/C or a CTRL/Q, it is necessary to remove this character before the end of the next tape has been reached, or it will never be played. To do this, simply type a space as soon as the CTRL/C or CTRL/Q has taken effect. (Typing another character could cause an error by injecting an undesired character into the middle of the input)

2. LISTENING TO THE MUSIC

The program plays music by periodically executing a CAF instruction (6007). To listen to MUSIC, it is somehow necessary to pick up this instruction. There are several ways, three of which are given below:

2.1 LISTENING WITH A RADIO

Simply hold a normal AM radio close to the computer or any of its peripherals. By experimenting with the radio dial and the positioning of the radio, it is possible to hear the music. Although this is the simplest, safest, and least expensive method, it is not as satisfactory as either of the following methods.

2.2 LISTENING BETTER WITH A RADIO

Find the signal INITIALIZE or POWER CLEAR. This is easiest somewhere on the positive (or negative) I/O bus, if present on the system. Connect this, through a small capacitor, to the antenna of an AM radio. The idea of the capacitor, which should be high-voltage, probably no larger than .001 mf, is to protect the computer and the radio from each other. ANY ELECTRICAL CONNECTIONS TO THE COMPUTER MUST BE DONE ONLY BY QUALIFIED PERSONNEL.

2.3 LISTENING VIA AN INTERFACE

Those knowledgeable in electronics could build a simple interface. For example, the INITIALIZE signal can be fed through a one-shot to lengthen it to about 6 us, and then shaped and amplified by circuitry of your choice. A capacitor in the right place can do wonders for smoothing out the harshness of the sound.

The player is capable of putting out some rather low notes, down to about 40 Hz or so. For the most impressive reproduction, make sure that the speaker and electronics are capable of reproducing these notes.

Note: If an instruction other than CAF is used, the INITIALIZE or POWER CLEAR signals will not be usable for interfacing. You can still listen with a radio, or an interface can be designed to pick up whatever instruction is being used.

3. ADVANCED OS/8 OPERATING INFORMATION

3.1. INVOKING MUSIC UNDER OS/8 VERSION 3

In addition to the standard R and RUN commands, MUSIC may be made known to CCL so that the PLAY command will invoke MUSIC. (see the section on modifying CCL) There are now two more methods of invoking MUSIC:

PLAY (command-decoder-line)

will play the music specified to the command decoder, and return to the keyboard monitor.

PLAY (command-decoder-line)\$

(where \$ stands for an ALTmode or an ESCape) will play the music specified to the command decoder, and return to the command decoder to accept another line of input.

3.2 COMMAND DECODER: MULTIPLE INPUT FILES AND ALTMODE

If more than one file is specified to the command decoder, MUSIC will compile the files in order until a dollar sign is found, the end of the last file is found, or it has run out of core. It then plays that music, and then returns to the compiler. The compiler continues compiling with the beginning of the first unused input file, as above. When the last input file has been processed, MUSIC returns to the command decoder, if the command line was terminated with a carriage return. If the command was terminated with an ALTmode (or ESCape), MUSIC will return to the keyboard monitor. (Note that this operation is complemented when using the PLAY command)

3.3 INSERTING THE PLAY COMMAND INTO CCL

As mentioned previously, the PLAY command may be inserted into CCL to start MUSIC. Use the following procedure to fix up CCL, inserting the applicable devices as necessary. (Note that \$ stands for ALTmode or ESCape below)

```
.R ABSLDR
*CCL.SV/I
*PLAYOV.BN$
.SAVE SYS CCL;12001=2003
.R CCL
```

This should activate the PLAY command.

Note: PLAYOV is written for CCL version B, and may not work on other versions.

3.4 USING BATCH

MUSIC is compatible with BATCH. For a sample batch control file, see MEDLEY.BI. Since there is less core available when under BATCH, tunes which previously played with no problem could be too long under BATCH.

