AI TR-242

A PARALLEL PROCESSING MODEL OF MUSICAL STRUCTURES

Stephen W. Smoliar

September 1971

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

CAMBRIDGE MASSACHUSETTS 02139

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Massachusetts Institute of Technology Artificial Intelligence Laboratory	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP None

3. REPORT TITLE

A Parallel Processing Model of Musical Structures

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Ph.D. Thesis, August 1971

5. AUTHOR(S) *(Last name, first name, initial)*

Stephen W. Smoliar

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1971	276	48

8a. CONTRACT OR GRANT NO. N00014-70-A-0362-0002 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) AI TR-242 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* MAC-TR-91

10. AVAILABILITY/LIMITATION NOTICES

Distribution unlimited to all.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3d-200 Pentagon Washington, D.C. 20301

13. ABSTRACT

EUTERPE is a real-time computer system for the modeling of musical structures. It provides a formalism wherein familiar concepts of musical analysis may be readily expressed. This is verified by its application to the analysis of a wide variety of conventional forms of music: Gregorian chant, Mediaeval polyphony, Bach counterpoint, and sonata form. It may be of further assistance in the real-time experiments in various techniques of thematic development. Finally, the system is endowed with sound-synthesis apparatus with which the user may prepare tapes for musical performances.

14. KEY WORDS

EUTERPE
Musical Structures
Music
Computer Music
Sound-Synthesis
Musicological Foundations
Counterpoint and Harmony

DD FORM 1 NOV 65 1473 (M.I.T.)

UNCLASSIFIED
Security Classification

A PARALLEL PROCESSING MODEL OF MUSICAL STRUCTURES

by

Stephen W. Smoliar

Massachusetts Institute of Technology

September 1971

Revised version of a dissertation submitted to the Department of Mathematics on August 10, 1971 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ACKNOWLEDGEMENTS

Were I to attempt to be thorough in the expression of thanks, I would have to write an autobiography, rather than a paragraph of acknowledgements. But, as the reader will soon discover, I have written enough in one stint; and I would rather not undertake another large project at this time. Nevertheless, there are a few persons who _must_ receive thanks for their roles in the project from which this document has sprung.

First and foremost, I shall always be indebted to Professor Marvin Minsky. When I first began my experiments, six years ago, he provided me with facilities which were more than adequate. Since then he has followed me, prodded me, and encouraged me in my efforts to unite two disciplines which had always been rather thoroughly separated.

Secondly, I cannot be too thankful to Ezra Sims, music director of the New England Dance Theatre and EUTERPE's honorary godfather. In serving as my composition teacher, he gave me the courage to approach music armed primarily with common sense. That he, in his capacity as a composer, has put the EUTERPE system to good use is perhaps one of the most encouraging results of the entire project.

I must also thank Professor David Hughes of Harvard University for having taken the trouble to read this manuscript and for having set me straight in matters of Mediaeval music in which, like many self-taught scholars, I was hopelessly naive.

It is customary at this point for the author to thank his wife. Being unmarried, I am unable to do so. Nevertheless, there are two "women in my life" whom I must thank. The one is Mrs. Elizabeth Martin, with whom I co-directed the ID Company for two years; she cast the die which has since determined my involvement in the musical arts. The other is Miss Carol Eisenberg who, roughly one year ago, asked a few simple questions whose answers have now been provided by this dissertation. Finally, I am more than grateful to Miss Eva Kampits and Mrs. Charlene Ferrante for their preparation of the final manuscript.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0002.

A PARALLEL PROCESSING MODEL OF MUSICAL STRUCTURES

Abstract

EUTERPE is a real-time computer system for the modeling of musical structures. It provides a formalism wherein familiar concepts of musical analysis may be readily expressed. This is verified by its application to the analysis of a wide variety of conventional forms of music: Gregorian chant, Mediaeval polyphony, Bach counterpoint, and sonata form. It may be of further assistance in the real-time experiments in various techniques of thematic development. Finally, the system is endowed with sound-synthesis apparatus with which the user may prepare tapes for musical performances.

Thesis Supervisor: Professor Marvin Minsky

TABLE OF CONTENTS

INTRODUCTION

1. The Modeling of Music

The object of this dissertation is to provide a framework for the modeling of musical structures of a more formal nature than conventional verbal style analysis. Our notion of "model" follows the definition given by Marvin [Minsky] in his paper, "Matter, Mind and Models":

> To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A.

In order to establish a formal approach, it is necessary to make some basic decisions about <u>what</u> is to be formalized. In other words, we shall have to make some assumptions (hopefully, of a general enough nature) about the sorts of questions which the observer will ask of the model. As Susanne [Langer] has noted, it is the <u>questions</u>, rather than the answers, which characterize different philosophies; and the same is true for the different approaches to musical analysis.

The actual questions of a philosophy are rarely stated in their most explicit form; rather, they are implicit in the <u>Weltanschauung</u> -- those basic attitudes which are taken for granted -- of the philosopher. Therefore, as a starting point for our own study, we should attempt to dig out the questions underlying conventional analytic techniques. It is, perhaps, unfair to characterize a school of thought by its extremes; but extremes are most useful in revealing basic foundations. Music analysis has two such extremes, and we shall consider examples of both.

The following is from an analysis by Daniel Gregory [Mason] of Haydn's Symphony No. 93:

> But the most Beethovenish trick of all is perhaps the modulation back to the last entrance of the main theme of the finale of this same symphony. The key of the movement is D-major; Haydn, however, getting himself well established in F-sharp minor, harps on C-sharp as the dominant of this distant key; many C-sharps are heard, in a persistent rhythm of two shorts and a long, until one has forgotten all about the original key of the piece; the C-sharps fade away to piano, then to pianissimo, then to silence; when suddenly, in the same rhythm, three loud D's bring the piece emphatically back to the home key, and forthwith it proceeds merrily upon its way.

At the other extreme we have György [Ligeti]'s discussion of Pierre Boulez:

> . . . the compositional process can be reduced to three working stages: <u>Decision I -- Automatism -- Decision II</u>.

Decision I

A. Selection of elements

B. Choice of an arrangement for these elements.

C. Choice of the further operations to be carried out with these arrangements ('arrangements of arrangements') and mutual relationships of the individual arrangements to each other.

Automatism

Elements and operations, once selected, are, as it were, fed into a machine, to be woven into structures automatically, on the basis of the relationships chosen.

Decision II

The automatically derived structure is to some extent crude, and one must work on it further, taking decisions in dimensions that are not employed mechanically. If, for example, the parameter 'dynamics' or 'register' has not been passed into the machine, then one can work over the crude structure by directing these left-over parameters. This can be done aleatorically, or with definite formal aims, such as to form or avoid particular connections within the given crude structure.

Fortunately, not all analyses which take Ligeti's approach are so negligent of what is actually heard in a composition, but many such analyses seem to stem from the question, "How may one build a musical composition?" Hence, they tend to be similar in form to a recipe in a cookbook. Mason's concern, on the other hand, regardless of his use of language, would appear to be based on the question, "What do we hear in a given piece of music?" The analysis is essentially a description of the events as they pass by in an effort to orient the ear of the interested listener.

What we would like to do is find a middle road between these two extremes. If we are to be at all concerned with the act of listening to a musical composition, then we cannot overlook, as does Ligeti, what is heard in the course of its performance. On the other hand, unless we want to accept every work of music as an entity totally isolated from all other compositions, we ought to account for certain structural and organizational concepts which may be found in a wide variety of musical works. Thus, given a composition, C, we should like a model, C*, of which we may ask such questions as, "What are the most commonly heard motifs in C?", "Is there any relation between these two fragments?", or "In what ways is C similar to another composition, C'?"

These questions are, of course, of a secondary nature. The expression of our own basic question takes a bit of probing into our own Weltanschauung. We might possibly express it as follows: "What are the processes underlying musical structure?" In his Guidelines for Style Analysis, Jan [La Rue] dismisses the word "form," replacing it with the notion of "the growth process":

> The style-analytical view of musical form as a resultant and combining element requires

> a fresh, stimulating term to express the vitality and immediacy of a functional approach as well as to dissolve the rigidities suggested by the unfortunately static word "form." Happily the word "Growth" admirably fulfills these needs, since its connotations include both the feeling of expansive continuation so characteristic of music and also a parallel sense of achieving something permanent. . . If the Guidelines have accomplished anything thus far, they should have instilled a settled habit of regarding music first as a process of growth, then attempting to understand this growth by an analysis that fully reflects the character of musical flow.

However, for our purposes, it will be preferable to regard such "growth" as an interaction of several processes, rather than as a single, unified process.

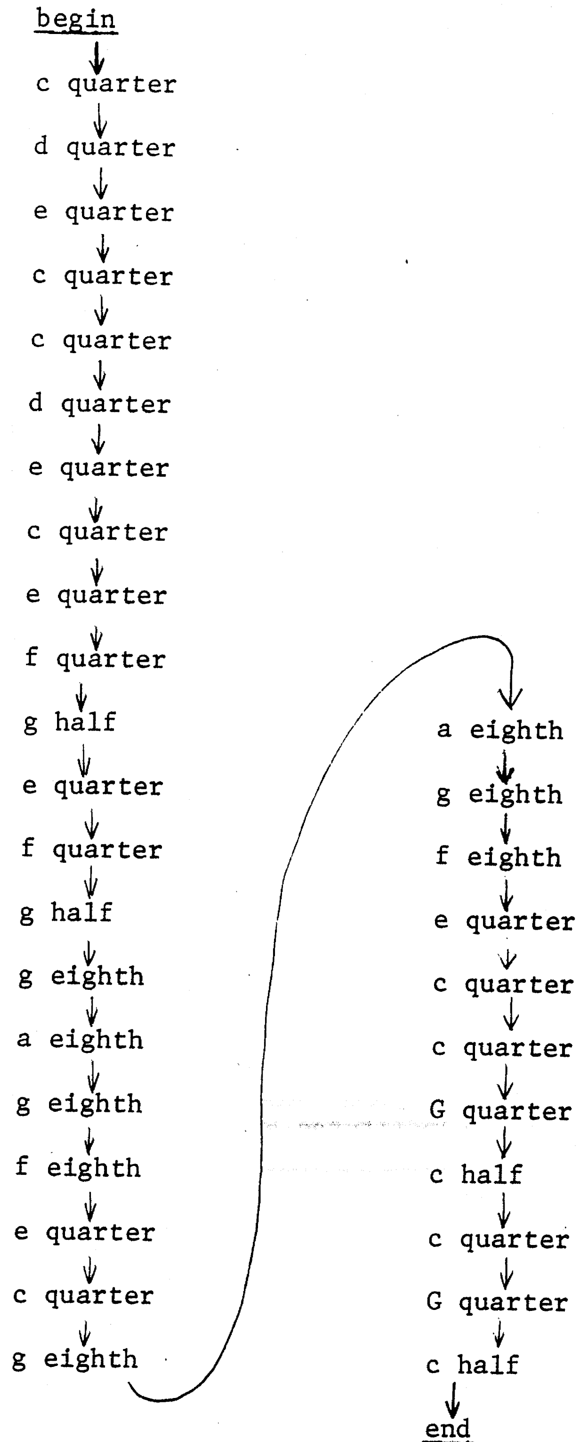
Hence, what we require is a formal representation of processes. However, this is basically what a computer program is. (See Knuth's definition of a computer program in his book, Fundamental Algorithms. [Knuth, 1968]) It is, therefore, reasonable to consider the design and implementation of musical models within the formalism of programming languages. The bulk of our work has been the development of such a language, EUTERPE, designed explicitly for this purpose. However, before we give any specific details, let us first consider a general example of how a piece of music may be modelled within the format of a computer language.

2. An Example

Consider the familiar round, "Frere Jacques:"

Example 2.1

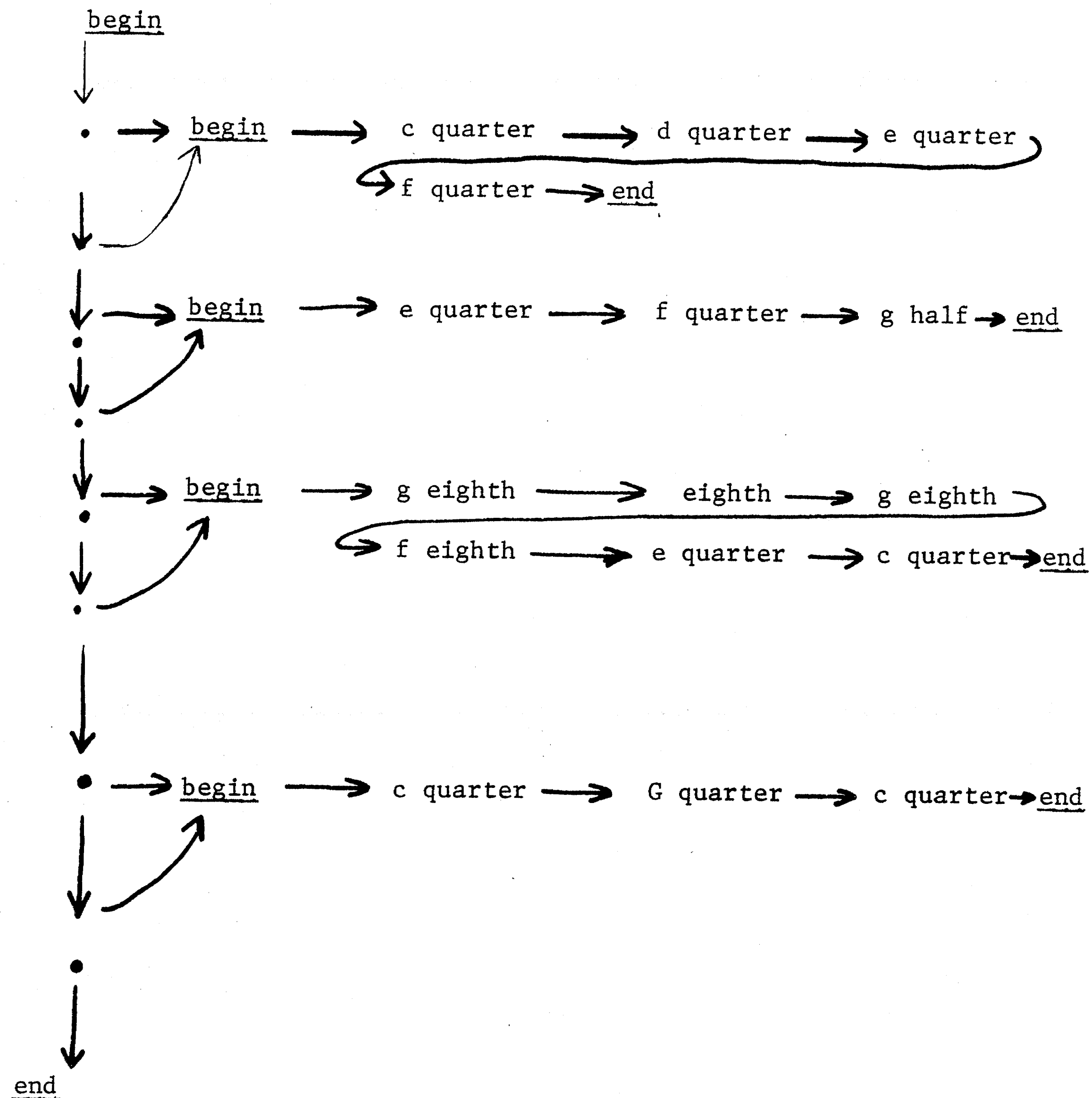
This score constitutes a symbolic model of the melody, but it says very little about structural organization. Its basic function is a representation in visual symbols of an auditory event. While this is a very elementary notion, our first task must be to translate these symbols into some format more closely related to a computer program. We might do this as in the following flow chart:


begin
c quarter
d quarter
e quarter
c quarter
c quarter
d quarter
e quarter
c quarter
e quarter
f quarter
g half
e quarter
f quarter
g half
g eighth
a eighth
g eighth
f eighth
e quarter
c quarter
g eighth
a eighth
g eighth
f eighth
e quarter
c quarter
c quarter
G quarter
c half
c quarter
G quarter
c half
end

Example 2.2

A flow chart, such as this one, is a diagrammatic abstraction of a computer program. It has a beginning (begin) and an ending (end); all other items in the diagram are specific events which may occur in the course of the program's execution. The order in which these events occur is specified by the arrows which connect them. Each event has two symbols: a letter of the alphabet and a word. These constitute a coding of those features of the music notation which we wish to represent, i.e. the pitch and duration of each note of the melody.

What more can we achieve other than a simple coding? One of the most obvious features of this melody is that every even-numbered measure is a repetition of its predecessor, and it might be useful to include this information in our model. In the following representation, we treat each of the four repeated measures as a separate entity, each of which is accessed twice by a main program which realizes the tune .


begin
begin
c quarter
d quarter
e quarter
f quarter
end
begin
e quarter
f quarter
g half
end
begin
g eighth
eighth
g eighth
f eighth
e quarter
c quarter
end
begin
c quarter
G quarter
c quarter
end
end

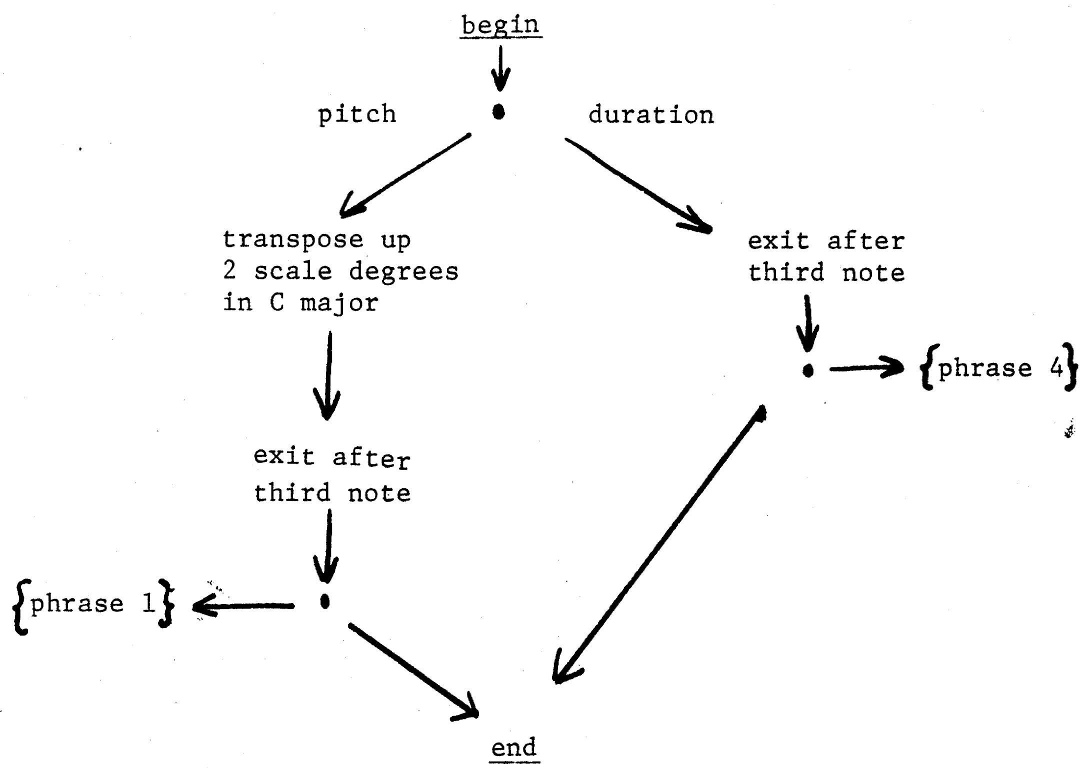
Example 2.3

In the terminology of computer languages, we would say that we have expressed the tune in terms of four subroutines, and the main program serves simply to outline the overall structure. [Knuth, 1968] defines subroutines as follows:

> When a certain task is to be performed at several different places in a program, it is usually undesirable to repeat the coding in each place. To avoid this situation, the coding (called a "subroutine") can be put into one place only, and a few extra instructions can be added to restart the outer program properly after the subroutine is finished. Transfer of control between subroutines and main programs is called "subroutine linkage."

Subroutine linkages may thus be used for the most basic modeling of redundancies. As we shall see, they may be applied not only to themes and phrases, but even to materials as small as simple motifs.

Now let us consider the contents of the phrases in more detail. For example, the rhythmic pattern of the second phrase is the same as that in the last phrase, while the melodic pattern is a truncation of that of the first phrase, only transposed up two scale degrees within the key of C major. If we express the determination of pitch and duration as two coordinated, but separate, processes, we can send the duration process to the fourth subroutine and the pitch process to the first subroutine, with a suitable pitch transposition. To represent the truncation within our formalism, we need the notion of an exit. This is an instruction which causes a subroutine to execute its end at some specified, premature time. In this particular example, we wish to enable the exit after three notes have sounded. Thus, we arrive at the following flow chart for the second phrase:



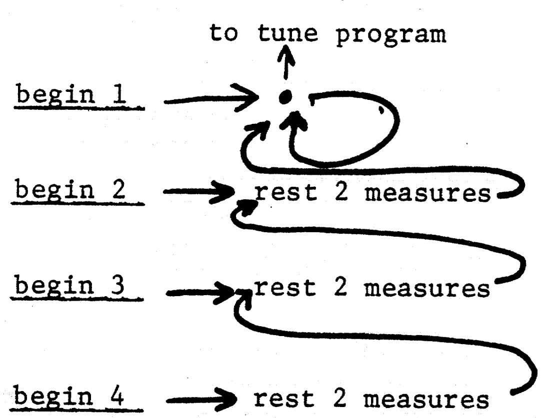
Example 2.4

(We have referred to the subroutines for the first and fourth phrases simply by the labels in the braces; they are the same as in Example 2.3.)

Notice that our flow chart for this particular subroutine has the same basic format as the flow chart for the main program, i.e.

it specifies a beginning, and ending, and a sequencing of specific events. Thus, a subroutine is basically a program, just like the main routine. The only reason it is subordinate is because it is accessed by a higher-level process. Furthermore, as is the case in this example, a subroutine may call <u>other</u> subroutines which are, for that computation, subordinate to it. The entire computation is thus a hierarchy of processes which depend on each other in a manner determined by the way they call each other.