4. CONFIGURING MUSIC FOR YOUR OWN SYSTEM

To assemble MUSIC to fit your system, make a file with up to four lines, one from each group below:

- CPU=1 This line should be included in the file if your computer is a PDP-8/E, F, or M. MUSIC will assume a cycle time of 1.2/1.4 microseconds, will use the BSW instruction, and will use the MQ.
- CPU=2 This line should be included in the file if your computer is a core memory PDP-8/A. MUSIC will assume a cycle time of 1.5 microseconds, will use the BSW instruction, and will use the MQ.
- CPU=4 This line should be included in the file if your computer is a PDP-8/I, L, or PDP-8 (the earliest version of the 8) or a link-8. Music will assume a cycle time of 1.5 microseconds (which is close...) and will not use the MQ or BSW instructions.
- OS8=10 This line should be included if it is desired for the program to run under OS/8. MUSIC makes use of all core on the system, unless it has been restricted with the CORE command. If BATCH is active, the batch monitor is not wiped out.
- OS8=20 This line should be included if it is desired to have a paper-tape only version of MUSIC.
- CORE=100 This line should be included for systems which have only 4K of core. This line may not be included if the line OS8=10 is included. MUSIC will not use core above 7600.
- CORE=200 This line should be included for systems which have at least 8K of core. All core actually present on the system may be used, unless restricted by the OS/8 CORE command. Core above 7600 in every field will not be used, and the BATCH monitor, if present, will be preserved.
- CORE=400 This line is equivalent to CORE=200, with the following exception: If the computer is an 8/E (CPU=1), or an 8/A (CPU=2), the line CORE=400 will enable a special set of code which results in

a much better reproduction of sound. The reason that this feature is not included automatically is that it uses all of field one as a special buffer area, limiting the amount of core available for tunes. Thus, this line should be included only on machines which have at least 12K of core. The OS/8 code in the top of field 1 is restored upon a normal exit from the player. However, if the player does not terminate normally, this page will not be restored. If the player is halted, a restart from 200 or 7600 will restore this area.

NOISF=CAF If this line is included, the special instruction which is executed to make noise will be a CAF. Use this on all 8/A, E, M, and F systems, unless they have an RX01.

NOISF=IOF This is the recommended alternative if **NOISE=CAF** can not be used.

NOISE=instruction Any instruction which does not skip or set the AC non-zero may be used if you wish to experiment. For best results, use an instruction which takes no longer than 1.5 microseconds to execute.
Note: If you wish to experiment by changing the "noise" instruction, its address can be found in core location 2. Thus, if location 2 were to contain 163 (which it probably does not), then location 163 would contain the noise producing instruction (such as CAF) which may be manually changed, as desired.

Thus, for a typical 8/F system, the file might read:

```
CPU=1
OS8=10
CORE=200
NOISE=CAF
```

Once the file has been created, and assuming it has the name FILE.PA, MUSIC may be assembled by typing:

```
PAL MUSIC<FILE,MUSIC
```

and loaded by typing:

```
LOAD MUSIC
```

and saved on the system device by typing:

```
SAVE SYS MUSIC=2401
```







If an assembly listing is obtained, it may be desired to use the /J option. This will prevent PAL8 from listing lines which have been conditionalized out of the assembly, reducing the clutter quotient of the listing.

5. REPRESENTATION OF MUSIC FOR THE COMPILER


The following conventions are used when transcribing from a musical score to the form required by the compiler, namely an ASCII file in standard OS/8 format. These files are normally created with a text editor such as EDIT or TECO.

5.1. REPRESENTATION OF SINGLE NOTES

The various lengths of note are represented by:

D	for Demisemi-quaver	()
S	for Semi-quaver	()
Q	for Quaver	()
C	for Crotchet	()
M	for Minim	()
B	for semiBreve	()
G	for Grace note	

The grace note is defined by the compiler to have a length equal to one-third of a demisemi-quaver.

The normal convention for "dotted" notes is followed. One dot after a note increases its length by $\frac{1}{2}$ and two dots after a note increase its length by $\frac{3}{4}$. For example, C. represents a crotchet and a half (), or, in other words, 3 quavers.

Note that demisemi-quavers may not be dotted, and semi-quavers may only be dotted once.

The notes themselves are represented by the letters A, B, C, D, E, F, G, with the normal meanings.

A note may be sharpened by appending # to the note, and flattened by appending !. Double sharps (e.g. A##) and double flats (e.g. B!!) are also allowed.

The note A is equivalent to 220 Hz (cycles per second). This makes the note C the same as middle C on the piano, with the other notes A through G forming the octave around this C. Other octaves may be obtained by appending "+" signs or "-" signs to the note.

For example, C++ represents C two octaves above middle C,
C- represents C one octave below middle C,
and A+ represents the standard A at 440 Hz.

The range of notes allowed is over 6 octaves, from D#---
to A++++.

There is **one** more note, R, which is defined as a rest.
It is "played" just like any other note, but it makes
no sound.

5.2. PUTTING THEM TOGETHER

A tune is made up of a series of pairs, length and note.

E.g. Q F

represents a note F lasting for a quaver, or

 C.. B!+

represents a note B flat one octave up, lasting for 7 semiquavers.

The tune:



would be C C
 Q E
 Q G
 M C+

The pairs may instead be separated by a semi-colon if so desired:

 C C;Q E;Q G;M C+

5.3. METER

The speed the tune is to go at is represented in a form similar to the standard notation, e.g.

♩=100 means 100 crotchets to the minute. For the purposes of this representation, this would be C=100. In general, the form is: [length] = [number]. For example, M.=45 represents a speed of 45 dotted minims to the minute.

If no indication of time is given, C=60, i.e. 60 crotchets to the minute, is assumed.

The speed may be changed at any point in the tune by a further instruction of this form. (Triplets and quintuplets etc. may be obtained in this way.)

The fastest speed which is correctly played is a speed equivalent to C=341. Asking for a faster speed results in no error message, but it will not play as fast as desired.

5.4. KEY SIGNATURES

A key signature is represented by:

= list of notes, or ! = list of notes.

For example, #=F represents the key having just F sharpened, i.e. the key of G. Wherever the note F appears it will now be sharpened. However, accidentals take precedence over the key signature in the usual way. E.g. F! would still be F flat, not F flat sharpened again to F natural, and F# would remain F sharp and not become F double sharp.

#=F, C, G, D

would represent the key of E, having the four sharps shown.

! = B, E

would similarly represent the key of B flat.

The key signature may be changed at any point in the tune. If it is desired to return to the key of C at any point, this may be done by specifying #=R.

The equivalent for the natural sign is " , so that any note sharpened or flattened by the key signature may be naturalized again. For example, F" represents F natural. If the note was already natural, the sign will have no effect.

Since there is no provision for bar lines, accidentals only apply to the note to which they are appended. For this reason also, the sign combinations #" and !" are never required.

5.5. END OF TUNE

The end of the tune is signalled by a dollar sign \$. However, if there is no dollar sign, the end of the last input file will be treated as the end of the tune.

Paper tapes must include the dollar sign.

5.6. REPRESENTATION OF CHORDS

The rules as described for single notes also apply for chords with the following modifications and additions:

The notation for chords is of the general form:

[length] ([list of notes])

For example, Q (C, E!, G, C+)

which indicates a chord consisting of the notes C, E flat, G, and C+ lasting for one quaver.

Notes which do not appear as part of a chord are typed in the standard single note format.

For example,



would appear as

```
C (F, A+, C+, F+)
Q E+, C (F, A+, C+)
Q G+
C (D+, F+), M (F, A+)
C (C+, E+)
C (F, A+, C+, F+)
```

Note that any notes which begin sounding at the same time should be separated by commas and appear on the same line.

The number of notes or "voices" which the player can play at any one time is limited to four. This includes any rests (R). The compiler prints an error message if an attempt is made to play more than four notes.

5.7. TYING NOTES

There are certain lengths of notes that cannot be achieved without tying notes together. The symbol T may be used for this purpose, as a tie-marker.

For example,  would be MTQ

In general, the form is [length] T [length] T [length]...

5.8. USING Y

In longer tunes, it is often desirable to have some kind of "paragraph" marker to divide the tune into convenient chunks. The letter Y may be used for this purpose. The Y should be specified only at times such that there are no notes held over the Y. Notes held over a Y will cause an error printout, and the notes will be prematurely terminated.