Our representation of the tune has been monophonic, but the song "Frere Jacques" is actually a round. To specify the polyphonic structure we may define several programs in parallel which all access this tune program (hence, treating it as a subroutine) but do so at different times. Thus, if we wish to have four voices in the round, we would have the following:



Example 2.5

Our representation of the tune has been monophonic, but the song "Frere Jacques" is actually a round. To specify the polyphonic structure, we may define several programs in parallel which all access this tune program (hence, treating it as a subroutine) but do so at different times. Thus, if we wish to have four voices in the round, we would have the following: Notice that this last flow chart specifies no endings. Whenever the melody is completed, it simply begins again. Hence, the realization will never stop; it is an ideal round which will continue ad infinitum.

Let us summarize the features of this new model which we wish to substitute for our score. Like a score, our model is a symbolic representation of those events which constitute a performance of the composition. We may say that an actual performance is a realization of the model. However, in addition to representing those events which form the composition, the model provides a symbolic representation of these events as a configuration. [Langer] writes of "the power of language to embody concepts not only of things, but of things in combination, or situations."

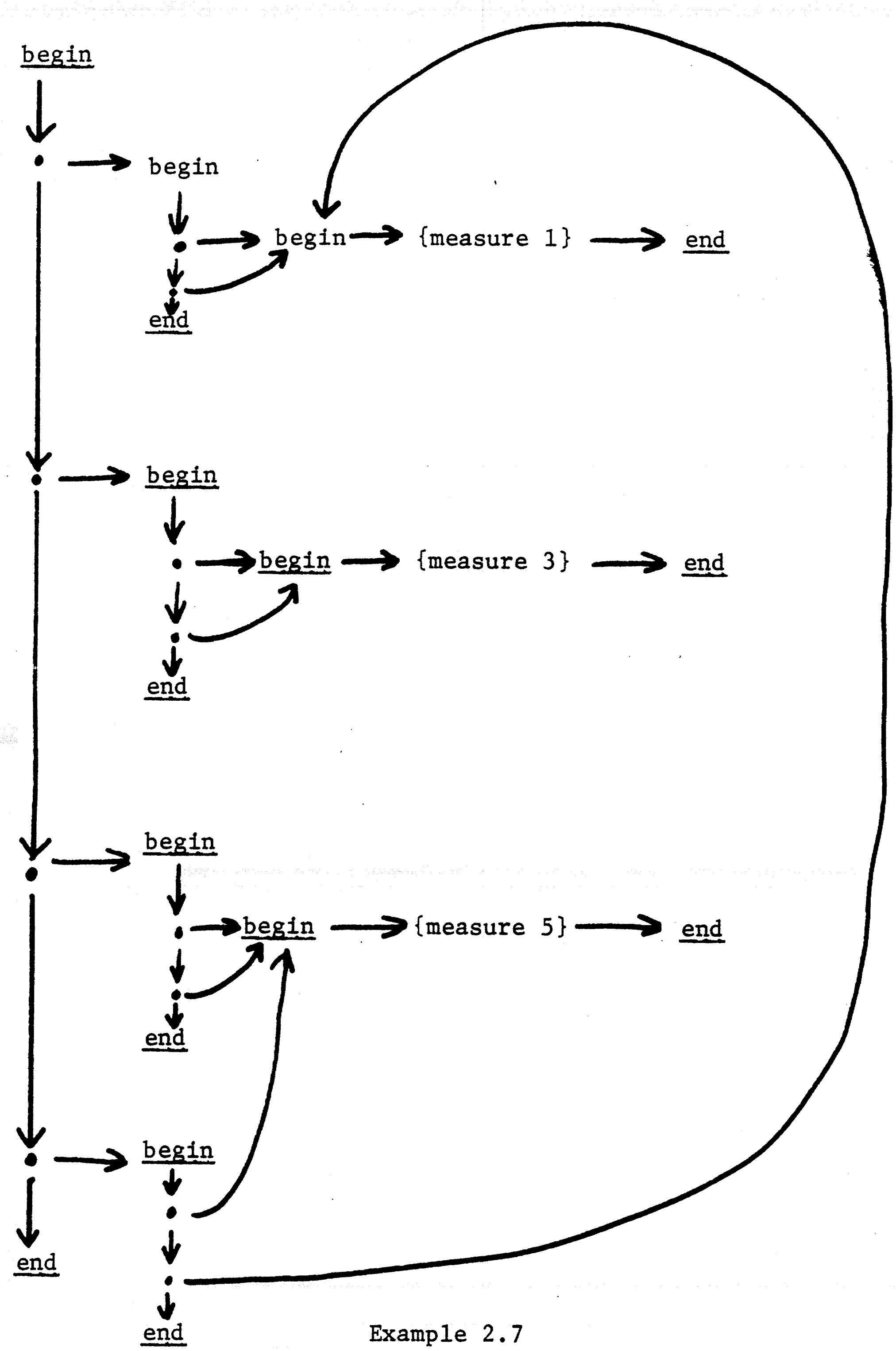
Similarly, we have this power in our model. Furthermore, these configurations are expressed as a hierarchy. Different relationships occur at different levels, some of which depend on each other, others of which are independent. Such a hierarchy attaches a priority to each element of the model and thus allows us to consider the <u>generality</u> of the model. For example, "Three Blind Mice" is another four-part round which is very similar to "Frère Jacques." Is this particular model of any use in the modeling of "Three Blind Mice?"

At the topmost level — that of the actual canonic formation -- it is perfectly compatible. The second voice enters two measures after the first; the third, two measures after the second; and the fourth, two measures after the third. As far as actual content is concerned, however, the structures are somewhat different. For purposes of argument, let us consider the following, slightly simplified, version of "Three Blind Mice."

Example 2.6

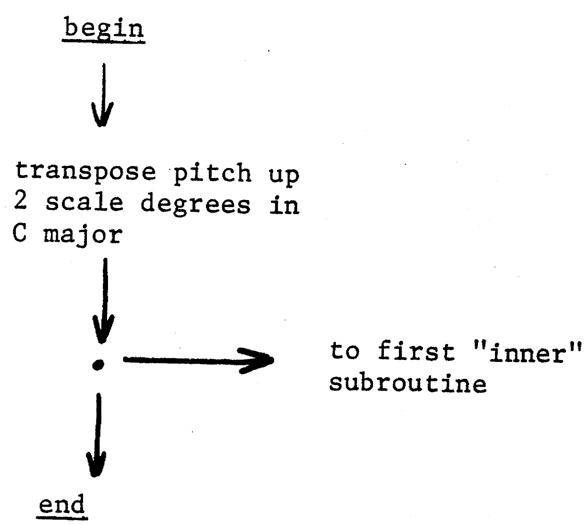
The first six measures follow the "Frere Jacques" model very closely -- each even-numbered measure repeats its predecessor. However, the last two measures do not conform to this model. The penultimate measure is another repetition of the fifth measure, and the last measure repeats the first.

This poses no serious problem. The fact that measures are repeated means that we can still store them as subroutines. However, because the repetition scheme is different, we require a different procedure of subroutine calls. Let us try to design a model for "Three Blind Mice" which is somewhat more general -- general enough to perhaps accomodate "Frere Jacques" as well.


begin
begin
begin
{measure 1}
end
end
begin
begin
{measure 3}
end
end
begin
begin
{measure 5}
end
end
begin
end
end

Example 2.7

In this example we have not written out the notes of the individual measures but have again simply referenced them in braces. Actually, the second of these inner subroutines could be written to call the first (using a technique similar to, but simpler than, the device used in the "Frere Jacques" example):



Example 2.8

However, we are more concerned with the compatibility of "Frere Jacques" with the higher-level structure. The main routine calls four subroutines; this much is compatible. The first three of these subroutines each call a thematic subroutine twice; this is also compatible. The last subroutine also has two subroutine calls, but these are to different thematic subroutines. If these are modified to call a fourth thematic subroutine twice, we have a model of "Frere Jacques." Thus, the two programs are alike to a depth of two levels of the subroutine tree, as well as sharing the same structure of voice entries in the round.

Hence, we have established a symbolic formalism wherein we may represent the similarities and differences between two tunes. With a few modifications, we could include other canons within our model; and the modifications which would be necessary would tell us about the structural relationships among these new additions. Thus, we see that in a practical sense, we have analysed a small corpus of music in the format of computer programs. We are no longer concerned simply with the notes from which a piece is composed but with a structural abstraction which is more general than a description of a single composition. Now we consider the matter of putting an actual computer system to such an application.

3. The Role of EUTERPE

EUTERPE first appeared in 1967 as "a computer language for the expression of musical ideas" ([Smoliar]). It was implemented to realize, in real time, musical scores, presented in the format of the language, using an elementary sound-synthesis program for output. As such, a composer could experiment with music in up to six parts of such complexity that it could not be easily realized at a piano keyboard.

At the time of its implementation, EUTERPE was used primarily for coding musical notation. While elementary features such as transposition and subroutine linkages were part of the language, it was not until the summer of 1970 that the author discovered that such features, with a few modifications, could provide many valuable "short cuts" in representing certain conventional musical forms. At that time, the language was expanded to its current form.

EUTERPE now has three basic uses. It still has the ability to play the music it represents; however, the primary significance of the language is that it provides a formalism within which models such as those discussed in the preceding section, may be designed and implemented. This, in turn, may eventually lead to a more formal approach to the understanding of music.

The understanding of music is an extremely vague notion, and there is very little which we can say about it in relation to EUTERPE at the present time. As we saw in the preceding section, we have established a technique of modeling which may be generalized beyond the sphere of a single musical composition. Any more formal work in the understanding of music will probably have to involve a language for manipulating models, such as C. [Hewitt]'s PLANNER. However, we suspect that EUTERPE will prove valuable in providing a data base with which such a higher-level language can work.

In the following chapter we shall offer a full presentation of EUTERPE. We anticipate that this will appeal to three fields of interest. First of all, there is the value of EUTERPE to the musicologist. Such a reader will probably be concerned mainly with the formal principles of the language and their relationship to musical structures. A composer, on the other hand, may have a more practical interest; he might want to know how to go about using such a system if he has it at his disposal. Finally, the computer scientist will probably be interested in the design and implementation of the language for its own sake. Hence, we shall also provide the details of this aspect for such interested persons. We shall try to organize our material so that the reader interested in only one of these approaches may safely bypass the other two.

Chapter 1

The EUTERPE System

1.1. Musicological Foundations

1.1.1. The Procedural Approach

As we stated in the Introduction, our primary goal is to establish a formalism for the modeling of music, and we intend to approach this goal through a consideration of the processes underlying musical structure. We have already cited Jan La Rue's consideration of musical form as a growth process, but this is really only half the story. La Rue still treats the musical score as a source of data which undergoes a series of transformations -- a reasonable approach within the confines of verbal analysis.

If we are working within the realm of computer languages, we may also choose a suitable representation of music as data; but we may also regard scores as programs, i.e. processes. The representation of data as processes has been used by Winograd very successfully in his language-understanding system ([Winograd, 1971]); and we shall see that it will also serve our musical purposes. This is, in fact, the approach we took in the flow chart in Example 2.2, where each note of the "Frere Jacques" tune was represented as a separate event (or, in computer terminology, instruction). The execution of the program consists of the execution of these instructions in their defined sequence, which is simply the playing of the notes in their proper order.

Computer instructions which specify notes will be called note words; those which are not, we shall call control instuctions. The

The major distinction between note words and control instructions is that the former are defined for some finite period of time while the latter are assumed to take place instantaneously. However, it is the control instructions which specify <u>how</u> the note words are to be executed; and thus the control instructions will form the basis of our abstract style analysis.

In this section we shall first establish some basic conventions of format. Then we shall consider the design of note words and the repertoire of control instructions. We shall describe these control instructions primarily on the basis of their musicological foundations and will defer a more specific definition to Section 1.2.

1.1.2 Note Words

In Example 2.2 note words were specified by two elements of data -- a specification of <u>pitch</u> and a specification of <u>duration</u>. This is the abstraction of conventional five-line staff notation. The level on the staff determines the pitch, and the shape of the note represents its duration. (When we actually define the implementation of note words, we shall account for a third factor, which we call articulation, which will specify a portion of a note's duration to be silenced before the next note is sounded; however, we need not consider such matters in this section.)

Our approach to pitch will be a microtonal one. We shall regard the octave as being divided into 72 equal units. (In this dissertation we shall call this smallest unit a <u>microtone</u>; more accurately, it is a twelfth-tone, i.e. a sixth of a semitone.) This apportionment contains the conventional chromatic apportionment of the octave into twelve equally-tempered parts (semitones); but each semitone is further tempered into six equal microtones.

We shall introduce the symbolism used in the EUTERPE language so that we may avoid the representation used in Example 2.2. Pitch is represented by a string of at least two symbols, each separated by a space. There is one exception to this rule, however, in that we shall represent rests by the single symbol R. (A rest is essentially the null pitch specification.)

The first of these symbols is a single letter specifying an octave. This octave encompasses the gamut from C up to B (along with any chromatic alterations by accidentals). The symbol is one of the following letters: H, I, J, K, L, M, N. K is the octave which begins at middle C.

The second symbol specifies the location within the given octave. Conforming to standard notation, this symbol is one of the following letters: C, D, E, F, G, A, B. Semitonal and microtonal chromatics are atained by adding extra symbols. FL and SH designate FLat and SHarp, respectively. Q denotes inflection by a Quarter-tone (i.e. three microtones) upward, while P, the corresponding inflection downward. $ and % indicate, respectively, raising and lowering by a sixth-tone (two microtones); and T and U stand for raising and lowering a single microtone. Since the resulting 72-tone octave is even-tempered, a note word containing several chromatic symbols will designate the proper enharmonic, i.e. C double-sharp (C SH SH) is identical to D and A U U is the same as A %.

By way of example, the following pitches specify an A melodic minor scale beginning and ending on the A below middle C: J A, J B, K C, K D, K E, K F SH, K G SH, K A, K G, K F, K E, K D, K C, J B, J A. The following is a microtonal passage using the notation of Ezra Sims (as specified in his article in the Harvard Dictionary of Music [Apel, 1969]), along with the corresponding pitch symbols in EUTERPE ([Sims]):

Example 1.1.2.1

We shall represent duration by a single symbol without spaces. 1T will denote a whole note, 2T a half note, 4T a quarter note, and so on. (Theoretically, we may continue indefinitely; however, in the actual EUTERPE system, 32T is the shortest of these durations.) Concatenation of a "3" to the right end of this symbol causes the duration to be interpreted as if it were under a triplet bracket (i.e. 8T3 is a triplet eighth note, a note whose duration is two-thirds that of 8T). Replacing the T by a D causes the note to be interpreted as if it were dotted (i.e. 2D is a dotted half note with duration three-halves that of 2T).

Conventional staff notation does not provide an absolute designation of pitch. The actual pitches are not known unless one knows the nature of the instrument playing them. A B-flat clarinet and an oboe, both playing the same line, will sound different pitches. Likewise, duration is related to some overall tempo indication which, in its most specific form, is expressed as a metronome marking.

We shall say that pitch and duration are defined relative to <u>pitch</u> and <u>duration parameters</u>, respectively. These parameters provide the necessary information for a specific realization of the note words. For example, in Example 2.4, the command to "transpose up 2 scale degrees in C major" did not specifically change the note words in the phrase subroutine but simple effected an alteration of the parameter which provided a specific interpretation of the note words.

Modification of parameters is one of the two basic tasks of control instructions. The other is transfer of control. We shall consider this latter aspect first, as it is more fundamental to the functioning of our system.

1.1.3. Transfer of Control

The main function of a flow diagram is that it defines the sequence of a given set of events. When certain parts of this sequence are redundant, we would like to be able to express them as such. Through devices of control transfer, it is possible to access a given event more than once and from more than one point in the sequence. There are two basic approaches to transfer of control: simple transfers and subroutine linkages. We have already seen examples of both in the Introduction.

A subroutine, as has already been observed, is actually a program in its own right. In our diagrammatic abstractions in the Introduction, subroutines had their own begin and end points. They were accessed by an arrow from a dot to the begin mark; and when the end was reached, the program would pick up where it left off at the dot. The only thing that made a subroutine subordinate was the knowledge that it was called by a program at a higher level; yet even a subroutine was capable of having subroutines of its own. The structure of subroutine calls is thus hierarchical, and it provides a more formal approach to what [La Rue] loosely calls "dimensions." Passages of smaller dimensions would be such elements as motifs and phrases, i.e. programs calling few subroutines, which would often be called as subroutines.

The larger dimensions encompass entire works or even groups of works, and these would correspond to top-level programs.

However, not all musical structures are hierarchical in nature. In Example 2.5, our four-voice representation of "Frere Jacques," the second voice does not "call" the first voice's program with a delay of two measures; it simple repeats it. Likewise, the third voice repeats the program of the second, again with a delay of two measures; and similarly for the fourth voice. The program for the first voice is not subordinate to that for the second, and we do not employ a subroutine linkage. In this case the transfer of control is accomplished to a simple transfer.

A simple transfer does not effect an entry to a subordinate program which has its own beginning and ending; it simply shifts the process to some remote sequence of events. In this case it causes the second voice to enter the program for the first voice. Moreover, the loop after the linkage to the tune program is also a simple transfer; it specifies that after the subroutine for the tune has been completed, it should be called again. Thus, the arrow points back to the location of the subroutine entry.

In this example the four voices of the round are defined by four independent programs which happen to share much of the same code. Alternatively, we may have a situation in which the different voices are processing different code, but the control of one voice is to be influenced by another. Willi Apel applies this principle in his interpretation of two-voice melismatic organa, in which the upper voice (duplum) sings lengthy, elaborate passages against sustained tones in the lower voice.

> Obviously, the singer of the duplum will take the lead, and the singer of the tenor will follow suit, beginning simultaneously with the first note, and changing to the second note somewhere in the middle of the melisma (always with the first note of a group sign, of course), where a suitable consonance occurs. ([Apel, 1953])

For example , here is Apel's transcription of an Alleluia from the Codex Calixtinus ([Apel, 1953]):

Example 1.1.3.1

The upper voice may be represented simply as a sequence of pitches as follows (we assign arbitrarily the duration of a quarter note):

begin
↓
K D 4T
↓
K D 4T
↓
K C 4T
↓
K D 4T
↓
K E 4T
↓
K D 4T
↓
K C 4T
↓
etc.

Example 1.1.3.2

The lower voice, on the other hand, may be represented as a series of loops, each of which sustain the tone which is being held and which are cued in by the upper voice:

```
begin
  ↓
tra 2   →  J D 4T  (loop)
  ↓
K D 4T
  ↓
K D 4T
  ↓
K C 4T
  ↓
tra 2   →  J A 4T  (loop)
  ↓
K D 4T
  ↓
K E 4T
  ↓
K D 4T
  ↓
K C 4T
  ↓
etc.
```

Example 1.1.3.3

While this representation will realize the score properly, it has one major disadvantage. The lower voice is not being represented as a continuous <u>line</u> (i.e. part) but rather as a series of isolated events which are triggered by the upper voice. It would be more desirable to specify the tenor in a sequential form similar to that of the duplum, and to restrict the upper voice's role to one of simply <u>advancing</u> the lower voice through its sequence. To accomplish this, we may use the notions of <u>indexing</u> and <u>indirect addressing</u>.

Let us begin with the notes for the tenor arranged in their proper sequence:

```
J D 4T
  ↓
J A 4T
  ↓
J G 4T
  ↓
J A 4T
  ↓
J G 4T
  ↓
etc.
```

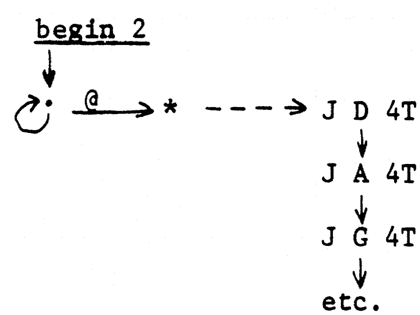
Example 1.1.3.4

Next we specify a <u>pointer</u> (which we shall indicate by an asterisk) which will simply indicate the first note of this sequence.

```
* ---→ J D 4T
         ↓
       J A 4T
         ↓
       J G 4T
         ↓
       etc.
```

Example 1.1.3.5

Now we may specify the repetition of a tone in the lower voice as follows:



Example 1.1.3.6

The sign @ denotes <u>indirect addressing</u>; it specifies execution of the location indicated by the pointer *, rather than a direct transfer to *.

However, this will not help us advance through the sequence of pitches. To do this we need to attach an <u>index</u> to the pointer. The index takes the location indicated by the pointer and <u>increments</u> it by some designated number of steps. Thus, we may denote the indexing of the pointer * by the index I as follows:

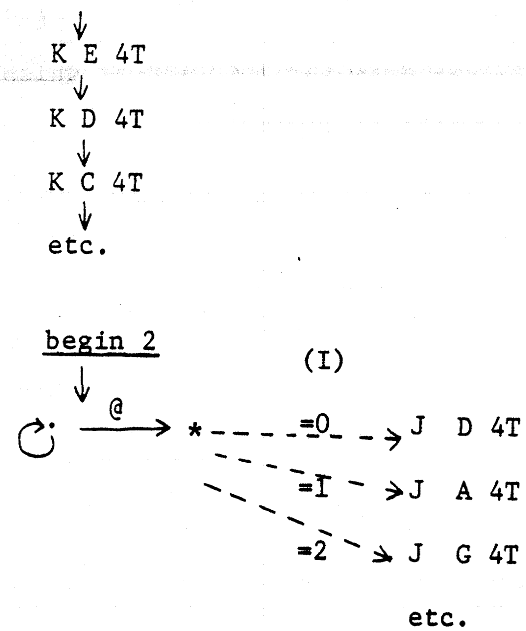
```
           (I)
* ---→ J D 4T
         ↓
       J A 4T
         ↓
       J G 4T
         ↓
       etc.
```

Example 1.1.3.7

When the index is set to zero, the pointer designates the first note (J D); if it is set to one, the next note (J A) is indicated; and so forth.