Each section of music has a "paragraph number" associated with it. Any music up to the first Y is considered paragraph zero. If there is a line of the form Y=n, (for example, Y=320) then the music between that line and the next Y is paragraph n (paragraph 320 in our example). If a Y appears by itself, then the music between that Y and the next Y has a paragraph number one greater than the preceding paragraph. The paragraph number is printed with any error messages. No paragraph number may be greater than 4095.

6.0. ERROR MESSAGES

After a tune has been requested to be played, it is "compiled". At this time, if any mistakes are found in the syntax, an error message will be printed on the terminal. The message will consist of two numbers followed by the offending line. In addition, an asterisk (*) will be printed, usually just before the character where the error was detected.

The two numbers printed with the error message are, respectively, the paragraph number and the line number. The paragraph number is determined by Y's as described above. The line number is the number of lines after the last Y that the error was found.

As an example of an error, assume that a tune includes the line:

C=A

The error message might appear:

23 2 C=*A

This means that the line in error is in paragraph 23, 2 lines after the Y. In this case, the compiler, seeing C=, expects a number (as a meter setting). Seeing instead a letter, it marks it as an error.

If a piece is too long to fit in core, an error message will also be printed. It can be recognized because it ends with a dollar sign, at the point where core ran out.

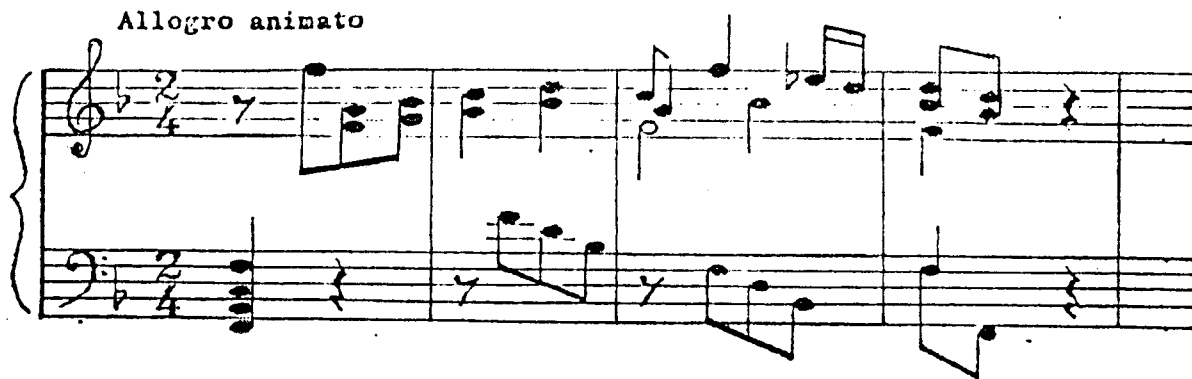
6.1 NON-PRINTING ERRORS

There are several types of errors which cause no print-out, and are detected only when the piece is played. The most common type of error is a simple mis-transcription, such as a missing sharp. The cure is obvious.

Another error is caused by excessive demands on the player. Some versions of the player (8/A and 8/E with CORE=400) calculate ahead of time what they will be doing in the future. As more notes, higher notes, and louder notes are played, the time demands on the player increase. If these time demands exceed the time available, the player gives up and just makes noise until it is able to catch up again. If the reproduction of a tune turns to noise during a high passage, do one of the following:

- 1) Make the notes lower. This is often easiest by preceding the tune with a line such as T=96, to lower the entire melody. Otherwise, some of the individual high notes could be changed.
- 2) Shorten the high notes, perhaps by inserting rests between the notes. This technique is used at one point in ENTER.
- 3) Reduce the volume of the offending passage by turning off some of switches 6-11

7. SAMPLE TRANSCRIPTION



could be transcribed as:

C=95

V SAMPLE: BACH'S ITALIAN CONCERTO

!=B

C (F-,C-,F--),Q A- **
 Q F+
 Q (A+,F)
 Q (B+,G)

Y
 C (C+,A+),Q R *
 Q F
 C (D+,B+),Q D
 Q B

Y
 Q C+,C A+,M F
 C F+,Q F-
 C B+,Q D-
 S E!+,Q B-
 S D+

Y
 Q (D+,B+,F-),C F
 Q (C+,A+,F--)
 C R
 \$

- * The rest here is necessary so that the following line begins playing at the right time
- ** Notice the useful technique of shortening one note in a chord to keep from playing more than four notes at one time.