In this case all the upper voice needs to do is increment the index at the appropriate times. The two voices now have individual programs, but the upper voice controls the lower voice by altering its index:

```
begin 1
  ↓
I←0   (set I to zero)
  ↓
K D 4T
  ↓
K D 4T
  ↓
K C 4T
  ↓
I←I+1 (increment I by one)
  ↓
K D 4T
  ↓
```



Example 1.1.3.8

Thus we see that transfer of control may extend from one voice to another in two ways. In our first example, the voice taking control simply gave a command which was to be executed by another voice. This is the basic mechanism of intervoice control in EUTERPE, and it applies to the specification of parameters as well as to sequencing of events. In the second technique the commanding voice altered a location accessed by the affected voice. In theory, this is really the same as in the first example; however, this latter case operates at the level of the program, while the former operates at the level of the processor.

In these examples we have considered intervoice control primarily with regard to simple transfers. However, it is easy to see how they may also be applied to subroutine calls. Furthermore, intervoice control may influence not only the time at which a voice <u>enters</u> a subroutine, but also the location at which it leaves the subroutine. Thus, through intervoice control, a voice may leave a subroutine before it has arrived at the end mark, owing to a command given by a remote voice.

Such intervoice transfer of control constitutes an interruption of the process being executed by the affected voice. If this interruption occurs between instructions, then it is easy to return from the subroutine to where the main program left off. However, as we shall see in Section 1.2.3, such an intervoice call might interrupt the sounding of a note; in which case the return is not as simple. We shall discuss the mechanism of such an interruption in this future section.

1.1.4. Pitch and Transposition Parameters

As we mentioned in Section 1.1.2, the symbolic representations of pitch and duration provided in note words establish definitions relative to parameters. For example, in the case of pitch, the fundamental parameter specifies that pitch which sounds when middle C is notated. (All other notated pitches are, of course, altered accordingly; this is the general principle of transposing instruments.) We shall also establish other parameters which will be employed in the definition of pitch.

There are two ways in which the fundamental pitch parameter may be defined. It may be defined absolutely, that is, by specifying actual pitch which it is to take as its value, or <u>relatively</u>, specifying an interval, ascending or descending, from its current value which determines its next value. For example, were we to transcribe a B-flat clarinet part, we might specify that the pitch parameter be set to J B FL (the B-flat below middle C); this would be an example of an absolute definition. Alternatively, we may describe the B-flat clarinet part as one which sounds all its pitches a whole tone lower than they are notated. Hence, we need only specify that the pitch parameter be displaced down a whole tone (assuming the initial value to be K C); this would be a relative definition.

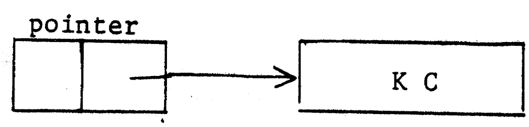
Thus far, we have conveyed the impression that the fundamental pitch parameter is the conents of some distinguished location in the processor. This is not quite the case, as all parameters are stored on <u>stacks</u>. A stack is the most fundamental means of altering data and keeping track of the alterations so that they may be later undone, and this will prove to be very valuable to us.

Knuth defines a stack as "a linear list for which all insertions and deletions (and usually all accesses) are made at one end of the list" ([Knuth, 1968]). We shall call this specified end the <u>top</u> of the stack, and it will be our only point of access. Thus, pitch parameters will be stored on a stack whose top contains the current value of the pitch parameter.

We are using a stack rather than simply a distinguished location because we intend to insert and delete information. In fact, stacks will be built and manipulated according to the following rule of thumb:

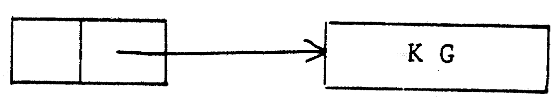
Initially, a stack contains only one value which is at its top; when a parameter is defined absolutely, this value at the top is replaced by the value prescribed by the definition; when it is defined relatively, the new value is inserted into the stack, becoming the new top; the earlier value may be restored simply by specifying that the top of the stack be deleted.

Let us consider an example of such stack manipulation. Initially, the pitch parameter is defined so that notated middle C is interpreted to sound as middle C. This means that the pointer designates a word whose contents is K C:



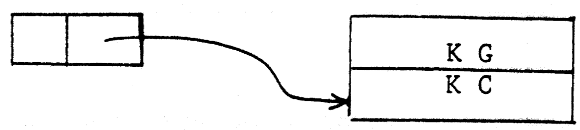
Example 1.1.4.1

An absolute alteration of the pitch parameter would entail, for example, an instruction stating that notated middle C is to sound as G above middle C. In this case, the word indicated by the pointer would be modified as follows:



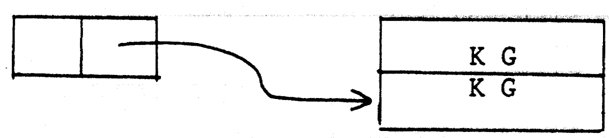
Example 1.1.4.2

From this position, we can reutrn to the original situation by another absolute definition, re-establishing middle C to sound as middle C and returning to the conditions of Example 1.1.4.1, or we can invoke a relative alteration of the parameter specifying that all pitches are to be transposed down a perfect fifth. This latter approach would add a new value to the stack:



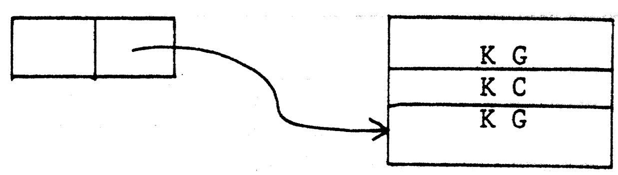
Example 1.1.4.3

Now we have three ways to have middle C sound as G above middle C from this position. If we use absolute definition to specify that middle C sounds as G above middle C, we obtain the following:



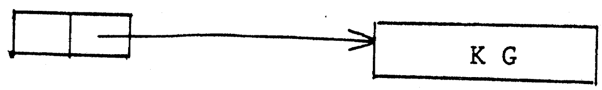
Example 1.1.4.4

If we use a relative definition and specify that all pitches are to be transposed up a perfect fifth, the stack would assume the following form:



Example 1.1.4.5

However, the simplest plan of action would be to merely delete the top element of the stack:



Example 1.1.4.6

(We shall occasionally speak of "pushing down" and "popping up" parameters when such alterations take place. These terms are a bit counter-intuitive, but they are also standard usage. Knuth has offered the following parenthetic apologia:

> People often say they push down an item onto a stack, and pop up the stack when the top item is deleted. This terminology comes from an analogy with the stack of plates often found in cafeterias, or with stacks of cards in some punched-card devices. The brevity of the words "push" and "pop" has its advantages, but these terms falsely imply a motion of the whole list within computer memory. Nothing is physically pushed down; items are added onto the top, as in haystacks or stacks of boxes. ([Knuth, 1968])

This pitch parameter thus defines strict intervallic displacement of the pitches designated by the symbols in a note word. However, this is not the only type of transposition we might wish to represent. In tonal music transposition is more often expressed in terms of scale degrees with respect to a given tonality. We shall account for such transpositions by a separate parameter which we shall call the transposition parameter.

Like the pitch parameter, the transposition parameter will be represented as a value at the top of the stack. This value may be assigned by an absolute definition which designates the number of scale degrees, up or down, of transposition. It may also be defined relatively by specifying an integer (positive or negative) to be added to the parameter; as with the pitch parameter, this will cause the insertion of a new value at the top of the stack. Finally, we may specify the deletion of the top of the stack.

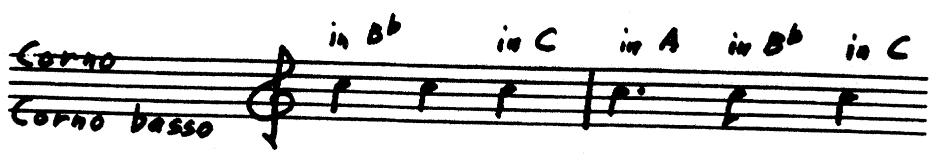
Of course, tonal transposition can only be defined with respect to a given tonality. For our purposes, the establishment of a tonality is determined by the ascending and descending forms of a particular scale. Such a point of view essentially reflects [Schoenberg]'s fundamental notion of tonality:

> A tonality is expressed by the exclusive use of all its tones. A scale (or part of one) and a certain order of the harmonies affirm it more definitely.

A scale consists of a <u>block</u> of data, rather than a single, distinguished location. However, scales, too, may be defined on either an absolute or a relative basis. In the former case, the entire block is written out at the current top of the stack, while in the latter, it is pushed down and a new block is created. A relative definition may be expressed by an integer from 1 to 7, i.e. a scale degree. (Conventionally, this argument would be given in Roman numerals; but this is an unnecessary inconvenience in an actual computer system.)

With two parameters, the pitch of a note word might be interpreted in two possible ways, depending on which was applied first. For example, suppose the current value of the pitch parameter is K F, the value of the transposition parameter is 3, the tonality is that of C major, and the pitch in the note word being processed is K C. If the pitch parameter is enable first, followed by an application of the transposition parameter, the resulting ptich is K B (first up a perfect fourth and then up three scale degrees). On the other hand, if the transposition parameter is processed first, the result is K B FL (first up three scale degrees to F and then up a perfect fourth). We shall establish the convention of the former interpretation -- an application of the pitch parameter followed by an application of the transposition parameter.

Actually, there are occasions when these two parameter stacks are somewhat less than adequate. Consider the following example, inspired by Richard Strauss and posed, after Cecil Forsyth, by Ezra [Sims]:



Example 1.1.4.7

When this rather complex set of crookings for horn and bass horn is realized, the following score obtains:

Example 1.1.4.8

Let us consider how this situation might be modeled.

First of all we shall attempt the representation in a program read by both voices which uses only the pitch parameter stacks. (In this example when we say "set" we shall mean an absoltue definition of the parameter, and when we say "alter" we shall mean a relative definition.

<u>begin 1</u> ↘   ↙ <u>begin 2</u>

set pitch parameter to J B FL

↓

alter pitch parameter for 2
down one octave

↓

L C 4T

↓

L C 4T

↓

alter pitch parameter up a
whole tone

↓

LC 4T

↓

alter pitch parameter down a
minor third

↓

L C 4D

↓

alter pitch parameter up a
semitone

↓

L C 8T

↓

alter pitch parameter up a
whole tone
↓
L C 4T
↓
<u>end</u>

Example 1.1.4.9

All parametric definitions apply to both voices, except for the second instruction which affects only the second voice.

Alternatively, we could design a program using the transposition parameter stacks as follows:

<u>begin 1</u> ↘ ↙ <u>begin 2</u>

set key to B-flat major
↓
set pitch parameter for 2 to J C
↓
set transposition parameter to -1
↓
L C 4T
↓
L C 4T
↓
set transposition parameter to 0
↓
L C 4T
↓
set transposition parameter to -2
↓
L C 4D
↓
set transposition parameter to -1
↓
L C 8T
↓
set transposition parameter to 0
↓
L C 4T
↓
<u>end</u>

Example 1.1.4.10

This example makes somewhat more efficient use of stack space than does Example 1.1.4.9, and this is achieved by employing the transposition parameter as well as the pitch parameter. However, the fundamental difference between these two programs is that in the latter, all transpositions are absolutely defined with respect to a fixed point, while in the former, each transposition is expressed in terms of the distance from the note just sounded.

Were we to attempt this technique using the pitch parameter:

```
begin 1                    begin 2
   ↘                      ↙
   set pitch parameter to J B FL
   ↓
   alter pitch parameter for 2
   down one octave
   ↓
   L C 4T
   ↓
   L C 4T
   ↓
   set pitch parameter to K C
   ↓
   L C 4T
   ↓
   set pitch parameter to J A
   ↓
   L C 4D
   ↓
   set pitch parameter to J B FL
   ↓
   L C 8T
   ↓
   set pitch parameter to K C
   ↓
   L C 4T
   ↓
   end
```

Example 1.1.4.11

It would sound as follows:

Example 1.1.4.12

The absolute definition of the pitch parameter following the second L C 4T destroys the information that the second voice is transposed an octave below the first by nature of the absolute definition. Thus, from that point on, the two voices sound identical parts. The advantage to Example 1.1.4.10 is that the pitch parameter and transposition parameter are on independent stacks, so that absolute definition of the transposition parameter does not affect the octave transposition in the second voice.

Now consider a similar passage in which a definition in terms of transposition by scale degrees is impractical, or even impossible (e.g. an atonal, non-diatonic passage). If the composer has conceived of this passage as a sequence of intervals, then he may use a program in the form of Example 1.1.4.9; if he is thinking in terms of pitches all at a relative distance from a fixed point, this technique is unsuitable.

This example is admittedly somewhat contrived; but the musical distinction we are considering actually comes into play in studies of twelve-tone music. The twelve-tone row may be regarded in two possible ways, analogous to our two programming examples.

In many analyses of twelve-tone music, the pitches of the twelve-tone row are written out in their specified order; since these pitches may be stated on any octave, Milton [Babbitt] refers to such a representation as an ordered set of "pitch classes." Babbitt's system is similar tothe approach of our latter example, defining all pitch classes in terms of their distance from the first note in the series.

Alternatively, the series may be defined by the sequence of intervals between its successive notes. These intervals can also be grouped into equivalence classes modulo octave transposition; and this approach has been studied by Stefan [Bauer-Mengelberg] and Melvin Ferentz, who used it to investigate a row by Alban Berg in which the eleven interval classes were all different. Such an approach would correspond with that of Example 1.1.4.9.

Thus, we see that it would be to our advantage to have another stack of secondary pitch parameters, which, together with an effective procedure, computes pitch in terms of the pitch field of the note word the pitch parameter, and this new, secondary pitch parameter. We shall define this parameter to be an interval, and like all other parameters, it may be specified relative to its current value or absolutely. The procedure of interpretation will be as follows: first, the pitch parameter is applied to the pitch field; then the resultant pitch is shifted by the interval designated by the secondary pitch parameter. This secondary parameter will be assumed to be zero unless otherwise stated. Now we may represent Example 1.1.4.7 as follows:

```
begin 1                    begin 2
     ↘                    ↙
      set pitch parameter to J B FL
               ↓
      set secondary pitch parameter for
      2 to one octave down
               ↓
      L C 4T
               ↓
      L C 4T
               ↓
      set pitch parameter to K C
               ↓
      L C 4T
               ↓
      set pitch parameter to J A
               ↓
      L C 4D
               ↓
      set pitch parameter to J B FL
               ↓
      L C 8T
               ↓
      set pitch parameter to K C
               ↓
      L C 4T
           ↓
      end
```

Example 1.1.4.13.

Clearly, we may also consider the possibility of secondary transposition parameters, tertiary parameters, and so on. We might even be able to concoct reasonable examples which would demonstrate a need for such features. However, the current implementation of EUTERPE doesn't even have a separate stack for the secondary parameters. Instead, it has a feature which is almost as effective which we shall discuss in more detail in Section 1.3.3.

1.1.5 Duration Parameters

The specification of durations is somewhat more complex than that of pitch. In the first place, pitches are restricted to a discrete set of values while the range of durations forms a continuum. Secondly, alterations in pitch are treated in terms of linear shifts along a microtonal keyboard, while alterations in duration are necessarily treated in terms of multiplicative relations. In fact, it is the potentially infinite compounding of such relations that brings about the continuum of possible durations. A remarkable demonstration of this may be found in one of Conlen Nancarrow's rhythm studies for player piano which explores the rhytmic ratio of $2/\sqrt{2}$. (This is easy enough to achieve on a piano roll since the ratio is constructable with ruler and compass.)

In terms of their frequencies, pitches also exhibit such multiplicative relations. However, equal temperament reduces these relations to the additive relations of a logarithmic scale. No such temperament exists for durations in standard notation. On the contrary, rhythmic notation is inherently multiplicative, stemming from successive binary divisions of a large unit or (in the opposite direction) multiplications of a small one ([Apel, 1969]). Groupettes can be accounted for by more complicated rational multiples and fractions.

Like our pitch parameters, we shall establish a duration parameter which is stored on a stack and which admits of an absolute and a relative definition. The absolute definition of the duration parameter will be modeled after standard metronomic indications, which generally have the following form ([Apel, 1969]):

♩ = 60.

Example 1.1.5.1

This notation defines a tempo wherin 60 quarter notes fill the duration of one minute, i.e. one quarter note per second. Any note shape may appear as the left-hand argument, and the number specifies the number of occurences of that note shape required to fill the duration of a minute.

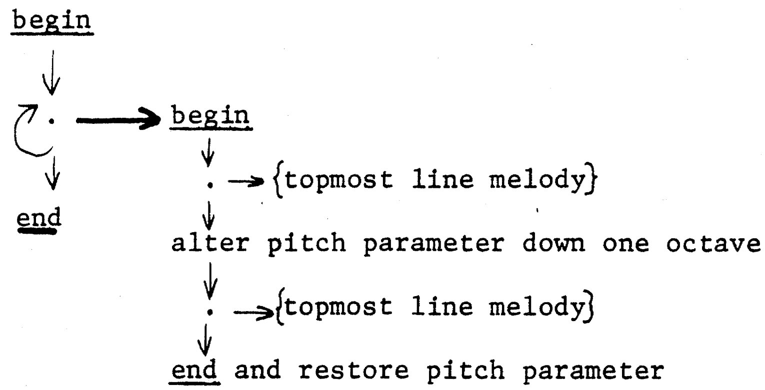
Relative definition of durations is accomplished by specifying their multiplicative relation. This involves an element of ambiguity. Is a given passage twice as fast as its predecessor, or is it half as slow? Rather than arbitrarily choosing one of these possible approaches, we consider another alternative. If a passage is twice as fast as its predecessor, then in the new section, two rhythmic units occupy the same durational space as one rhythmic unit in the old one. This may be called a transofrmation of "two in the space of one." In general, a transformation of "x in the space of y" causes x new rhytmic units to occupy the same duration as y old ones.

The only restriction we shall impose in our theory is that x and y both be integers. This makes a rhythmic ratio such as Nancarrow's impossible to realize unless one allows an infinite number of definitions. While this is conceivable in theory, it is impossible in practice, and we shall have to content ourselves with rational approximations in such circumstances. There are other discrepancies between theory and practice which we shall also have to consider. For example, in theory there is no smallest, indivisible unit of time, while in a practical realization on a computer system, there must obviously be one. However, we shall defer such matters to Section 1.3.

1.1.6 Stack Manipulations

Not only are stacks useful in storing parameters, but they are also valuable in keeping track of subroutine calls. Every program is equipped with a pointer, called a program counter, which keeps track of which instruction is being processed. When the program calls a subroutine, the current value of this pointer is pushed down on a stack; and the beginning of the subroutine is placed at the top of the stack. This new pointer traces through the subroutine until it is completed. At this point, the top of the stack is deleted, restoring the value of the pointer at the time the subroutine was called; and the main program picks up "where it left off."

We shall encounter occasions in which a subroutine will specify certain parametric modifications which will want to occur only within the body of that subroutine. For example, in Erik Satie's Vexations the topmost line is played once and then repeated an octave lower ([Dinwiddie]). These two renditions constitute a single playing of the score, a performance of which consists in 840 repetitions. We might represent this one voice by the following model:



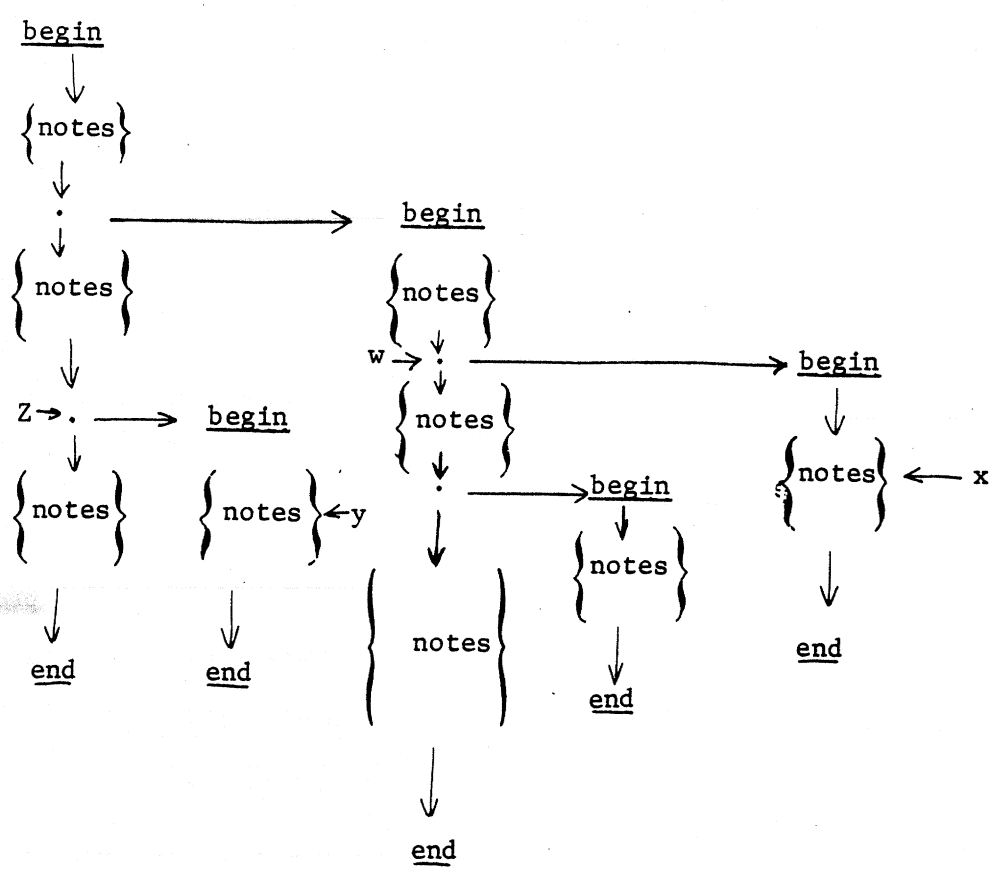
Example 1.1.6.1

Obviously, we may equally well specify subroutine exits which similarly affect the transposition and duration parameters.

In the Introduction we cited an example of a musical situation in which conventional subroutine linkages are inadequate. We wanted the subroutine for the third measure of "Frere Jacques" to call the subroutine for the first measure, but we wanted it to quit after the first three notes. This could be accomplished by assigning an extra voice to cause this voice to leave the subroutine at the desired time as we described in Section 1.1.3; but since we are only concerned with the behavior of a single voice, it would be more desirable to specify the action entirely within that voice's control structure.

In our flow chart for this example, we designated this stipulation by the command "exit after third note" immediately prior to calling the subroutine for the first measure. We shall now consider this procedure in greater detail. We shall also consider a dual problem: Given a procedure tree of nested subroutine calls, we would like to be able to enter this tree at some point other than the beginning and proceed as if we had started at the beginning. This may seem somewhat confusing, but we hope to clarify it before the end of the section.

Consider a melodic passage which might be described by the following model:



Example 1.1.6.2

Let M be the musical passage defined by this model for some realization of note words in the braces. Consider now a second musical passage M', obtained by truncating M at its beginning and end. Suppose the first note of M' occurs at the note word labeled x and that the last note occurs at the note word labeled y. Suppose we wish to access M' as a subroutine.

First consider the exit problem, i.e. let us only concern ourselves with starting M as usual but ending it early. As we already observed, when a subroutine is called, the current value of the program counter is pushed down on a stack and replaced by the beginning of the subroutine. Then, when the subroutine terminates the top of the stack is deleted and this previous value of the program counter is restored. However, if we begin M as usual and simply specify a premature termination a y, then when the top of the stack is deleted at y, the program counter will be restored to the value it had before entering the subroutine containing y, i.e. it will be restored to the point marked z in Example 1.1.6.2. In this particular case, in order to terminate M, one must delete <u>two</u> levels from the top of the stack; but, of course, in other circumstances, one might have to delete some other number of levels.

Thus, the specification of an exit, as we first described it in the Introduction, involves more than just a premature ending of a subroutine. It essentially involves <u>marking</u> the program counter stack before entering the nest of subroutines. Then, when the exit is encountered, one simply deletes elements from the stack until the mark is encountered; and one has returned to the level from which the subroutine was called.

The entry problem is similar in nature, although it entails a different solution. A subroutine call to the location x will cause execution only of the innermost subroutine. When that routine terminates, it will return to the topmost level, rather than to the appropriate location (w). In this case, the solution is effected by a procedure we call nesting, which takes two addresses as arguments. One of these addresses is x -- the goal, so to speak. The other is the address of the subroutine within which x is contained; in this case, it is the address at which the routine for M begins. This instruction causes the processor to maintain the program counter stack, as if it were doing a subroutine call to this latter address but to hold off actual execution until the former address (x) is attained.

With these notions of nesting and exiting, we may represent a model of M' as follows:

```
begin
  ↓
exit at y
  ↓
nest to x  ——>  {routine for M}
  ↓
end
```

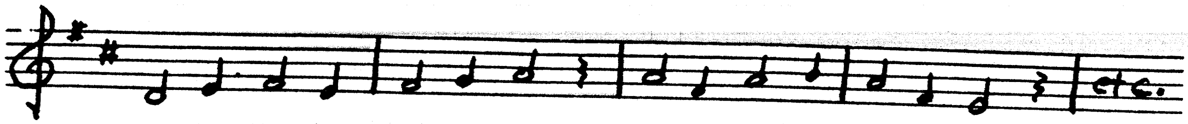
Example 1.1.6.3

Clearly, we may also consider the marking of parameter stacks; so that when an exit is encountered, not only is the program counter stack suitably restored, but the parameter stacks are, too. Similarly, we may specify a nesting procedure in which all alterations to parameter stacks are carried out, and the only instructions which are ignored are the actual note words. Finally, we may allow any of these operations to be subject to intervoice control; that is, we may allow one voice to specify that such operations be executed on another voice.

1.1.7 Melding

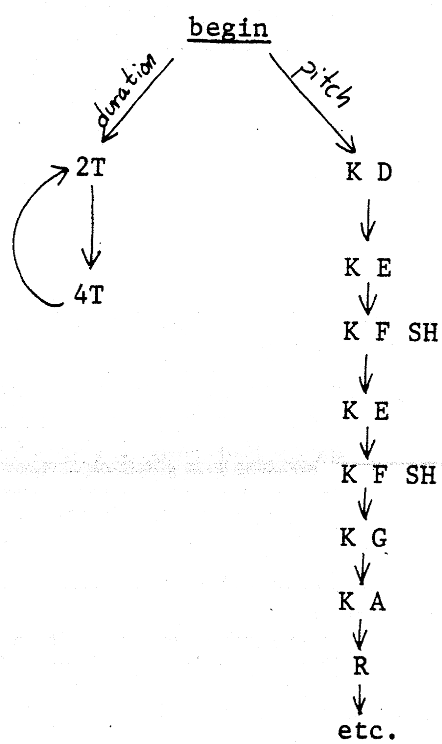
In Example 2.4 we described a subroutine consisting of two separate, but coordinated, processes, one of which determined pitch, the other of which determined duration. This technique of splitting a program into separate routines, which, when united, supply the necessary pitch and duration information, we shall call <u>melding</u>. The basic mechanism of coordination is that each separate process reads one note word at a time, but different processes may restrict their concern solely to either pitch or duration.

We may find examples of such a technique in Mediaeval music which are somewhat less contrived than Example 2.4. The <u>rhythmic modes</u> were fixed rhythmic patterns which attached themselves to melodic lines with little, if any, variation. Thus, a melody might be specified solely by a list of pitches to which the performer would attach a modal rhythm. For example, the following hymn tune (which is <u>not</u> Mediaeval, but rather by Lowell Mason) is a modal melody ([Hymnal]):



Example 1.1.7.1

Using melding, we might model it as follows:



Example 1.1.7.2

We shall discuss rhythmic modes in greater detail in Chapter 3.

1.2 The EUTERPE Language

1.2.1 The Programming Format

Thus far we have only used flow diagrams for the presentation of our examples. Now we shall consider the actual details of the language EUTERPE which we have used for the practical implementation of our models. The language takes the format of an assembly language; and for the benefit of those who are unfamiliar with this format, we shall review some of its features.

The basic unit of our programs is an instruction -- the specification of a single command to be executed by the computer. For example, each of the events as we isolated them in our flow charts corresponds to an instruction. These instructions are written out in order on successive lines and are usually executed one after the other. The lines on which they are written are called their addresses. Addresses are labeled by positive integers increasing one at a time. (Numbering systems are generally either decimal or octal; we shall distinguish decimal integers by always following them with a decimal point.) In addition, addresses may be assigned symbolic names, which are strings of letters and numbers of length at most six. A symbolic name is assigned to an address by writing it out followed by a colon on the line it is to indicate. A symbolic name cannot be assigned to more than one address.

For example, here is how we would represent a note word coding of "Frere Jacques" beginning at address 1ØØ:

```
LOC 1ØØ

K C 4T
K D 4T
K E 4T
K C 4T
K C 4T
K D 4T
K E 4T
K C 4T
K E 4T
K F 4T
K G 2T

etc.
```

Example 1.2.1.1

(We often distinguish the letter "O" from the number zero by drawing a slash through the number.) "LOC 1ØØ" specifies that the first address is 100; then next address is 101, then 102, and so forth. We might attach symbolic names based on the syllables sung to these notes as follows:

```
LOC 1ØØ

FRE:          K C 4T
RE:           K D 4T
JAC:          K E 4T
QUES:         K C 4T
FRE1:         K C 4T
RE1:          K D 4T
JAC1:         K E 4T
QUES1:        K C 4T
DOR:          K E 4T
MEZ:          K F 4T
VOUS:         K G 2T

etc.
```

Example 1.2.1.2

Since we can't use the symbol FRE twice, we used the modification FRE1 in the repetition. It is also possible to refer to addresses by their distance from labeled addresses. For example, rather than assign 101 its own symbolic name, RE, we might refer to it as FRE+1 or FRE1-3, indicating it as the address one after FRE or three before FRE1.

We shall also have occasion to refer to the distinct fields of a specific instruction. For example, a note word instruction has a pitch field (K C) and a duration field (4T). Processes such as indexing and indirect addressing, such as we described informally in Section 1.1.3, will have distinguished fields in their associated instructions, as will the specifications for intervoice control. An instruction which is not a note word will generally take the following form:

```
INS  AC,@ADR(IND)
```

Example 1.2.1.3

The elements which are essential are the fields labeled INS and ADR; all others are optional.

INS is called the instruction field and designates the instruction to be executed. ADR is called the address or argument field and specifies the principal argument of the instruction. The accumulator field, AC, if used, is delimited by a comma and specifies the voice for which the instruction applies. The current version of EUTERPE is limited to six voices, and therefore AC is either an integer from one to six or the symbol ALL (designating all six voices). When there is not accumulator field, the instruction applies to the voice which executes it. Indirect addressing is expressed by the symbol @, while if an address is to be indexed, the in-ex is delimited by parentheses.

This index field also has applications in instructions which are not concerned with indexed addresses.

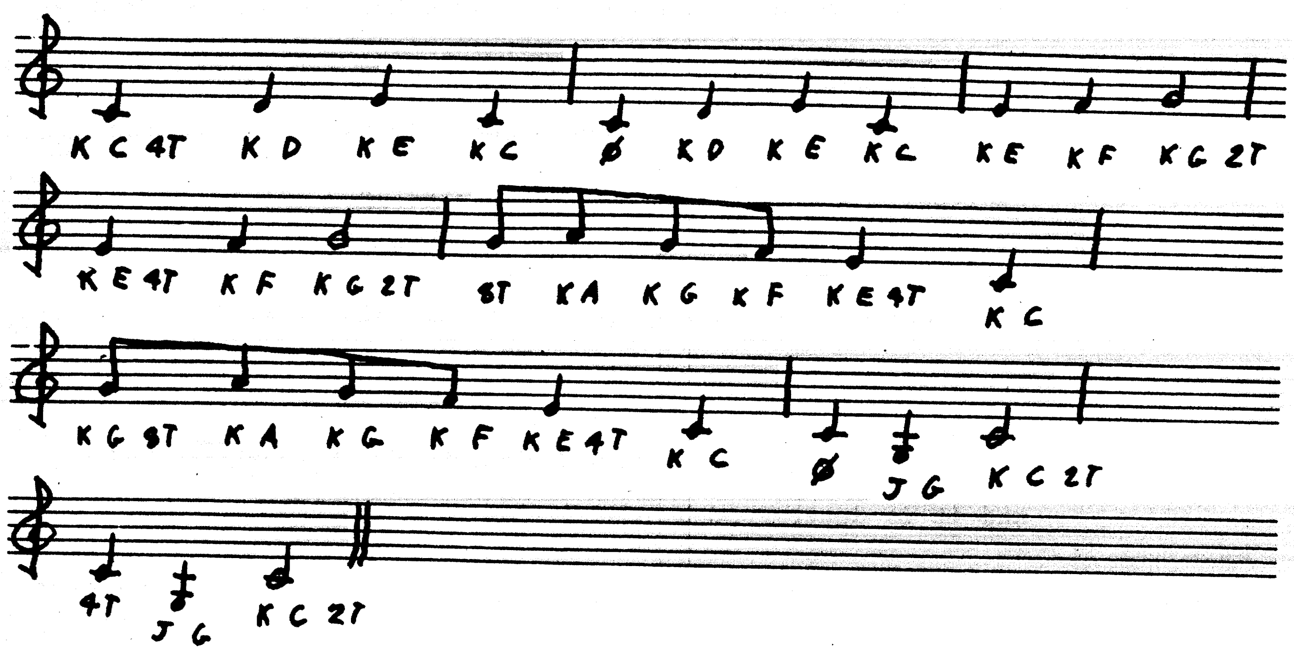
Programs written for the six voices of EUTERPE are called voice programs; and the six voices have the symbolic names VOICE1, VOICE2, ...,VOICE6. If a program is to be assigned to some VOICEn, it begins with the command, LOC VOICEn. When we give examples which may apply to any voice, we may omit the LOC command and give all our descriptions in terms of symbolic addresses.

1.2.2 Note Word Instructions

Note word instructions are almost exactly as specified in Section 1.1.2. Pitch is represented by a string of symbols specifying octave (H through N), position in octave (C, D, E, F, G, A, B), and optional chromatic inflections (FL, SH, Q, P, $, %, T, U). Duration corresponds basically to note shapes ranging from a whole note (1T) to a thirty-second note (32T), with possible modifcations for triplets (e.g. 8T3) and dotted notes (e.g. 2D).

In addition, the index field is used to specify an articulation factor for each note. This is a designation of a fraction of the note's duration which is held silent before the next note is sounded. There are six possible symbols for the articulation field. SLUR designates no silence between notes. Normal mode in this particular system is LEGATO, in which the silence is equal to one sixteenth of the duration of the note (i.e. if articulation is not specified at the outset, it is assumed to be LEGATO). Other possible fractions of silence are one eighth of the duration (SLEG), one quarter of the duration (SSLEG), half the duration (STACO), and five-eighths of the duration (SSTACO).

Any field whichis omitted from a note word assumes the same value it had in the preceding note word. There is an exception to this rule. When we define note words by melding, as was discussed in Section 1.2.6, we shall do so by leaving certain fields unspecified. However, in most cases, the zero word will be interpreted as a note word which repeats its predecessor. Other omissions allow one to alter pitch while preserving the same duration, alter articulation while leaving pitch and duration intact, etc. Furthermore, as we mentioned in the above paragraph, articulation is assumed to be LEGATO if it has no initial specification. We may represent, by way of example, "Frere Jacques" by the following program of note words:



Example 1.2.2.1

1.2.3. Control Transfer Instructions

Associated with each voice of EUTERPE is a program counter. This is a register located at the top of a stack (vide Section 1.1.3). Under most circumstances, this word gives the address of the next instruction which its associated voice is about to process and it is incremented by one after this instruction is completed. This process is altered by those instructions which affect transfer of control.

The simplest such control instruction which we call "TRAnsfer" established the next instruction to be processed as one other than that which follows in the next address location. This instruction is denoted by the symbol, TRA; it takes as argument field the address of the next instruction to be executed and it sets the program counter to this address. The program counter is directly altered without any change in stack structure.

A simple transfer instruction may also employ its index field and indirect field. The address determined by indirect addressing and indexing is called the effective address of the transfer instruction, and it is determined by the following algorithm ([PDP-6]):

1. Let W be the word being processed (i.e. the word containing the TRA instruction).
2. Set E to the number contained in the address field of W.
3. If the index register field is nonzero, add to E the contents of the index register specified.
4. If the indirect bit is off, E is the effective address;

otherwise let W be the word in the address currently specified by E, and go to step 2.

For example, the following program will simply repeat the first three notes of the C major scale:

```
LOC 1ØØ

K C 4T
K D
K E
TRA 1ØØ
```

Example 1.2.3.1

Now suppose the following instruction appears in some other voice's program:

```
TRA 1ØØ
```

Example 1.2.3.2

If the contents of index register 1 is zero, then this voice will enter the same loop at 1ØØ; however, if 1 contains the value 1 or 2, it will enter the loop at 1Ø1 or 1Ø2, respectively. Similarly, the instruction:

```
TRA @1Ø3
```

Example 1.2.3.3

will effect an entry into the loop at 1ØØ; and if 1 happens to contain the value 3, so will the instruction:

```
TRA @1ØØ(1)
```

Example 1.2.3.4

In its simplest form TRA is used to establish loops, as in the example just given or for the canonic procedure of "following at a distance," as in our model of "Frere Jacques." For example, if the program which plays "Frere Jacques" begins at VOICE1 (the first address to be processed by the first voice), then the second voice would begin as follows:

```
LOC VOICE2

R 1T
R
TRA VOICE1
```

Example 1.2.3.5

With a voice argument in the accumulator field, TRA becomes the fundamental mechanism for cueing. For example, VOICE1 may cause VOICE2 to begin a particular process concurrent with an event in the VOICE1 program. Even if VOICE2 is in the middle of sounding a note, TRA has an immediate effect on it.

As an example of intervoice control transfer, we may code the model given in Example 1.1.3.3 as follows:

```
LOC VOICE1

    TRA 2,AL
    K D 4T
    Ø
    K C
    TRA 2,AL1
    K D
    K E
    K D
    K C
. . .

AL: J D 4T (SLUR)
    TRA AL

AL1: J A 4T (SLUR)
     TRA AL1
```

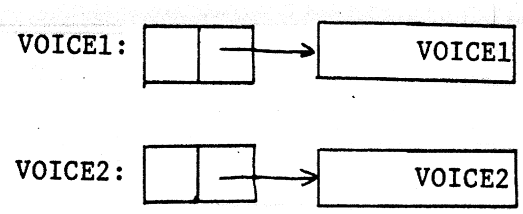
Example 1.2.3.6

After direct transfer the primary means of control transfer is the subroutine call. This was the object of the demonstration instruction PUSHJ$. Like TRA, this instruction takes an effective address as its argument; but this address is appended to the top of the program counter stack rather than replacing the current program counter. Prior to adding the new level to the stack, the program counter is incremented so that later, when the subroutine is finished and the top level of the stack is deleted, the program counter is pointing to the address following the PUSHJ$ which effected the transfer.

Now let us consider how intervoice control is applied to subroutine calls. Suppose VOICE1 begins with the following instruction:

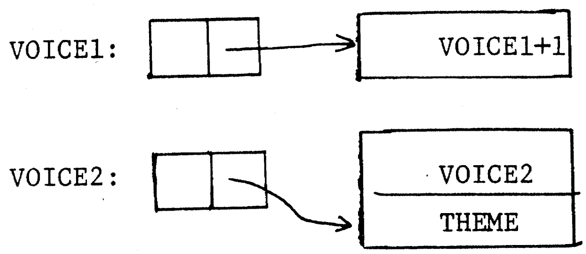
PUSHJ$ 2,THEME

The initial conditions of the two voices' program counter stacks are as follows:



Example 1.2.3.7

In processing this instruction, VOICE1 does two things. First, it increments its own program counter by one; and then, it places the address THEME on top of the program counter stack for VOICE2. Hence, after this instruction is executed, we have the following:



Example 1.2.3.8

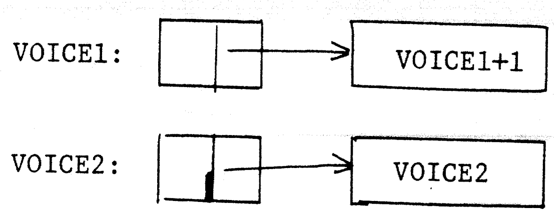
Notice that the former value of VOICE2's program counter has not been incremented. An intervoice subroutine call represents an <u>interruption</u> of the affected voice's processing; and when that subroutine is completed, VOICE2 should be able to pick up where it left off. (This particular example is a somewhat degenerate case of "interruption" since VOICE2 has not yet begun its processing.)

Now consider the following set of instructions for VOICE1 and VOICE2:

```
LOC VOICE1

K C 4T
PUSHJ$ 2,THEME

. . .

LOC VOICE2

J G 2T
J C 2T
```

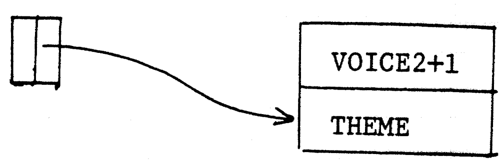
Example 1.2.3.9

In this case, the first two instructions are note words, and the result is a two-note chord sounding for the duration of a quarter note. After this chord is sounded, the program counter stacks are as follows:



Example 1.2.3.10

The program counter for VOICE2 has not yet been incremented because it has not yet finished executing its instruction; the note is only half complete. In this case, the PUSHJ$ instruction has the following effect on the program counter stack for VOICE2:



Example 1.2.3.11

In other words, when the subroutine is exited, VOICE2 will not try to play the note at VOICE2 again, but will pick up with the next note at VOICE2+1. Hence, if a note is interrupted while it is sounding, execution will return to the <u>following</u> instruction when the interruption is completed. Subroutine exits are generally accomplished by the instruction POPJ$, either with or without a voice argument, as the case requires. When used with a voice argument, POPJ$ has the same interrupt facility as PUSHJ$. Since it is returning to an address, however, it does not have to worry about incrementing the program counter; this is all accomplished when the PUSHJ$ is enabled.

By way of example, let us turn to the melismatic organum model in Example 1.1.3.8. Since we have the feature of interruption at our disposal, we may notate the tenor voice with notes of overly long duration and have them interrupted by the duplum in VOICE1. This may be done as follows:

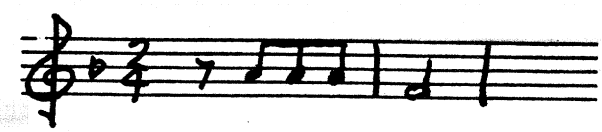
```
      LOC VOICE1

          SETZM I          ;SET INDEX I TO ZERO
          K D 4T
          Ø
          K C
          AOS I            ;INCREMENT I BY ONE
          POPJ$ 2,         ;INTERRUPT TENOR
          K D
          K E
          K D
    . . .

      LOC VOICE2

          PUSHJ$ TENOR(I)
          TRA VOICE2
          TENOR:  J D 1T(SLUR)
                  J A 1T(SLUR)

          etc.
```

Example 1.2.3.12

With I initialized to zero, the first thing VOICE2 does is enter a subroutine to sound its first note. This noe is terminated by the subroutine exit commanded by VOICE1. VOICE2 returns to VOICE2+1, transfers back to VOICE2 and executes another subroutine call, this time to TENOR+1 because VOICE1 has incremented I by one.

1.2.4. Parametric Definitions

Let us now turn to instructions associated with the definition and alteration of parameters. The primary instruction for absolute specification of the pitch parameter is called PITCH. This instruction takes as argument a string of symbols, such as would be placed in the pitch field of a note word, which designates the current setting of pitch parameter; that is, it is interpreted as that note which sounds when middle C is notated. Therefore, it effectively makes a voice a "transposing instrument." For example, the following B-flat clarinet part for the opening measures of Beethoven's fifth symphony:



Example 1.2.4.1

will sound in the proper key of C minor if programmed as follows:

```
PITCH J B FL
R 8T
K A
Ø
Ø
K F 2T
```

Example 1.2.4.2

Whereas PITCH provides an absolute definition of the pitch parameter and therefore directly alters the top of the appropriate stack, RELPIT defines a new pitch parameter relative to the current value and therefore augments the stack with a new value at the top. The argument of RELPIT is an integer which is added to the current pitch parameter. This integer designates a number of microtones (as defined in Section 1.1.2) which constitute an interval, ascending if positive, descending if negative. The RELPIT instruction thus offsets the pitch parameter by a specified interval.