8. ADDITIONAL COMPILER FEATURES

Some of the following features may be useful at times:

8.1 SUBTRACTION OF DURATIONS

To get a note length "just a little shorter", use a minus. For example, a note which is shorter than a Crotchet by the length of a grace note could be written as QTSTDTGTG or Q..TGTG, but using subtraction it would be simply C-G. Combinations such as BTC-G or B-C-G are also valid.

8.2 TRIPLETS

To facilitate transcription of triplets, use a 3 before a length. This will shorten that length to 2/3 of what it would otherwise be. For example, 3C is a length equal to 2/3 of a crotchet.

8.3 COMMENTS

Any line beginning with a \vee is ignored (up to the first carriage return or semicolon). Therefore, such a line can be used to insert comments. This is often useful to help correlate the score with its transcription.

8.4 FORMATTING

All non-printing characters other than CR and LF are ignored. They may be used freely for formatting the file for ease of reading. Note that spaces but not tabs will be printed in any error message.

8.5 BLANK LINES

Blank lines may be inserted anywhere. A Carriage Return is syntactically equivalent to a semi-colon. A line feed increments the error message line counter. To prevent confusion, it is imperative that CR and LF appear only in pairs in that order. (This is normally done automatically by EDIT and TECO)

8.6 TRANSPOSITION

An entire section of music may be transposed up or down by a line of the form T=n. The default is T=100. T=102 will transpose up one whole note, and T=88 will transpose down one octave. A transposition shifts the range of notes which the compiler accepts as valid.

9. MISC.

9.1 TUNING

On some systems (particularly if CPU=4 or CORE is not 400) the player may need "tuning". The constant in core location 3 determines the relationship of high notes to low notes. If the player goes flat when higher notes are played, try making location 3 smaller. If the player goes sharp when higher notes are played, try making location 3 greater.

9.2 HAVING FUN

It's rather interesting to work with this program, because bugs are sometimes so interesting to listen to. For an example of what I mean, try de-tuning the player by changing the contents of core location 3.

9.3 OTHER HARDWARE

Those who have digital to analog converters, floating point processors, and other specialized hardware can probably use this program as the basis for a much more flexible music-reproducing system. As it is, the poor PDP-8 is spending most of its time trying to make sure that everything stays in tune.

9.4 CHANGING THE SPEED

If it is desired to change the overall speed of the player, without changing the pitch, the constant in location 100 can be changed. Dividing this constant by two would cause the player to play approximately twice as fast.

VECTAPE

MUSIC .SV	9	12-FEB-76
MUSIC1.SV	9	12-FEB-76
MEDLEY.BI	1	12-FEB-76
PLAYOV.PA	1	12-FEB-76
PLAYOV.BV	1	12-FEB-76
MUSIC2.MU	14	12-JAN-75
MUSIC3.MU	14	12-JAN-75
MUSIC5.MU	4	12-JAN-75
USA .MU	3	05-MAR-75
MUSIC6.MU	13	06-OCT-75
YANKEE.MU	3	17-OCT-75
INV212.MU	9	28-AUG-75
INV213.MU	9	28-AUG-75
FUGUE .MU	39	01-FEB-76
ENTER .MU	23	01-FEB-76
INV201.MU	7	01-FEB-76
INV212.MU	14	01-FEB-76
INV214.MU	10	01-FEB-76
FIFTH1.MU	40	01-FEB-76
FIFTH3.MU	35	01-FEB-76
MAPLE .MU	36	09-FEB-76
MUSIC1.MU	10	09-FEB-76
MINUET.MU	16	09-FEB-76
GRINCR.MU	27	09-FEB-76
INV315.MU	12	09-FEB-76
MINUTE.MU	25	09-FEB-76
11NSLD.MU	20	09-FEB-76
LDVAIL.MU	17	12-FEB-76
MUSIC4.MU	13	12-FEB-76
BABYEL.MU	11	12-FEB-76
JCOOK2.MU	8	12-FEB-76
JCOOK1.MU	7	12-FEB-76
INV204.MU	9	12-FEB-76
INV208.MU	10	12-FEB-76
MUSIC .FA	92	09-FEB-76