It is sometimes inconvenient to think of most intervals in terms of the number of microtones they contain, so EUTERPE's representation of pitches provides a convenient shorthand. The separate symbols which make up a pitch argument each have an integer value, and the entire argument is the sum of these values. The resulting integer is the distance, in microtones, from the note H C. Hence, H and C are both equal to zero, I is the number of microtones an octave away from H C (i.e., 72), F is the number of microtones between F and C, and I F is the number of microtones in an octave plus a perfect fourth. Similarly, SH (SHarp) is the number of microtones in a semitone (namely, six), FL (FLat) is the negative of SH, and likewise for the other microintervals. For additional convenience, the symbols TONE and SEMI are defined as equivalent representations of D and SH, respectively. For example, the following program will generate a chromatic scale starting on middle C:

```
LOC 1ØØ

PITCH K C
K C 4T
RELPIT SEMI
TRA 1Ø1
```

Example 1.2.4.3

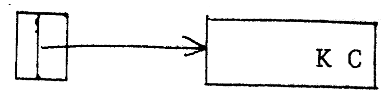
Having no termination, this program will continue until either the range of the player or the capacity of the parameter stack is exhausted.

The standard method of deleting of the top of the pitch stack, i.e. popping back one level, is accomplished by the instruction UNREL PIT. UNREL is a universal popping instruction; and if PIT is one of its arguments, then the top of the pitch parameter stack will be deleted. The following program makes use of storage on the stack to generate a chromatic scale in ascending and descending forms beginning on that pitch which is the current value of the pitch parameter:

```
LOC 1ØØ

K C 4T
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
RELPIT SEMI
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
Ø
UNREL PIT
TRA 1ØØ
```

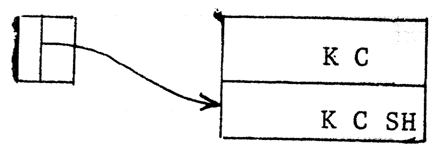
Example 1.2.4.4

The stack behavior in this example is relatively straightforward. Let us suppose the following initial condition:



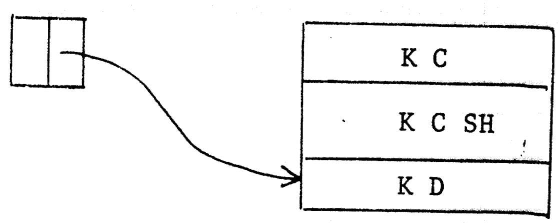
Example 1.2.4.5

At location 1Ø2 we have the following:



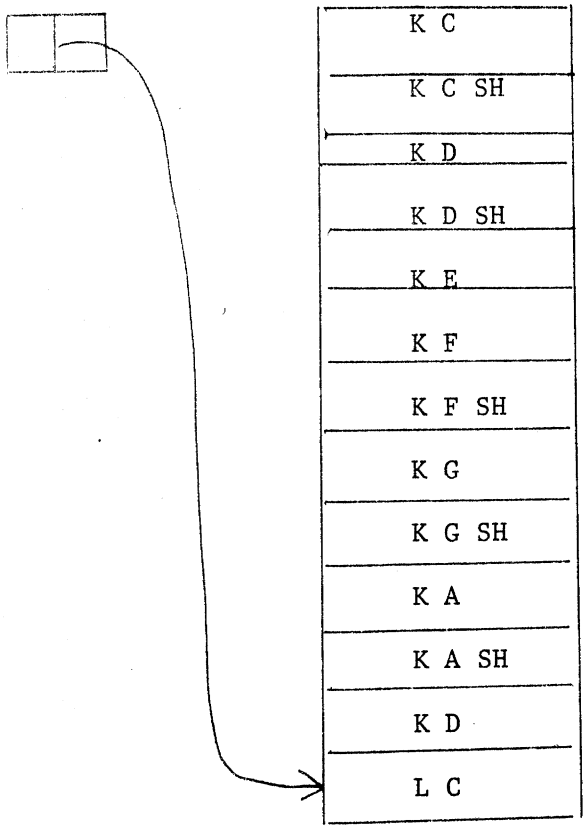
Example 1.2.4.6

At 1Ø4:



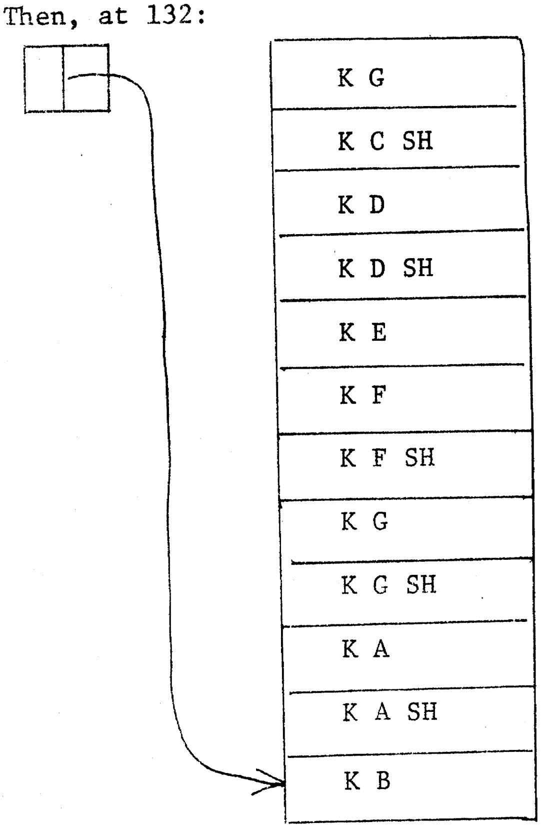
Example 1.2.4.7

Finally, at 13Ø (octal):



Example 1.2.4.8

Then, at 132:



Example 1.2.4.9

This continues until, when we hit the transfer at 160, the stack has returned to its original state and the chromatic scale repeats.

The definition of transposition parameters is essentially analogous to that of pitch parameters. The fundamental instruction is SETRAN; and it takes as its principal argument an integer which specifies the number of scale degrees of transpositin -- ascending if the number is positive, descending if negative. SETRAN also takes an optional second argument in the index field, which is ASC, DES, or AD. This argument indicates that the transposition should be computed with respect to the scale's ASCending, DEScending, or entire (Ascending-Descending) form. (This argument is actually significant only in dealing with the melodic minor scale.) If it is omitted, it is taken to be AD. When such is the case for the melodic minor scale, the decision of which form to use is made as follows: the entire scale form is regarded as a closed loop (i.e. ascending followed by descending, followed by ascending, etc.), and the processor keeps track of the last note to have been transposed; it then advances through the loop in search of the first occurrence of the next note to be transposed; if this note is found in the ascending portion of the scale form, this half is used to compute the transposition; if it is found in the descending portion, then that is employed instead. (By this algorithm, the transposition of a note of a higher scale degree than its predecessor will always be found in the descending portion, unless the note to be transposed is in only one half.)

For an example, let us again consider our simplified version of "Three Blind Mice." The first measure may be coded as follows:

```
M1:  K E 4T
     K D
     K C 2T
```

Example 1.2.4.10

To obtain the third measure, we need only write the following instructions:

```
KEY C(MAJOR)
SETRAN 2
TRA M1
```

Example 1.2.4.11

Analogous to the instruction RELPIT is the instruction RELTRN. Like RELPIT, RELTRN takes an integer argument which is added to the current transposition parameter. It also takes an optional second argument in the manner of SETRAN which specifies reference to the ascending and/or descending portion of the scale form. Unlike the case for SETRAN, however, when this argument is omitted, it is assumed to be the same as that of the previous transposition parameter. The integer argument may also be omitted, in which case it is assumed to be zero. Hence, RELTRN can be used simply to alter the domain of interpretation of the transposition operation (e.g. from ASC to DES), without altering the actual distance of transposition.

The argument for UNREL which pops back the transposition parameter is TRN. To pop back both the pitch and the transposition parameters simultaneously, both arguments are given, in either order, separated by a space: UNREL PIT TRN or UNREL TRN PIT. It is also possible to restore the transposition parameter to a zero value. This is done by the instruction UNTRAN. However, if an UNTRAN is executed while the stack is not empty, a subsequent UNREL TRN will restore the transposition parameter to a non-zero value and it will again take effect.

The standard mode for defining the scale form to which the transposition parameter refers, is the instruction KEY, which takes two arguments. The first of these arguments is a pitch level of the chromatic scale, without octave specification; the second specifying the "mode" is one of the following symbols in the index field: MAJOR, HARM, MELO, NAT. According to this index argument, a major, harmonic minor, melodic minor, or natural minor scale is constructed on the specified pitch level in ascending and descending forms. This is stored in an uninterrupted block of memory with one pitch per word. As an example, the instruction KEY C(MELO) will create the following block of data:

```
C
D
E FL
F
G
A
B
I C
B FL
A FL
G
F
E FL
D
C
```

Example 1.2.4.12

The value of I C designates that C to be an octave higher than the initial one.

Scales may also be defined on a relative basis. In this case, the entire block of scale information is pushed down and a new block is defined. This is accomplished by the instruction RELKEY, which also takes two arguments. The first argument is an integer from 1 to 7, i.e. a scale degree; if this argument is omitted, it is assumed to be 1. The second argument may be the same as any of the second arguments for KEY, or it may be either RELMIN (for relative minor) or RELMAJ (for relative major). When this argument is omitted, the processor checks the distance of two scale degrees from the first argument. If this interval is a minor third, it assumes the second argument to be RELMIN; otherwise, it assumes RELMAJ. Relative minor is always taken to imply the natural minor scale. The argument KY applied to UNREL causes the scale block to be popped back.

Secondary pitch and transposition parameters may also be defined either absolutely or relatively. The absolute definitions of these parameters are enabled by the instructions GAMUT and GAMTRN, respectively. Both these instructions take integer arguments; however, GAMUT expresses an interval in microintervals, while GAMTRN expreses it in scale degrees. GAMUT may use the same notational abbreviations as RELPIT. Relative definition of these parameters is accomplished by RELGAM and RGMTRN, respectively. These also take integer arguments which are added to the current value of the designated parameter. Finally, secondary parameters may be deleted by the respective instructions UNGAM and UNGAMT.

As we shall see in Section 1.3, these secondary parameters do not actually have their own stacks. They are handled, rather, as "spurs" to the stacks for the pitch and transposition parameters. This has certain disadvantages, but it will suffice as a temporary measure until the more ideal version is implemented. In its current version, for example, EUTERPE may realize the model in Example 1.1.4.13 as follows:

```
LOC VOICE1

PITCH J B FL
L C 4T
Ø
PITCH K C
Ø
PITCH J A
4D
PITCH J B FL
8T
PITCH K C
4T
FINE

LOC VOICE2

GAMUT -I
TRA VOICE1
```

Example 1.2.4.13

The duration analog to PITCH (i.e. the instruction which sets the duration parameter to some absolutely determined value) is the instruction TEMPO. This instruction takes two arguments and is modeled after standard metronomic indications, which (as we saw in Section 1.1.5) generally have the following form:

♩ = 60

Example 1.2.4.14

This particular tempo may be established by the instruction TEMPO (4)60. (4) stands for the quarter note; the number may also be 1, 2, 8., or 16. The second argument (which is the right-hand argument of the metronomic indication) indicates the number per minute of occurrences of the note specified by this first argument. (Both arguments must take decimal points if the user intends the representation to be decimal rather than octal.)

The analog to RELPIT is RELTEM; and it, likewise, alters the stack of duration parameters. As in Section 1.1.5, we designate a relative definition of duration as a transformation of "x in the space of y," where x and y are integers. This means that x of the new rhythmic units will occupy the same duration as y of the old ones, and it is notated RELTEM x(y).

Either of the two arguments for RELTEM may be omitted, but they have different default conditions. If the first argument is omitted, it is assumed to be one. However, if the second argument is omitted, it is taken to be the largest power of two less than or equal to the first argument. This convention was suggested by Sims because it conforms with the standard interpretationof odd-numbered groupettes. Triplet notation is used to specify three notes assuming the duration of two; quintuplets denote five in the space of four; and in general, an n-note groupette sounds in the space of m, where m is the largest power of two less than or equal to n. (This applies for some even-numbered groupettes, as well, but those which are powers of two are subject to less consistent interpretations.) Thus, an alternative way of attaining triplets would be by the instruction RELPIT 3, which would be an abbreviation for RELPIT 3(2).

There is one limitation to the RELTEM instruction: the second (parenthesized) argument must not exceed five binary bits, i.e. it must be less than 32 (decimal). This is because of the size of the index field; a higher number would overflow into the accumulator field, where it would be interpreted in terms of intervoice control. This limitation is not overly severe and may be overcome by successive applications of RELTEM. (The default condition still holds if the index field is empty and the argument field is greater than 32 (decimal).)

Secondary duration parameters are precisely analogous to secondary pitch parameters. Corresponding to the instructions GAMUT, UNGAM, and RELGAM are the duration instructions GRUP, UNGRUP, and RELGRP. GRUP and RELGRP have the same format as RELTEM, as well as the analogous immediate effect. Finally, the appropriate argument for UNREL, which pops back a duration parameter, is TEM. This can, of course, be combined with PIT, KY and TRP.

Articulation may also be controlled independently of note words. The instruction ARTIC takes as argument any symbol which may appear in the articulationfield of a note word, and it sets the articulation appropriately. Articulations are not stored on a stack but, rather, in a single register, one for each voice. This is the same register which is set by the articulation field of a note word, so ARTIC is only effective until the occurrence of a note word with an articulation field.

1.2.5 Stack Manipulations

The subroutine exit POPJ$ may specify a voice in its accumulator field for intervoice control. It normally has no argument in its address field, but it is possible to place in the address field of a POPJ$ any argument or group of arguments which may be placed in the address field of UNREL. This will have the effect of the execution of the appropriate UNREL at the time of the subroutine exit; that is, in addition to popping back the program counter stack, a POPJ$ instruction may also pop back the stacks for the pitch parameter, transposition parameter, duration parameter, and scale.

We considered such a feature at the beginning of Section 1.1.6 and applied it to an example by Erik Satie. If we regard the melody line of this example as a subroutine stored at the address THEME1, having a conventional POPJ$ exit, then we may program the model of Example 1.1.6.1 as follows:

```
REPEAT 840.  PUSHJ$ THEME2
FINE
THEME2:  PUSHJ$ THEME1
         RELPIT -I
         PUSHJ$ THEME1
         POPJ$ PIT
```

Example 1.2.5.1

"REPEAT" simply specifies that the PUSHJ$ instruction is to be repeated 840 times. The subroutine THEME2 plays the melody line in THEME1, sets its pitch parameter down an octave, and plays the melody line again. When the subroutine terminates, it resets the pitch parameter back up an octave and is ready to repeat the entire passage.

It may be the case that a parametric alteration within a subroutine may be desired to be <u>binding</u>. Of course, if one exits by a POPJ$ with no arguments in the address field, any parametric alterations made within the body of the subroutine will remain; and if an absolute parametric definition has been applied, the stack itself will not even be altered in structure.

One may wish, however, to define an alteration in relative terms (such as by a ratio in a tempo definition) without adding a new level to the stack. For this purpose there is a special subroutine exit called RETURN. In matters of control, RETURN behaves exactly the same as POPJ$; and like POPJ$, it may take parametric arguments. The effect on parameters, however, is quite different. A parametric argument to POPJ$ deletes the top of the stack and drops back one level; in RETURN it backs down one level carrying the top parametric value along and recopying it at the lower level.

As we remarked earlier, because of limitations of format, certain tempo ratios may be impossible to express as a single fraction and will require successive multiplications. Most likely, however, the programmer will not want to advance the stack more than one level.

Consequently, all applications of RELTEM after the first might be followed by a PUSHJ$ to a word containing the instruction RETURN TEM; and this would keep the duration parameter stack from piling up.

We also considered in Section 1.1.6 the processes of exits and nests. The EUTERPE instruction for these processes are EXIT, EXIT1, and NEST. The instruction EXIT takes an address in its argument field. When it is executed, it "marks" the program counter stack and places the address on a special list against which the program counter is compared at each step. We shall discuss the actual mechanism of marking in Section 1.3.5. At this point, all we need know is that the current level of the stack is "remembered" by a mark. When the program counter matches the address on the list, the stack is popped back until it finds this mark. Thus, one may specify a premature ending within a hierarchy of subroutine calls and still have the appropriate number of POPJ$'s simulated in order to return to the top level. The address of this ending remains on the speciallist until it is removed by an UNEXIT instruction; however, if it is established by the instruction EXIT1, it is deleted after its first application.

Parametric stacks may be similarly marked. EXIT and EXIT1 take optional arguments in the index field which mark the indicated parametric stacks as well as the program counter stack. There are also several ways to mark stacks without specifying exits and related instructions for popping back to marks. The instruction MOP may be substituted for POPJ$, and will pop back the program counter stack to its most recent mark. Like POPJ$, MOP takes optional arguments specifying parametric stacks. Parameters may be restored in the same manner without affecting the program counter through the instruction MUNREL. This instruction is the same as UNREL except that it pops the specified parameters back to their most recent mark. MARK marks the current level of the program counter stack along with any specified parameter stacks, and UNMARK deletes the marks so established. MARPAR and UNMARP have the same effect without influencing the program counter stack. All these instructions take an optional voice argument.

The format of NEST is somewhat different from that of other instructions because of the information it must specify. For example, in Example 1.1.6.2 if the subroutine for the melodic passage is at the address M and our desired entry point is the address X, we would specify the nest by the instruction NEST [(M)X]. The address contains a pointer to a word which has two addresses, one in each of its halves. The right half specifies the entry point; the left (parenthesized) half specifies the "virtual" entry point, i.e. the address which is accessed by a PUSHJ$, from which the "imaginary processing" is conducted until the address in the right half is encountered. PNEST has a similar effect to that of NEST, except that only note words are ignored. This means that all parametric alterations between two specified addresses are observed.

1.2.6 The MELD Instruction

We mentioned an alternative default condition for note words in Section 1.2.2. This arises in melding, in which default conditions are specified by a "silently processed" program. The basic idea is this: When a voice program melds to another process that process is executed in parallel with the voice program without sounding its note words. However, if a field in a note word is absent from a voice program, the corresponding field of the melded program is substituted. A melded program may, in turn, meld on another program, so that if its field is also empty, it can check down another level.

The instruction MELD takes an address as its argument, at which time a parallel process is started at the specified address. All non-note words are executed. However, any stack manipulation is performed with respect to the stacks of the voice which melded the process. This is safe enough for parameters, but one must be careful in handling transfer of control. A TRA in melded program simply transfers it without affecting the voice program. The process is halted by the instruction UNMELD.

Now we can return to our model for "Frere Jacques" and specify the subroutine for the third measure more explicitly. First of all, let us suppose that the subroutines for the first and last measures are written out in note words as follows:

```
MEASU1:  K C 4T
         K D
         K E
         K C
         POPJ$

MEASU7:  K C 4T
         J G
         K C 2T
         POPJ$
```

Example 1.2.6.1

In defining the subroutine for the third measure, which we shall call MEASU3, there are several things we must specify. We wish to express the pitch as the transposition of the first three notes in MEASU1 up two scale degrees in the key of C major. First of all, then, we must specify key and transposition parameters:

```
KEY C(MAJOR)
SETRAN 2
```

Example 1.2.6.2

Next, we wish to specify an exit after the third note. This may be accomplished by the instruction EXIT1 MEASU1+4. All that remains is the determination of rhythm, but this can be accomplished by melding since only the first instruction of MEASU1 specifies a duration field. Thus, we arrive at the following specification for MEASU3:

```
MEASU3:  KEY C(MAJOR)
         SETRAN 2
         MELD MEASU7
         EXIT1 MEASU1+4
         PUSHJ$ MEASU1
         UNMELD
         UNTRAN
         POPJ$
```

Example 1.2.6.3

1.2.7. Miscellaneous Instructions

There are a few more EUTERPE commands which have not yet been mentioned. FINE is the general halt instruction; it transfers control back to the operating program. CANCEL, if it is given a voice number argument, turns off the specified voice. If not, it turns off the voice which processes it. If all voices become cancelled in this manner, execution halts and control is returned to the operating systems program.

There are available tables of sine, square, and sawtooth waves which can be associated with any voice. There is also a WHITE table filled with random numbers within the range of the other tables. These wave forms can be assigned by the WAVE instruction taking as argument SINE, SQUARE, SAW or WHITE. All six voices are initialized by the system SINE. RAN generates a 36-bit random number and places it in the address specified by its argument. Since EUTERPE has a random number generator for the creation of WHITE, it was decided it might as well be made available to the user.

In addition, much of the repertoire of PDP-6 machine instructions is available to EUTERPE. PDP-6. For the purpose of employing these instructions, EUTERPE keeps its own set of accumulators, numbered 1 to 17 (octal). Accumulator 0 is not available to EUTERPE. Memory address 20 is also not available, as this register is used in producing the acoustical output.

There are also some special locations in memory which contain useful information. Memory address VOICE the number of the voice which refers to it. This is helpful in sorting out different voices which are reading the same program. SCALE has the same contents as the pointer to the scale stack for the voice which reads it. Hence, a voice may determine the state of its scale parameter at any time.

1.3. Programming Details

Most of the details concerning the functioning of the EUTERPE instruction set have now been covered. However, there remain a few fine points which essentially depend on the eccentricities of a practical computer system. These include such matters as stack allocation, duration control, and the general problem of coordinating six voice programs. These points will all be discussed in this section. However, we begin with a more detailed look at the overall processor.

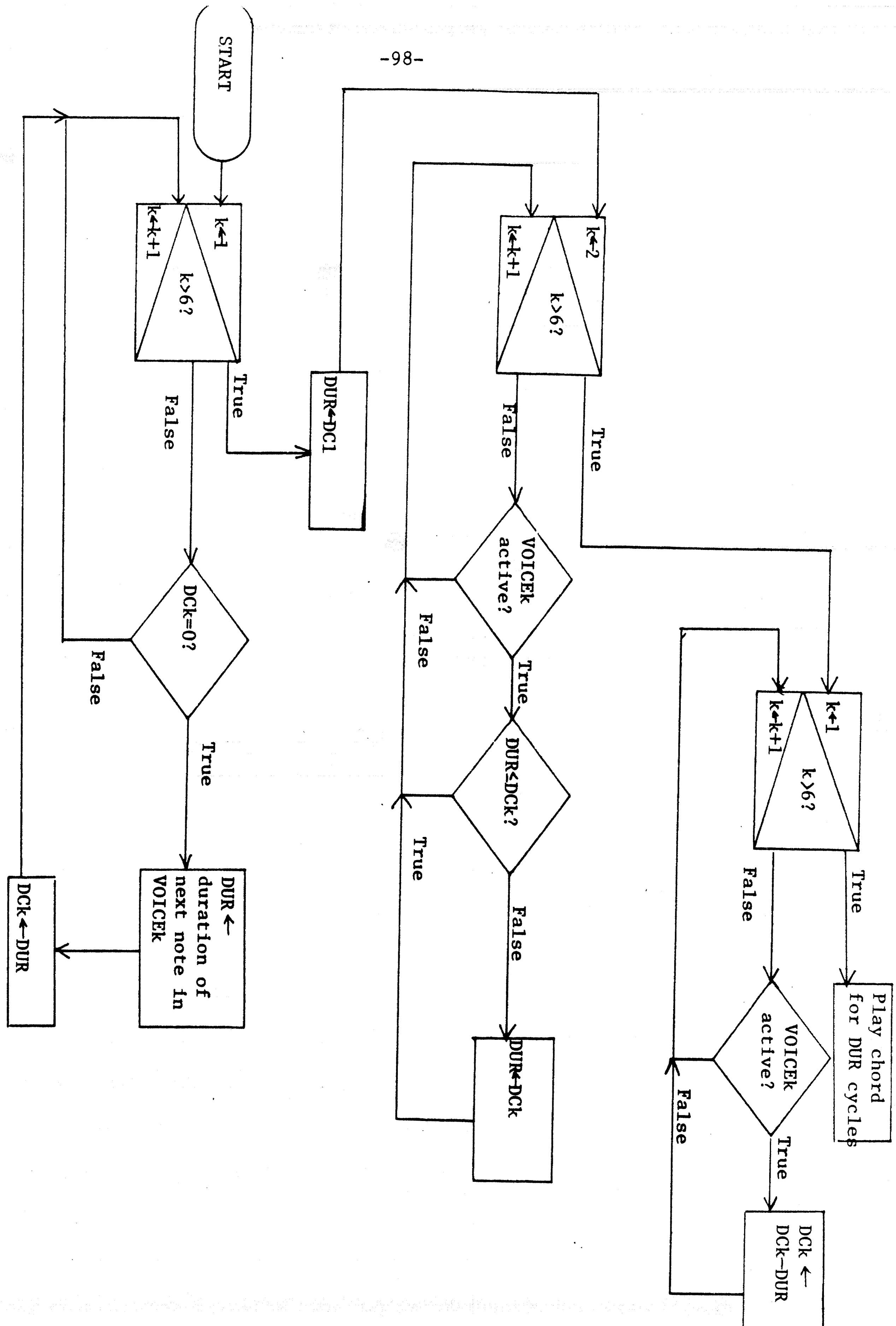
1.3.1. The EUTERPE Processor

The sound generator of the EUTERPE processor is a simple loop program. This loop reads values from a block of memory which contains a digitization of a wave form. The values are sampled at an interval determined by the desired pitch and the duration for which that sampling continues, is expressed as a number of cycles through the playing-loop. Values read from the digital representation of a wave form are then placed in a digital-to-analog converter which drives sound system. Since EUTERPE allows six simultaneous tones (i.e. six voices) but has only two output channels, each D-to-A converter receives the sum of the sampled values for three of the voices' respective wave forms.

Generally, all the voices do not sound notes of the same duration; therefore, some sort of coordination is necessary if all six voices are to share this one loop. This is handled by a set of six duration counters associated with the six voices. At the outset, each voice computes the number of loop cycles required for its first duration and stores this number in its corresponding duration counter. (The duration counters for inactive voices are appropriately marked.) The processor then selects the lowest of the numbers stored in the active duration counters and sets that as the number of cycles which the loop will execute. This number is then subtracted from the duration counters of all the active voices, and the loop is run -- sounding some chord for the length of time of the note of shortest duration in that chord. The processor then inspects the duration counters and obtains the next note words for those voices whose duration counters are zero. The sampling rates in the loop are altered for the new pitches, the new duration numbers are deposited in the appropriate duration counters, and the process is repeated. Hence, those voices whose durations have not run out simply continue sampling at the same rate, continuing their pitches.

The following flow chart illustrates this process. We use the conventions of B. A. [Galler]'s The Language of Computers -- the arrow denotes the assignation of a value, rectangular boxes represent actions, diamonds are decisions, and the rectangle containing three triangles is a decision which increments an index. The registers DCk (k = 1,2,3,4,5,6) correspond to the duration counters (observe, by the way, that the duration counter of an inactive voice is non-zero), and DUR is set to the number of playing-loop cycles at the end of the procedure. (This is a simplification of the actual

EUTERPE process. The portion above the dotted line is basically accurate; the portion below will be discussed in greater detail later in this section.)



Example 1.3.1.1

While this is the basic theory of operation, the reality is not quite so simple. In the first place, in addition to specifying wave form, pitch, and duration, each note has an articulation factor. This is a fraction of the note's total duration which is silent before the next pitch in that voice is sounded. Thus, for each duration there are associated two numbers of loop cycles -- the first of which sets the sampling rate for the appropriate pitch, and the second of which sets the sampling rate to zero -- and a voice does not want to get another note until both these durations have been sustained.

Consequently, when a voice (say, VOICEn) computes its pitch and duration numbers, it sets an Articulation FLAG, AFLAGn. The duration value is broken into two parts and stored in two separate registers, DURn (DURation) and DURSVn (DURation SaVer). The duration counter, DCn, is then loaded with the value of DURn. After the loop runs out, those voices which need new values first check their articulation flags. Any voice for which the flag is set loads its duration counter from the appropriate DURSV register, removes the flag, and sets the sampling rate to zero (i.e. silence). Those voices for which the flag is not set proceed to the next note word.

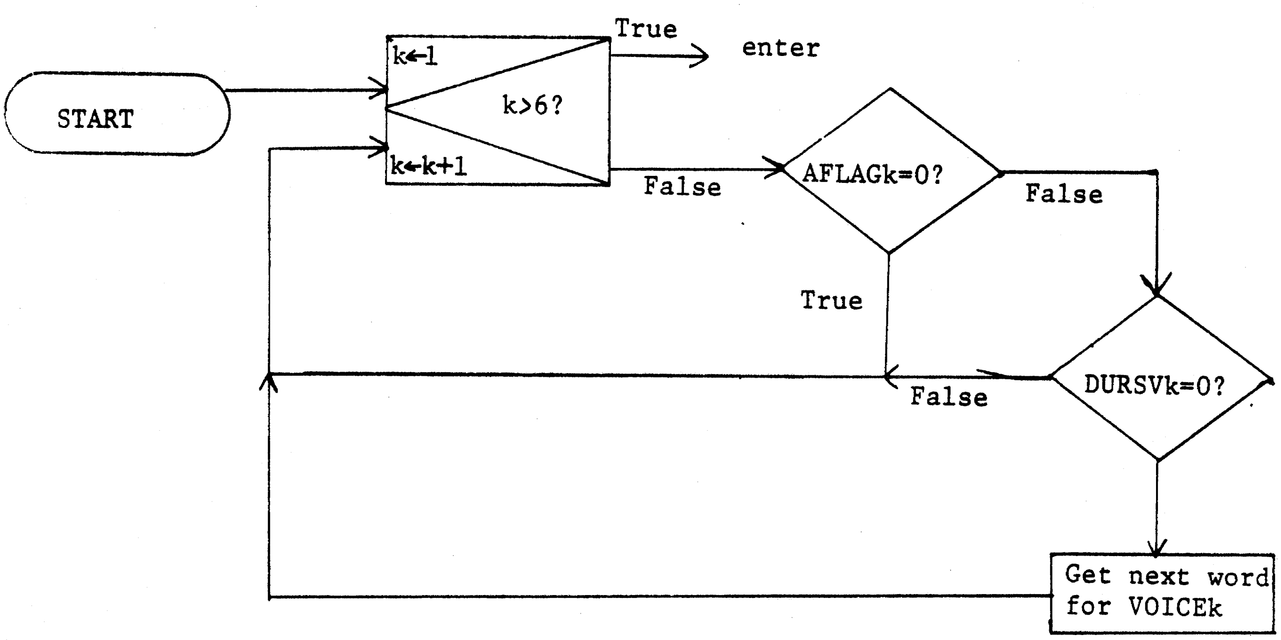
This accounts for the processing of note words, a relatively small percentage of many EUTERPE programs. However, note words are distinguished from other instructions in that they endure for a specific interval of time while all others are intended to operate instantaneously in the musical performance; that is, they are processed in (effectively) zero time units. No playing-loop cycles are intended to be executed during the processing of these instructions. Nevertheless, in the user-controlled real-time operation of the system, these do, in fact, consume small amounts of time. This causes individual chords to be separated by very short gaps of silence which sound as "clicks." A "compiled" version of a composition is obtained by storing away a table of all chords sounded by the player, together with the number of playing-loop cycles associated with each chord.

With each voice is associated a program counter, a register which contains the address of the next instruction to be processed. This register is incremented by one just before the articulated (i.e. silent) portion of a note is to be executed. The program counter is advanced by a subroutine called NEXTWD which also inspects the instruction field of the next word to be processed. (This is contained in the leftmost nine bits of the word and is equal to zero only in note words.) This byte is stored in the register MACFLn (a mnemonic for MACro FLag, since it indicates those macros which are not note words) for the appropriate VOICEn.

When the inner playing-loop runs out, all voices whose duration counters are zero must be refreshed. However, it is also necessary to execute all macro instructions of effectively zero duration before processing any note words, since they may have an effect on the integration of these note words. The very first case for which the processor checks arises only when the articulation mode is set to

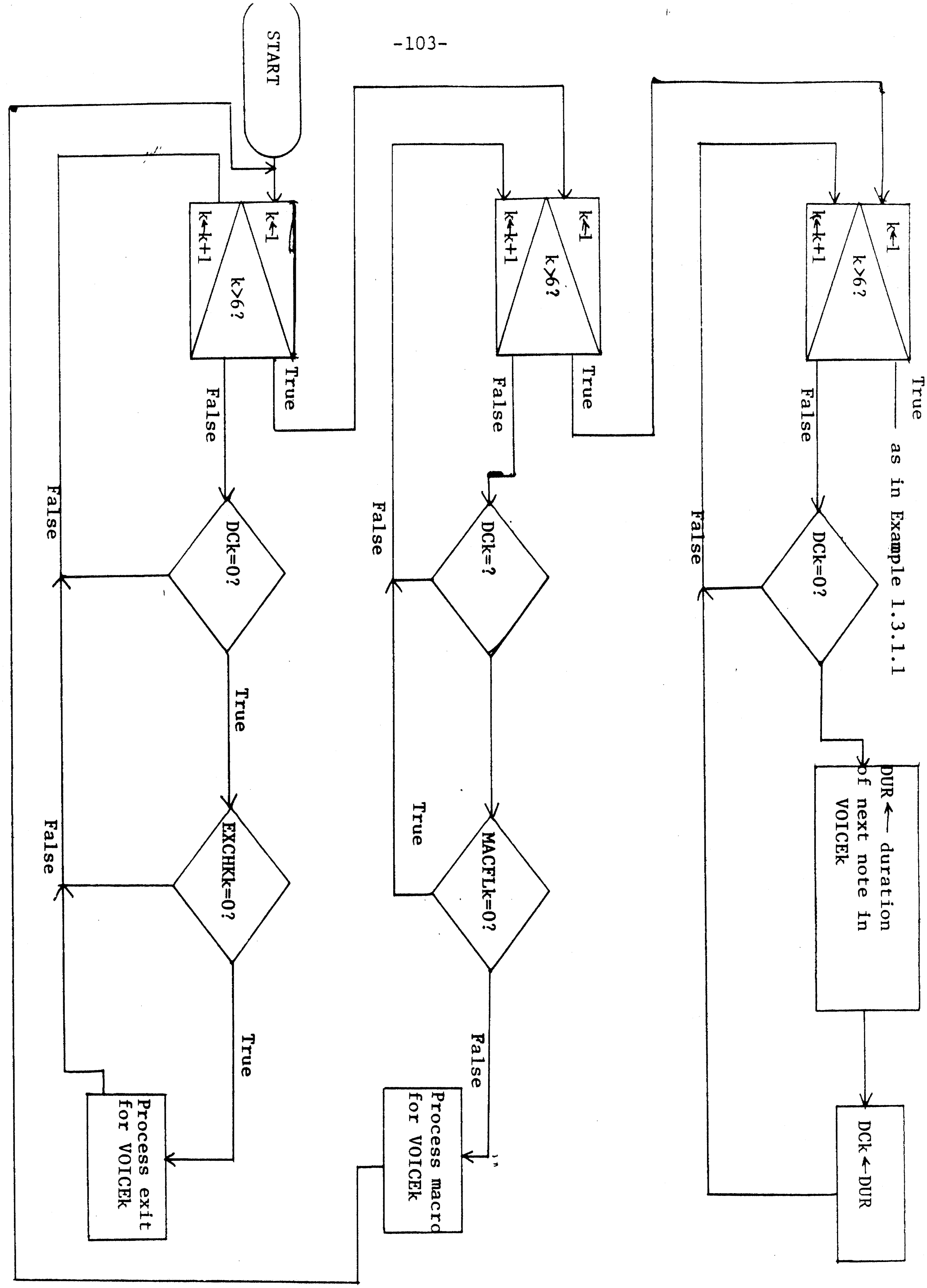
SLUR, so that there is no pause between notes. In this case, a note which has sounded for the duration specified in its DUR register is ready to call the subroutine NEXTWD and reset its articulation flag. Therefore, if a voice has its articulation flag set but has a zero in its DURSV register, the processor immediately takes care of this "pseudo-articulation," calls NEXTWD, and prepares the voice for the main body of the processing loop.

The following flow chart illustrates this processing of pseudo-articulations. The portion of the program it represents is processed just before that portion represented by the flow chart in Example 1.3.1.1.



Example 1.3.1.2

Now let us consider in greater detail what actually happens below the dotted line in Example 1.3.1.1. The following flow chart is a more accurate representation of this portion:



Example 1.3.1.3.

Example 1.3.1.3

Two things must be checked out before EUTERPE is ready to process note words. The first of these is the existence of exits, which were discussed in Section 1.2.5. Secondly, there are those instructions which are not note words, both control macro instructions and assembly language code. A table is kept of those addresses which are exit points, labeled by the numbers of the voices to which they apply. When the program counter for a voice, say VOICEn, is set to such an exit point, a register, EXCHKn (EXit CHeck), is appropriately marked. This signals the processor to call a subroutine which enables the exit. These EXCHK registers are inspected in the order of their respective voices (EXCHK1, EXCHK2, . . . , EXCHK6) after the inspection for pseudo-articulations. If an exit is enabled, the check must be made a second time, since the exit might have returned control to another exit point.

After the exits are processed, the appropriate MACFL registers are checked, again in the order of their voice numbers. Whenever one of these registers is non-zero, the instruction it represents is known not to be a note word; and it is immediately processed. Then it is necessary to check for exits again, as this may have transferred control for some voice to an exit point. Eventually, however, each voice reaches a point at which its next instruction is a note word (unless, of course, a halt instruction has been executed). At this point, the note words are all computed, the duration counters are checked out as in Example 1.3.1.1, and the playing-loop is loaded to sound the appropriate chord.

EUTERPE runs under the control of DDT, an interactive debugging routine. This means that the user is capable of executing and dianosing his program in real time. Not only can he use the features of DDT for inspecting and altering the contents of the six voice programs, but in addition, the EUTERPE processor is equipped with eight break points. Whenever a voice's progra counter reaches an address which is stored in a break point, control returns to DDT. The user may then either proceed or restart, as is the case for conventional DDT break points.

1.3.2. Control Coordination

As we mentioned in Section 1.2.3, the program counters are kept at the tops of stacks. We begin this section with the basic details concerning EUTERPE's allocation of storage for stacks -- not only for the program counters, but for the parameters, as well. Terminology has been chosen to conform with D. E. Knuth's Fundamental Algorithms ([Knuth, 1968]):

Associated with each program counter and each parameter for each voice is an unbroken block of storage. Each voice maintains a set of pointers to locations within these blocks -- one pointer per block. These pointers indicate the tops of their respective stacks, i.e. those locations containing the current values of the program counters and parameters. When a parameter is altered by a relative definition or a subroutine is called, the address of the corresponding pointer is incremented by one and the appropriate value is copied into the new top level of the stack. The reverse process of deleting the top of the stack simply involves decrementing the pointer by one. It is not necessary to actually remove the old value as it will be ignored until the next addition to the stack, at which time it will be erased.

The other mechanism which we need to discuss is that of inter-voice applications of control transfer. As we mentioned in Section 1.2.3, a simple transfer or subroutine linkage applied to a remote voice causes an immediate interruption of that voice's processing, even if that voice is sounding a note. This occurs because these transfer instructions immediately clear out the duration counter of the effected voice so that, as was described in Section 1.3.1, it is ready to process the next instruction as indicated by its program counter. It is also necessary, as far as this mechanism is concerned, to clear the articulation flag of the effected voice.

1.3.3. Secondary Parameter Manipulation

The manipulation of pitch, transposition, and scale parameters is straightforward and essentially as outlined in Section 1.2.4. However, as we mentioned, the secondary parameters are not currently implemented as separate stacks. Instead, we have adopted a temporary measure which allows the construction of a "spur" to any point along the pitch parameter stack. As we shall see, this spur is a secondary stack of depth one. It is, therefore, considerably weaker than an actual stack; but it still has many benefits.

We divide each word in the pitch parameter stack into two halves. The left half is normally zero, in which case pitch is computed with respect to the parameter in the right half; this is the parameter which is set and altered by PITCH and RELPIT. However, when the left half is non-zero, its value is treated as the pitch parameter; and it is read _instead of_ the right half.

The left half of the pitch parameter word is made non-zero by the instruction GAMUT. GAMUT takes the same argument as RELPIT, and it has exactly the same immediate effect. However, instead of adding a new word to the top of the stack with a new pitch parameter, this parameter is deposited in the left half of the current top of the stack. We have not seemingly achieved anything that could not have been accomplished by a RELPIT, however, subsequent PITCH and RELPIT instructions now have different consequences. RELPIT acts exacly as before, only now it adds its argument to _both halves_ of the parameter word. This means it functions as usual, since the lefthalf is readjusted so that _its relative distance from the right half is preserved_. Thus, the program given in Example 1.2.4.13 achieves the desired effect.

Parametric control is restored to the right half of the parameter word by zeroing out the left half. This is accomplished by the instruction UNGAM. Of course, if lowe elements of the stack have non-zero left halves, these will not be eliminated by the UNGAM instruction. The user should therefore be careful about where he uses this instruction, keeping track of practically what is already on the stack.

There is also an analog to RELPIT, which adds new elements to the stack but only modifies the left half of the parameter word. This instruction is RELGAM. In this case, UNGAM will restore an unaltered right half. RELGAM may conceivably be of use in manipulating the rule of octave equivalence in twelve-tone music; but it was implemented solely out of a sense of consistency on the part of the author. There is also an analogous set of instructions for the transposition parameter list. The analogs for GAMUT, UNGAM, and RELGAM are, respectively, GAMTRN, UNGAMT, and RGMTRN. The corresponding instructions for a secondary duration parameter, GRUP, UNGRUP, and RELGRP, are also handled similarly, with one important exception which we shall now discuss.

1.3.4. Duration Parameters

As we mentioned in Section 1.1.5, our theory of duration parameters does not admit of a smallest indivisible unit of time. However, as far as EUTERPE's processor is concerned, such a unit exists — namely, the time of a single cycle through the inner playing loop described in Section 1.3.1. This unit of time is small enough to be inaudible; but, as we shall see, a difficulty arises from the stipulation that any duration must be an integral multiple of this unit of time.

Clearly, EUTERPE's duration parameter must be related to this unit of time for a loop cycle. In fact, the duration parameter specifies the number of loop cycles which sustain the unit of time of a thirty-second note. Under optimal circumstances, this is an integer. However, it may also be a fraction if such is necessary. Initially this parameter is set to the octal integer 640, which was empirically determined to be the appropriate parameter for sixty quarter notes per minute.

In the ideal situation the duration parameter is an integer; and a RELTEM instruction which expresses a transformation of x in the space of y will take this integer, multiply it by y, divide it by x, and obtain an integer result which is the new duration parameter. Unfortunately, this is not always the case. The division does not always come out evenly, and the quotient alone will not be a sufficiently accurate representation of the temp alteration. (It may not be observable within the context of a single voice; but if one voice is trying to sound five notes against another voice's seven, a noticeable phase shift may result.)

Hence, there are situations in which it becomes necessary to represetn a fractional tempo parameter as an ordered pair of integers. The most convenient way to do this would be to store numerator and denominator in the right and left halves of a single word. However, this would interfere with our representation of the secondary duration parameter, so it is necessary to allocate more than one word on the stack for a single parameter.

Consequently, the storage allocation is as follows: If both duration parameters are integers, then they are stored in the right and left halves of the word indicated by the stack pointer, exactly as is the case for pitch parameters. If either of these is a fraction, an alternative representation is used. The address indicated by the stack pointer contains either a -1 or a -2. The latter is used if there is a secondary duration parameter, in which case two fractions must be stored. (If the primary duration parameter is integral but the secondary is not, then both must be stored as fractiions.) The former is used if there is no secondary duration parameter. There follows on the stack either one or two words containing fractional representations with the numerator in the right half and the denominator in the left.