159 FREE BLOCKS

FLOPPY 1

MUSIC .SV	9	10-FEB-76
MUSIC1.SV	9	10-FEB-76
DODLEY.BI	1	10-FEB-76
PLAYOV.PA	1	10-FEB-76
PLAYOV.BK	1	10-FEB-76
MUSIC2.MU	14	12-JAN-75
MUSIC3.MU	14	12-JAN-75
MUSIC5.MU	4	12-JAN-75
USA .MU	3	05-MAR-75
MUSIC6.MU	13	06-UCI-75
YANKEE.MU	3	17-UCI-75
INV210.MU	9	20-AUG-75
INV213.MU	9	20-AUG-75
FUGUE .MU	09	01-FEB-76
ENTER .MU	23	01-FEB-76
INV201.MU	7	01-FEB-76
INV212.MU	14	01-FEB-76
INV214.MU	10	01-FEB-76
FIFTH1.MU	40	01-FEB-76
FIFTH3.MU	05	01-FEB-76
MAPLE .MU	06	09-FEB-76
MUSIC1.MU	10	09-FEB-76
MINUET.MU	16	09-FEB-76
GRINOR.MU	27	09-FEB-76
INV310.MU	12	09-FEB-76
MINUTE.MU	25	09-FEB-76
INSLD.MU	20	09-FEB-76
WIL.MU	17	10-FEB-76
MUSIC4.MU	13	10-FEB-76
BARYEL.MU	11	10-FEB-76
JCOCK2.MU	8	10-FEB-76
JCOCK1.MU	7	10-FEB-76
INV204.MU	9	10-FEB-76
INV206.MU	10	10-FEB-76

6 FREE BLOCKS

FLOPPY (2)

MUSIC.PA	92	09-FEB-76
PLAYOV.PA	1	10-FEB-76
PLAYOV.BN	1	11-FEB-76
MUSIC2.MU	14	12-JAN-75
MUSIC3.MU	14	12-JAN-75
MUSIC5.MU	4	12-JAN-75
USA.MU	3	05-MAR-75
MUSIC6.MU	13	06-UC1-75
YANKEE.MU	3	17-UC1-75
INV210.MU	9	28-AUG-75
INV213.MU	9	28-AUG-75
FUGUE.MU	39	01-FEB-76
ENTER.MU	23	01-FEB-76
INV201.MU	7	01-FEB-76
INV212.MU	14	01-FEB-76
INV214.MU	10	01-FEB-76
FIFTH1.MU	40	01-FEB-76
FIFTH3.MU	35	01-FEB-76
MAPLE.MU	36	09-FEB-76
MUSIC1.MU	17	09-FEB-76
MINUTE1.MU	16	09-FEB-76
GMINCR.MU	27	09-FEB-76
INV315.MU	12	09-FEB-76
MINUTE.MU	25	09-FEB-76
ITNSLD.MU	20	09-FEB-76
JCOOK2.MU	8	12-FEB-76

2 FREE BLOCKS

CONTENTS OF MEDIA:
PDP-8 MUSIC PROGRAM

- A. PDP-8 Dectape
Contains two save-format versions of ~~the~~ MUSIC, MEDLEY (a BATCH control file), 29 tunes, PLAYOV (PAL source and binary), used to modify CCL, and the source of ~~the~~ MUSIC.
- B. FLOPPY NUMBER ONE
Contains everything which is on the PDP-8 Dectape, except for the source for MUSIC.
- C. FLOPPY NUMBER TWO
Contains the source for MUSIC, PLAYOV (PAL source and binary), and 23 of the 29 tunes.
- D. Binary paper tape
Contains a non-OS/8, 4K version of ~~the~~ MUSIC, for any PDP-8 with a 1.5 microsecond cycle time
- E. music tape one:
Contains The Entertainer by Scott Joplin
- F. music tape two:
Contains "Love Will Keep Us Together", "America the Beautiful", and "Yankee Doodle".
- G. music tape three:
Contains the Minuete Waltz
- H. Music tape four:
Contains three two-part inventions by Bach: numbers 1, 4, and 8.