Such a situation will not arise from an application of the instruction TEMPO. This instruction always assigns an integer approximation to the metronome marking which it represents. When a RELTEM occurs, the processor checks the top of the stack to see if the current parameter is re-compressed into a single word. If the parameter is not in fractional form, EUTERPE checks to see if the division is even. If so, it is executed; if not, the results are converted into fractional form. Most of the fractional arithmetic routines are those prescribed by Knuth in his Seminumerical Algorithms ([Knuth, 1969]).

Representation of a fractional duration parameter is, however, the easy half of the problem. When this parameter is multiplied appropriately for the proper duration symbol, the result may still be a compound fraction; and if we take the integral part of that fraction as the number of loop cycles, we are practically back where we started. Since the loop cannot be cycled a fractional number of times, we need to keep track of the remainder.

Each voice has an appropriate FRAC register which keeps track of the FRACtional portion of the loop it has "theoretically" traversed. Whenever this fraction becomes greater than or equal to the integral portion of the duration number is incremented by one and the fraction is renormalized to be less than one. For example, suppose the remainder is consistently 4/7. Assuming that FRAC was initially zero, the first duration computation sets it to 4/7. The next time, when 4/7 is added to it, it becomes 8/7; this effects an extra increment for the duration number and the fraction is reset to 1/7. Subsequently it becomes 5/7 and then 9/7; this causes another increment and a readjustment to 2/7. Next, it becomes 6/7 and then 10/7, causing another increment and readjustment to 3/7. The next value is 7/7, which causes an increment and restores FRAC to zero. Thus, the duration number was incremented by one four out of seven times — precisely the desired effect when the remainder is 4/7. The same algorithm will work in less consistent cases, even when the fraction of the remainder takes on a new base. Furthermore, because the explicit notation of triplets entails a division by three which may not be even, similar adjustments are made for these circumstances, even though the duration parameter may not be fractional.

1.3.5. Stack Marking

To understand the marking process, one must understand the actual working of the stack. As we have already mentioned, the top of the stack is indicated by a pointer word whose right half contains the address of the top of the stack. Initially, this word has the address of the stack bottom in its right half and zero in its left. Each time a new value is pushed onto the stack, both halves of the pointer word are incremented by one. When the top of the stack is popped off, both halves are decremented by one.

When the stack is marked, the current value of the left half of the pointer word is saved on a stack, and the left half is set to zero. Subroutine entries proceed as usual, incrementing both halves of the pointer word. When an address specified by an EXIT instruction is encountered, the stack is popped back until the left half of the pointer word is zero. The left half is then restored with the value from the top of the stack where the data was saved, and the processor continues. By saving this value on a stack, exits may be nested. By using zero as the mark, one can return to the very bottom of the stack as a default condition. The case is similar in the marking of parametric stacks.

INTERMEZZO -- By Way of an Apology for a Cursory Treatment of the History of Music

When we outlined the underlying theory of the EUTERPE system in Section 1.1.1., we tried to provide simple musical examples to motivate the principal features. Now we shall consider the applicability of the system as a whole, and this will involve a presentation of a corpus of models of different forms of music of the past. In its most thorough sense, this would involve a study of the history of music, which is to say a life's work. . . at least. Consequently, we hope the reader will be content if we simply pick out a few prime excerpts from this history of music and demonstrate how the basic ideas from these periods may be modeled in EUTERPE.

In the next four chapters we shall examine examples of Gregorian chant, early polyphonic writing, techniques of counterpoint and harmony in the music of Bach, and the sonata form of the classical style. Our treatment of Gregorian chant is the broadest although it is far from exhaustive. It seemed desirable to consider the aspects of writing a single voice program before worrying about the parallel processing problems of polyphonic models. The study of early polyphony is scantier than its predecessor. It is basically an examination of the notational techniques of the period and the translation of these techniques into the terminology of computer languages.

The bridge from early polyphony to the baroque period is unfortunately abrupt. We warn the reader that in the course of a few pages, he will find himself snuttled from "Sumer is Icumen In" to Bach. Our considerations of polyphony and harmony are centered around J. S. Bach, primarily because of his skill in both techniques. The major analysis in this section is of the two-part invention in D minor, which illustrates EUTERPE's capabilities in representing both contrapuntal and harmonic techniques.

The chapter on sonata form is somewhat of a token chapter. When he gets there, the reader will find another apology for <u>its</u> brevity. Nevertheless, it would have been criminal to totally overlook sonata form.

After having given sonata form the short shift, we turn to a chapter which we call "Applications." These feature samples of original music and programs which were composed and/or designed with EUTERPE's assistance. We wish to express our gratitude to the composer Mr. Ezra Sims for having taken the trouble to learn the fundamentals of EUTERPE, as well as for provoking the author into teaching her some new tricks.

Chapter 2. Plainsong

2.1. Psalmodic Chant

In this chapter we shall consider Gregorian chant and related forms of monophonic and rhythmically free melody dating from the Middle Ages. This is in no way meant to be a historical study. We are concerned only with certain formal techniques and the manner in which such techniques may be modeled in EUTERPE. Consequently, our choice of examples will be somewhat out of proportion, from a historical viewpoint. This does not preclude the possibility that EUTERPE may be valuable in historical studies, but such matters are beyond the scope of our current knowledge of music of the Middle Ages.

Following the study of Dom Paolo [Ferretti], O. S. B., we shall divide our work into three categories: strophic chant, psalmodic chant, and commatic chant. Strophic chants, which tend to postdate the classical Gregorian era, are most common as settings of hymns, consisting of a single musical passage (i.e. strophe) which is strictly iterated for each verse of the hymn. Because the verses of psalms are not always the same length, psalmodic chant admits of certain minor variations, although the overall form is still basically an iterative one. The term "commatic," which is now somewhat antiquated, refers to those chants which are freest in form.

In considering the representation of chant in the format of a computer language, we cite the following passage by [Ferretti]:

> Gregorian chant possesses a style and a musical language of its own, consisting of characteristic modes of expression, entirely different from those used in other sorts of music. These modes of expression, familiar to the composers of liturgical melodies, we call formulas. There are special formulas which are proper for such-and-such a melody. But for an aesthetic study, the formulas common to several melodies, be they of the same Mode or of the same tupe, present a particular interest. These are the ubiquitous formulas. They represent the most traditional and most characteristic element of classical Gregorian art. These formulas have a certain analogy to modern musical themes; for example, they admit of variations. The size of the formulas is not absolutely fixed; it varies from a minimum of two notes to as much as an entire phrase. The size depends especially on the melodic style. A thorough knowledge of the intimate structure of formulas is indispensable to the composer, esthete, and archaeologist. It is thus impossible to compose or judge a Gregorian melody without knowledge of this characteristic language which is the classical Gregorian musical language; it is equally impossible, in the process of comparative and analytical studies, to reduce a particular melody to its primitive purity without first establishing the melodic, modal, and rhythmic values of the formulas employed in its composition.

If we substitute the word "subroutine" for "formula," we establish the essential interpretation of Gregorian Chant through EUTERPE. Note words, as such, are almost entirely absent from chant programs at the top level; and the primary task is to establish a vocabulary and hierarchy of subroutines which will suit "this characteristic language which is the classical Gregorian musical language."

Before we give any specific examples, we are obliged to say a few words about the materials of Gregorian chant. (Readers whose background is primarily mathematical may, at this point, appreciate the position of those musicians who negotiated the preceding chapter.) It is hard to be brief on the subject of Gregorian chant. Perhaps the most terse account which may be reccommended is Alec [Robertson]'s chapter in The Pelican History of Music. At the other extreme, we have a very thorough study by Willi Apel entitled Gregorian Chant, ([Apel, 1958]).

The notational system we shall employ is the one used by [Davison] and Apel in their Historical Anthology of Music. It is basically a representation of the square neume system on the conventional five-lined staff. Most neumes are not single notes but rather a compounding of note heads. This notation preceded the origins of the staff, and originally the neumes were merely a nuemonic aid for the intervallic motions of the melody line. The direction of the intervals was easily indicated; the distance was more uncertain. We shall represent compound neumes by phrase marks which span the individual note heads in a single neume.

For note shapes we shall use only quarter notes without stems, as in Example 1.1.3.1. On the matter of rhythm, the musicologist Gustave [Reese] writes, "The whole problem of the rhythmical interpretation of plainsong in the Middle Ages . . . has been a subject of intense controversy, a controversy that still rages today." He cites three approaches to the issue: the accentualist theory of Dom Pothier, the Solesmes school, and the mensuralist theories.

Concerning the first, [Reese] writes, "The accentualists believe that the Chant adopted the equal time-values of the syllables, and they consider the accent the principal -- according to some, the only -- rhythmical determinant of its melodies." On the subject of the second, he says, "The monks of Solesmes, under the leadership of Dom Mocquereau, retained Dom Pothier's theory that all the notes of the neumes are basically equal in duration and that plainsong rhythm is free, as opposed to measured, but they discarded his theory that the verbal accent is the predominating rhythmical element." The Solesmes editions provide several diacritical marks -- dots and dashes -- which indicate prolongations of certain notes. The mensuralists go one step further in the belief that each note has its own durational interpretation. There appear, however, to be as many theories of mensural interpretation as there are scholars advancing these theories; and, in general, the Solesmes approach is the most firmly established. We shall not, however, apply the Solesmes prolongations, since our concerns will be strictly with melodic information. This would put us in the same camp as the accentualists, were it not for EUTERPE's inability to express any sort of stress on its notes.

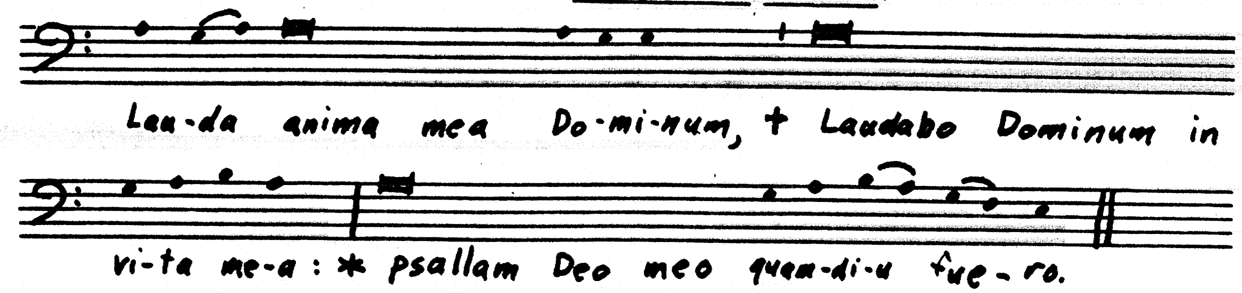
We shall use two post-Gregorian theoretical devices in our programs. These are the so-called "gamut of Odo of Cluny" and the eight "church modes." While both these devices will prove to be valuable, we must remember that they are after the fact and that when they were first introduced, the theorists tended to corrupt original source material so that it might better conform to their theories.

Odo's gamut was simply a list of pitches which were permissible in the composition of a melody. Its lowest pitch is the second G below middle C (I G, in EUTERPE's format), and it rises through the "white keys" to the A above middle C (K A). That is, it consists of the pitches G, A, B, C, D, E, F, G, A, etc. The only chromaticism which Odo recognizes is that of the B below middle C, which may be lowered a semitone to B flat, ([Reese]).

The church modes may be regarded as "distinguished subsets" of Odo's gamut. Each mode is a sequence of eight pitches spanning an octave. There are eight modes which may be grouped into four pairs, each pair being called a _maneria_. Each mode has two distinguished tones, a _final_ and a _tenor_. The two modes in each _maneria_ both have the same final. The _authentic_ mode of each _maneria_ has this final as its lowest tone, and proceeds upward one octave. The _plagal_ mode, the second of the pair, begins a fourth below the final. Thus, the two modes share five tones within the interval of a fifth (a pentachord) and differ over the interval of a fourth (a tetrachord) on either side of this pentachord. The four finals of the four _maneriae_ are D, E, F, and G, all in the octave below middle C. These _maneriae_ are known, respectively, as first, second, third, and fourth, or, in Greek, _protus_, _deuterus_, _tritus_, and _tetrardus_. The authentic modes are also called *dorian*, *phrygian*, *lydian*, and *mixolydian*, respectively, while their corresponding plagals are called *hypodorian*, *hypophrygian*, *hypolydian*, and *hypomixolydian*. Reese provides the following rule of thumb for determining the tenor of a given mode:

> . . . except where the note would be b, the tenor of an authentic mode is always its fifth, while the tenor of a plagal mode is always a third below that of the corresponding authentic. Where the tenor would be b, c is substituted. ([Reese])

We shall begin our study with an examination of psalmodic chant. This has a basic design which may be expressed in terms of a simple computational format. The following psalm setting, transcribed in Davison and Apel, is taken from the *Antiphonale Romanum*:



Nolite confidere in principibus:* in filiis hominum, in quibus non est salus.
Exibit spiritus ejus, et revertetur in terram suam:* in illa die peribunt omnes cogitationes eorum.
Beatus cujus Deus Jacob adjutor ejus,†spes ejus in Domino Deo ipsius:*qui fecit caelum et terram, mare, et omnia quae in eis sunt.
Qui custodit veritatem in saeculum,†facit judicium injuriam patientibus:*dat escam esurientibus.
Dominus solvit competitos,*Dominus illuminat caecos.
Dominus erigit elisos,*Dominus diligit justos.
Dominus custodit advenas,†pupillum et viduam suscipiet:* et vias peccatorum disperdet.
Regnabit Dominus in saecula, Deus tuus Sion,*in generationem et generationem. ([AR])

Example 2.1.1

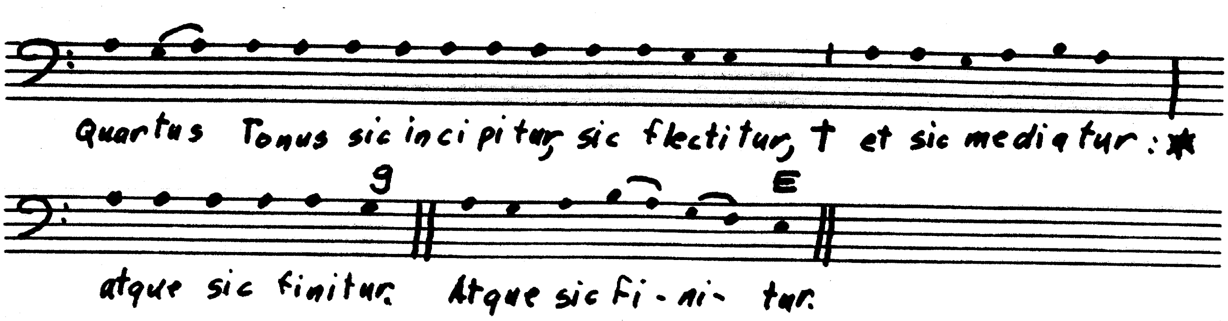
[Davison] and Apel provide the following analysis of this chant:

> The verses of the psalm are sung to a recitation melody (psalm tone) which consists of a reciting note called *tenor* and a number of short inflections called *initium* (intonation), *flexa* (flex), *mediatio* (mediation), and *terminatio* (cadence). The intonation is used for the first verse only, the flex only for longer verses which are divided into three sections instead of the usual two, for which mediation and cadence only are used.

Concerning the psalm tones themselves, Gustave [Reese] provides the following information:

> There are nine systematized psalm-tones grouped together in the Vatican Edition of the chant books -- eight regular and normal, of which one is associated with each of the modes, and one "irregular" called the *Tonus peregrinus* ("strange, foreign") which has two reciting tones. Texts are fitted to the Tones according to definite rules, so that it is not necessary for the books to print out the music for each psalm. The second, fifth, and sixth Tones have a single final cadence, but the others have more than one. These optional cadences are called *differentiae*. They arose from the necessity of ending the verse with some note that would be in harmony with the first notes of the different antiphons (upon their repetition), which do not all begin on the final: an antiphon may start with any note in the petachord and, in plagal modes, with any in the lower tetrachord also.

In this particular example the fourth psalm tone is employed. Here is the specification of this psalm tone as it appears in the *Antiphonale Romanum* ([AR]):



Example 2.1.2

The predominant tone in this passage is the _tenor_ of the fourth psalm tone. The intonation consists of the first three notes, and the cadence in its optional forms consists of the concluding notes. The flex is the motif to which is sung the word "_flectitur_" (preceding the dagger), and the mediation is specified by the word "_mediatur_" (preceding the asterisk). The daggers and asterisks in the psalm text correspond to those in the psalm tone, specifying those syllables to which are applied the flex and mediation. The remaining syllables are sung on the _tenor_, except for the syllables of the opening intonation.

We shall approach the encoding of chant through the preparation of a series of "templates" -- special - purpose files which contain information relevant to particular situations. The most fundamental of these templates is entitled GAMUT AND NEUMES. This file contains the gamut of Odo of Cluny. The finals of the four _maneriae_ are labeled with the names of the authentic modes (DORIAN, PHRYGIan, LYDIAN, MIxoLYdian), while the names of the palgal modes are assigned to the pitches four steps below the corresponding finals (HYPODOrian, HYPOPHrygian, HYPOLYdian, HYpoMIxoLYdian). Accumulator 17 (TONOS) is set aside for purposes of indexing through this linear array.

Next are supplied routines for the most commonly used neumes. These are used so frequently in all the modes that it is most convenient to store them along with the gamut. The simplest of these are CLIVIS (a pair of notes descending a single scale-degree), PODAT (a pair of notes ascending a single scale-degree), and PES (a pair of notes ascending two scale-degrees). These routines are composed out of very little material; in fact, they all share the same exit -- the POPJ$ at the end of CLIVIS. The last two instructions of this routine are used to specify the final tone of each formula. PES, of course, is defined simply by adding another scale-degree to the interval in PODAT.

The next two formulas are three-note patterns formed by linking two two-note patterns together.

TORCU rises one scale-degree with a PODAT and returns back with a CLIVIS. (The linkage is accomplished by transfering to the address CLIVIS+1.) SALIC rises one scale-degree with PODAT and then two more with PES. Finally, the REGIAS idiom is simply a descending four-note pattern.

It should be mentioned that this file is by no means a finished product, nor are any other templates which are discussed in this chapter. All such files are simply the results of analyses which have been performed to date. This encompasses a relatively small corpus of material in comparison to the entire repertoire. However, all these files are capable of being updated as new information is made available. Such revisions will be implemented as the result of future research.

Bearing all this in mind, here is the current version of the **GAMUT AND NEUMES** template:

```
TITLE GAMUT AND NEUMES

TONOS=17

          I  G
HYPODO:   I  A
HYPOPH:   I  B
HYPOLY:   J  C
DORIAN:   J  D
PHRYGI:   J  E
LYDIAN:   J  F
MILY:     J  G
          J  A
          J  B
          K  C
          K  D
          K  E
          K  F
          K  G
          K  A

HYMILY=DORIAN

CLIVIS:  XCT @TONOS
         SOS TONOS
         XCT @TONOS
         POPJ$

PODAT:   XCT @TONOS
         AOS TONOS
         TRA CLIVIS+2

PES:     XCT @TONOS
         AOS TONOS
         TRA PODAT+1

TORCU:   PUSHJ$ PODAT
         TRA CLIVIS+1

SALIC:   PUSHJ$ PODAT
         TRA PES+1

REGIAS:  PUSHJ$ CLIVIS
         SOS TONOS
         TRA CLIVIS


;
```

EXAMPLE 2.1.3

The next necessary template contains information relevant to the specific maneria. Since we are currently considering an example in the hypophrygian mode, we need a template for the second maneria. The current version of this file contains the routines FINAL, which sets the pointer TONOS to the final of the mode, and HYPOTN, which sets the pointer to the tenor of the plagal mode, i.e. the tenor for the fourth psalm tone. The file also contains some additional formulas which were found in examples of chant in the second maneria. The .INSRT instruction effects the loading of the GAMUT AND NEUMES file whenever this file is accessed.

```
TITLE TEMPLATE FOR SECOND MANERIA CHANT

.INSRT NEUMES >          ;LOAD GAMUT AND NEUMES

FINAL:  MOVEI TONOS,PHRYGI
        POPJ$

HYPOTN: MOVEI TONOS,PHRYGI+3
        POPJ$

CLIPOD: PUSHJ$ CLIVIS
        SOS TONOS
        TRA PODAT

CODA:   PUSHJ$ TORCU
        TRA CLIVIS+1

AMEN:   PUSHJ$ CODA
        XCT @FINAL
        POPJ$

;
```

EXAMPLE 2.1.4

Finally, a file is needed to specify the fourth psalm tone. This simply entails converting the information in Example 2.1.2 into EUTERPE format. This involves subroutines for the intonation (INCI4), flex (FLEC4), mediation (MEDIA4), and the different possible cadences (FINI4G and FINI4E).

```
TITLE QUARTUS TONUS

.INSRT DEUTER >                 ;LOAD TEMPLATE FOR SECOND MANERIA

INCI4:   PUSHJ$ HYPOTN
         XCT QTONOS
         TRA CLIPOD+1

FLEC4:   PUSHJ$ HYPOTN
         PUSHJ$ CLIVIS
         TRA CLIVIS+2

MEDIA4:  MOVEI TONOS,PHRYGI+2
         PUSHJ$ PODAT
         NEST |(TORCU)PODAT+1|
         POPJ$

FINI4G:  PUSHJ$ HYPOTN
         XCT QTONOS
         TRA CLIVIS

FINI4E:  PUSHJ$ INCI4
         AOS TONOS
         PUSHJ$ REGIAS
         TRA AMEN+1

;                       EXAMPLE 2.1.5
```

Little needs to be said about the actual encoding of the psalm that is not obvious from the preceding analysis. A table is used to keep track of the syllables of incantation and to set a flag which determines whether or not the flex is to be sung (XFLAG). (The flex is only sung for those verses marked by a dagger.) Likewise, a flag is set to delimit the introductory portion of the antiphon (IFLAG). The program begins with a quarter note rest so that all subsequent pitches with unspecified duration are interpreted as quarter notes.

```
TITLE PSALM 146 WITH ANTIPHON

.INSRT EUTERP >
.INSRT PT4 >            ;LOAD PSALM TONES

XFLAG=5

LOC VOICE1

        R 4T
        PUSHJ$ INCI4       ;INTONATION
        SETZM 4            ;4 IS INDEX REGISTER
VERSE:  IRP X,,|1,2,3,XFLAG|      ;LOAD AC'S 1,2, AND 3 WITH
                                  ;SYLLABLE COUNTS
                MOVE X,SYTAB(4)   ;XFLAG IS SET TO ZERO
                                  ;WHENEVER FLEX IS
                AOS 4             ;OMITTED
        TERMIN
        PUSHJ$ PSALM       ;PLAY A VERSE
        SKIPE SYTAB(4)     ;ZERO IF LAST VERSE
        TRA VERSE
        FINE

PSALM:  XCT QHYPOTN        ;REPEAT TENOR
        SOJG 1,.-1
        SKIPE XFLAG        ;SKIP FLEX IF NECESSARY
        PUSHJ$ FLEC4       ;FLEX
        XCT QHYPOTN        ;REPEAT TENOR
        SOJG 2,.-1
        PUSHJ$ MEDIA4      ;MEDIATION
        R                  ;PAUSE AFTER MEDIATION
        XCT QHYPOTN        ;REPEAT TENOR
        SOJG 3,.-1
        PUSHJ$ FINI4E      ;CADENCE
        REPEAT 2 R         ;LONGER PAUSE AFTER CADENCE
        POPJ$
```

```
SYTAB:  5           ;SYLLABLE COUNTS FOR EACH VERSE
        7           ;FIRST VERSE
        6
        1           ;MEDIATION IN FIRST VERSE
        4           ;SECOND VERSE
        4
        8.
        0           ;NO MEDIATION
        8.          ;THIRD VERSE
        6
        13.
        0           ;NO MEDIATION
        12.         ;FOURTH VERSE

        9.
        12.
        1           ;MEDIATION
        9.          ;FIFTH VERSE
        10.
        3
        1           ;MEDIATION
        3           ;SIXTH VERSE
        2
        3
        0           ;NO MEDIATION
        3           ;SEVENTH VERSE
        3
        3
        0           ;NO MEDIATION
        6           ;EIGHTH VERSE
        7
        3
        1           ;MEDIATION
        10.         ;NINTH VERSE
        2
        8.
        0           ;NO MEDIATION

END TUNE
```

; EXAMPLE 2.1.6

2.2. Strophic Chant

The antiphon of the preceding psalm setting is an example of neumatic chant generally consisting of one neume per syllable.

Such chant is also used in the setting of hymns. Hymn settings usually involve somewhat longer melodies, called strophes, which are strictly repeated for each verse of the hymn text.

A single strophe encompasses a melodic unit which is called a period. In studying the Gregorian period, we may again benefit from the observations of Dom Paolo [Ferretti]:

> In every period, and in every element of a period, we can and must consider two things: 1) The form of its line, i.e. the melodic design, the periodic architecture; 2) The logical internal correspondence among the diverse elements of which it is composed.

This may best be illustrated by an example; and, once again, we turn to the Antiphonale Romanum:

2. Verusque sol illabere,
Micans nitore perpeti:
Jubarque Sancti Spiritus
Infunde nostris sensibus.

3. Votis vocemus et Patrem,
Patrem potentis gratiae,
Patrem perennis gloriae:
Culpam releget lubricam.

4. Confirmet actus strenuos:
Dentes retundat inridi:
Casus secundet asperos,
Agenda recte dirigat.

5. Mentem gubernet et regat,
Sit pura nobis costitas:
Fides calore ferveat,
Fraudis venena nesciat.

6. Christusque nobis sit cibus,
Potusque noster sit fides:
Laeti bibamus sobriam
Profusionem spiritus.

7. Laetus dies hic transeat:
Pudor sit ut diluculum,
Fides velut meridies:
Crespuculum mens nesciat.

8. Aurora lucem prorehit:
Cum luce nobis prodeat
In Patre totus Filius,
Et totus in Verbo Pater.

9. Deo Patri set gloria,
Eiusque soli Filio,
Cum Spiritu Paraclito,
Nunc et per omne saeculum. ([AR])

Example 2.2.1

The fundamental period, which is sung nine times for the nine verses of the hymn, divides into four phrases, the last of which is an exact repetition of the first. Furthermore, the second phrase differs from the first only in its cadence, while the third phrase has the same cadence as the first (an instance of what is commonly called "musical rhyme"). Thus we have three phrase routines each of which break down into two members. The member routines consist of the opening figure of the first phrase, the cadence of the first phrase, and the opening figure of the third phrase. The cadence of the second phrase is simply a repeated tone.

Because this hymn is in the hypophrygian mode, we may again use the template designed for the second maneria. Here is a program for it:

```
          R 4T
          REPEAT 9. PUSHJ$ PERIOD
          FINE

PERIOD:   PUSHJ$ PHRAS1
          PUSHJ$ PHRAS2
          PUSHJ$ PHRAS3
          PUSHJ$ PHRAS1
          R
          POPJ$

MEMBR1:   PUSHJ$ FINAL
          XCT @TONOS
          SUBI TONOS,2
          PUSHJ$ SALIC
          PUSHJ$ CLIPOD+1
          POPJ$

MEMBR2:   AOS TONOS
          PUSHJ$ CLIVIS
          PUSHJ$ CLIVIS+1
          POPJ$

MEMBR3:   XCT @TONOS
          PUSHJ$ PODAT
          XCT @TONOS
          SOS TONOS
          PUSHJ$ CLIVIS+1
          POPJ$

PHRAS1:   PUSHJ$ MEMBR1
          PUSHJ$ MEMBR2
          R
          POPJ$

PHRAS2:   PUSHJ$ MEMBR1
          AOS TONOS
          REPEAT 3 XCT @TONOS
          R
          POPJ$

PHRAS3:   PUSHJ$ MEMBR3
          PUSHJ$ MEMBR2
          R
          POPJ$

;
```

EXAMPLE 2.2.2

2.3 Commatic Chant

A form which is freer than the hymn but which still maintains a strophic character is the _prosa_ or _sequence_. Instead of consisting in a single strophe which is strictly repeated for the duration of the chant, the prosa is composed of several strophes which are each repeated once. The "classical" form of the sequence may be expressed symbolically in the form, a bb cc dd . . . jj k; however, like all forms, this is not always strictly obeyed. For example, the unrepeated strophes are often absent; and even the repeated pairs may sometimes be interrupted, ([Reese]). The progression from one strophe to the next in a prosa is relatively free compositional process, although there are many aspects which link the strophes together and lend a unity to the entire composition. This is a basic feature of original Gregorian melody, and it is worth considering [Ferretti]'s commentary on the actual techniques which are employed to this end:

> Concerning the musical period . . . we remark that _musical discourse_ demands that its parts be bound by intimate and reciprocal agreements, such that the openings prepare and call the later parts which reply and complete them. This logical necessity gives rise to all those artifices which have received the name _demand_ and _response_: _antecedent_ and _consequent_; _rhyme_; _identical_ or _reverse movements_; _ascending_ and _descending melodic progressions_; _thematic recollections_; _imitations_; etc.

For example, let us consider the Lauda Sion prosa, which appears in the Liber Usualis. This is a rather late work, having been written by St. Thomas Aquinas (d. 1274) ([Reese]). It also has a strongly rhythmical text, a feature which is not typical of earlier prosae. However, since we are only interested in matters of melodic development, this is a relatively suitable example.

Here are the melodies for the first two strophes (i.e. the first two pairs of verses) ([Liber]):

Example 2.3.1

The second strophe possesses many of the traits of a consequent to the antecedent first strophe. Each phrase of the latter is a development of the corresponding phrase of the former, with a final resolution consisting of an exact repetition of the final phrase.

In preparing a EUTERPE program for this prosa, we need a template for the fourth maneria similar to that for the second maneria used the preceding examples. Some of the routines from the second maneria appear again in this template, and there are a few additional routines of compound neumes:

```
TITLE TEMPLATE FOR FOURTH MANERIA

.INSRT NEUMES >              ;LOAD GAMUT AND NEUMES

FINAL:   MOVEI TONOS,MILY
         POPJ$

TENOR:   MOVEI TONOS,MILY+4
         POPJ$

TORCUL:  PUSHJ$ PES
         SOS TONOS
         TRA CLIVIS+1

SALI:    PUSHJ$ PES
         TRA PODAT+1

TORMOD:  XCT TORCU
         TRA TORCUL+1

CLIPOD:  PUSHJ$ CLIVIS
         SOS TONOS
         TRA PODAT

CLISAL:  PUSHJ$ CLIVIS
         NEST |(SALIC)PODAT+1|
         POPJ$

SANCTU:  XCT @TONOS
         TRA CLIPOD

CODA:    PUSHJ$ TORCU
         TRA CLIVIS+1

CHCODA:  PUSHJ$ REGIAS
         PUSHJ$ CLIPOD+1
         TRA CLIVIS+2

AMEN:    PUSHJ$ CODA
         XCT @FINAL
         POPJ$

;
```

EXAMPLE 2.3.2

Now let us consider the EUTERPE programs for these two strophes:

First strophe

```
V1P1:   SUBI TONOS,2
        PUSHJ$ SALI
        PUSHJ$ TORCU+1
        ADDI TONOS,3
        PUSHJ$ REGIAS
        POPJ$

V1P2:   AOS TONOS
        PUSHJ$ TORMOD
        SUBI TONOS,2
        PUSHJ$ CODA
        PUSHJ$ CLISAL+1
        POPJ$

V1P3:   PUSHJ$ V13P3+2
        PUSHJ$ CHCODA+1
        POPJ$
```

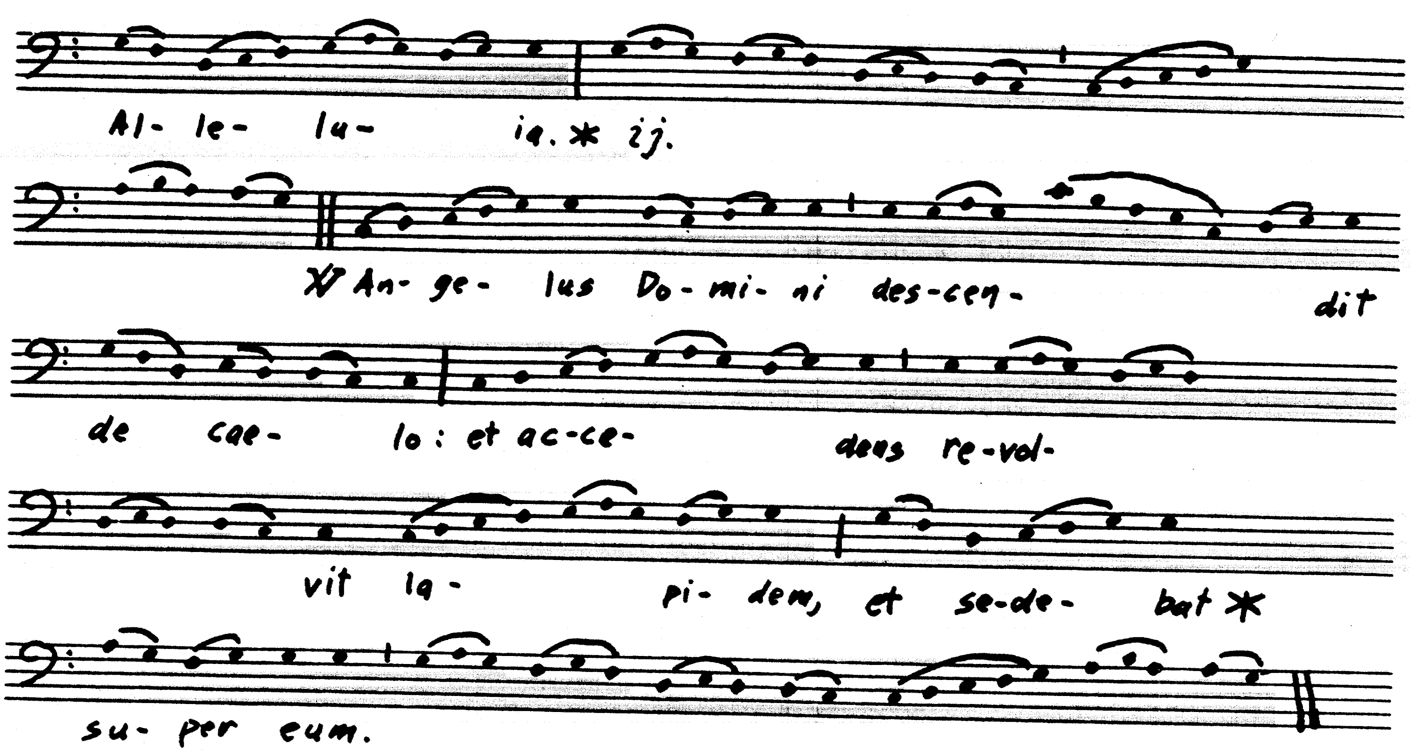
Second strophe

```
V3P1:   AOS V1P1+3
        NEST [(TORCUL)PES+1]
        NEST [(V1P1)SALI+1]
        PUSHJ$ CLIPOD+1
        PUSHJ$ PODAT+1
        SOS V1P1+3
        POPJ$

V3P2:   PUSHJ$ CLIVIS
        PUSHJ$ PES+1
        SOS TONOS
        EXIT1 V1P244
        NEST [(V1P2)TORMOD+1]
        POPJ$

V3P3:   PUSHJ$ FINAL
        TRA V1P3
```

Example 2.3.3

Most of the developmental processes are realized through programming the latter phrases to nest into their antecedents. The phrase V3P1 picks up the last five notes of V1P1 by this method, although a modification is necessary to enlarge the interval before the descending REGIAS routine. This is accounted for by the instruction AOS V1P1+3. The instruction at V1P1+3 raises TONOS by the interval of a fourth; this AOS increases the interval by an extra scale degree. Similarly, the last five notes of V3P2 are extracted from V1P2 by a NEST and an EXIT1. Furthermore, the final phrase not only completes the consequent by its exact repetition but also sets up an instance of musical rhyme which pervades the entire sequence. Indeed, the whole antecedent-consequent relationship hangs on the proper manipulation of thematic recollection and appropriate imitation with alteration, i.e. appropriate subroutine extraction via nesting and exiting.

For examples of identical and reverse movements, as well as ascending and descending melodic progressions in general, we turn to another original melody. This is the Alleluia for the Easter Monday Mass. The chant (again in the fourth <u>maneria</u>) and corresponding program are reproduced below ([Liber]):



Example 2.3.4

```
TITLE ALLELUIA ANGELUS DOMINI

.INSRT EUTERP >
.INSRT TETRA >              ;LOAD TEMPLATE FOR FOURTH MANERIA

LOC VOICE1                  ;SOLOIST

        R 4T    ;SET RHYTHMIC UNIT AS QUARTER NOTE FOR BOTH VOICES
        CANCEL 2,
        PUSHJ$ FINAL        ;SET FINAL
        SETZM FLAG'
PAL:    PUSHJ$ ALLELU       ;ALLELUIA
        R
        TRA 2,VOICE2+1      ;CHORUS SINGS JUBILUS
        CANCEL

VERSE:  IRP X,,|1,2|        ;VERSE BROKEN INTO SUBROUTINE SEGMENTS
                IRP Y,,|1,2|
                        IRP Z,,|1,2|
                                PUSHJ$ M!X!P!Y!!!Z
                                R
                        TERMIN
                TERMIN
        TERMIN
        PUSHJ$ M2P311
        R
        TRA 2,JUBILU        ;RECAP OF JUBILUS AT END OF VERSE SUNG BY
        CANCEL                          ;CHORUS

LOC VOICE2          ;CHORUS

        TRA VOICE1
        SKIPN FLAG          ;REPEAT ALLELUIA FIRST TIME AROUND
        TRA .+3
        PUSHJ$ CHORUS+2
        FINE                ;DONE AFTER SECOND CHORUS
        PUSHJ$ CHORUS
        TRA 1,VERSE         ;SOLOIST SINGS VERSE
        CANCEL

JUBILU: PUSHJ$ M2P312
        PUSHJ$ CHORUS+1
        SETOM FLAG
        TRA 1,PAL
        CANCEL

CHORUS: PUSHJ$ ALLELU
        R
        IRP X,,|1,2|
                PUSHJ$ CAUDA!X
```

```
                R
        TERMIN
        R
        POPJ$

ALLELU: PUSHJ$ CLIVIS
        SUBI TONOS,2
        EXIT TOP
        PUSHJ$ M1P111+1
        UNEXIT TOP
        NEST |(CLIPOD)CLIVIS+1|
        TRA TOP

CAUDA1: PUSHJ$ TORCU
        SOS TONOS
        PUSHJ$ TORCU
        SUBI TONOS,2
        PUSHJ$ TORCU
        PUSHJ$ CLIVIS
        POPJ$

CAUDA2: EXIT TOP
        PUSHJ$ M1P111
        UNEXIT TOP
        AOS TONOS
        TRA CAUDA1+4

M1P111: MOVEI TONOS,HYMILY-1
        PUSHJ$ PODAT
        REPEAT 3 PUSHJ$ PODAT+1
TOP:    XCT @TONOS
        POPJ$

M1P112: PUSHJ$ CLIVIS+1
        SUBI TONOS,2
        TRA M1P111+2

M1P211: XCT @TONOS
        PUSHJ$ TORCU
        ADDI TONOS,3
        PUSHJ$ REGIAS
        SUBI TONOS,3
        TRA M1P111+2

M1P212: EXIT ALLELU+2
        PUSHJ$ ALLELU
        UNEXIT ALLELU+2
        PUSHJ$ CAUDA1+4
        TRA TOP
```

```
M2P111: XCT @TONOS
        AOS TONOS
        TRA ALLELU+2

M2P112: POPJ$

M2P211: XCT @TONOS
        PUSHJ$ CAUDA1
        TRA TOP

M2P212: TRA M2P111

M2P311: EXIT ALLELU+2
        PUSHJ$ ALLELU
        UNEXIT ALLELU+2
        EXIT TOP-1
        PUSHJ$ ALLELU+3
        UNEXIT TOP-1
        TRA TOP

M2P312: AOS TONOS
        PUSHJ$ CLIPOD
        XCT @TONOS
        TRA TOP

END TUNE

;                           EXAMPLE 2.3.5
```

In this chant the text, "Alleluia," is followed by a two-phrase *jubilus*. The verse consists of three members, the third of which is drawn from the "Alleluia" portion. The first two members each divide into two phrases, each of which divides into two incises. (The first phrase of the second member actually has only one incise; however, in the program we interpret the "second" incise as a null routine.) This instance of "recapitulation" at the end of the verse is not typical of "Alleluiatic" melodies but is a structural feature worth observing.

Most of the movement of this chant either rises or falls with little change of motion *en route*. The rising line is presented in its plainest form as the first incise of the verse; the falling line is the familiar REGIAS routine which accompanies (appropriately enough) the text, "descendit." The *jubilus* begins with a descending progression of *torculi*, achieved by repeated calls with successive subtraction from the TONOS. The ascending line is simpler in nature and is accomplished by repeated calls into PODAT.

As explanation of the general layout of this program, we offer the following note by [Reese]:

> The "classic" manner of performance is as follows: The soloist sings the word *Alleluia* up to the melisma (or *cauda* or *jubilus*) added to the last syllable. (The point at which the *cauda* begins is marked by an asterisk in modern chant books.) The choir repeats what the soloist has sung and adds the melisma. The soloist sings the verse up to the point marked by an asterisk in

> modern chant books, and the choir continues it from there on. After its conclusion the soloist sings the portion of the Alleluia he had sung at the beginning and the choir, without repeating this as at first, continues with the melisma, with which the performance ends.

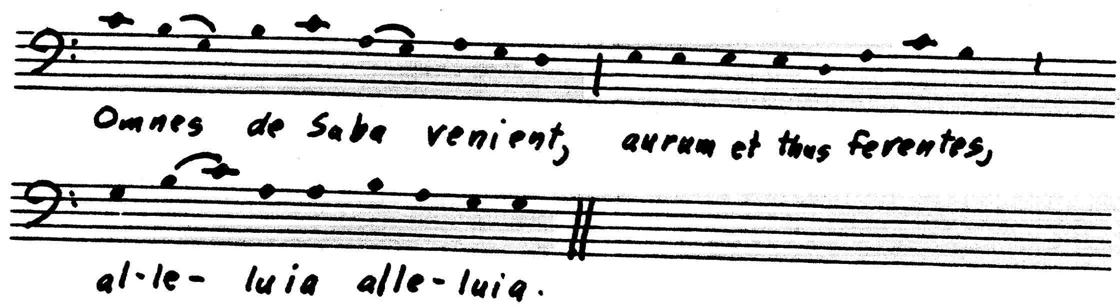
The parts of the soloist and the choir are allotted in this program to two distinct voices which never sound simultaneously. Transitions are accomplished at the appropriate locations by a TRA instruction followed by a CANCEL to silence the currently sounding voice. FLAG is set to enable the chorus to pick up after the soloist during the repetition of the "Alleluia."

Not all commatic chants are original melodies; originality was not a particularly important criterion in the composition of chant. In fact, there are two approaches to commatic chant which are founded solely on processes of adaptation. These are composition using melody-types and composition by centonization ([Ferretti]).

Composing with melody-types is described by [Ferretti] as follows:

> The artist does not create a new melody; he uses a traditional air which he takes as a model -- as a type -- and which he applies and adapts to a new text, introducing modifications and variations which this text, be it longer or shorter, necessitates. Here the melody has an expression independent of the text and of purely musical value; its beauty is thus autonomous, intrinsic, and transcendental.
>
> In these melody-types, however, the artist is not free to dispose of the musical elevations and depositions wherever and however he wishes; he is bound by the exigencies of the notes or by groups of notes which demand a tonic accent in the text. In this respect, there is a great difference between these chants and those of strophic form.

To illustrate this process, we consider the melody-type embodied in the antiphon Omnes de Saba venient. This chant, included in the Antiphonale Romanum for the fifth day after Epiphany, is reproduced below ([AR]):

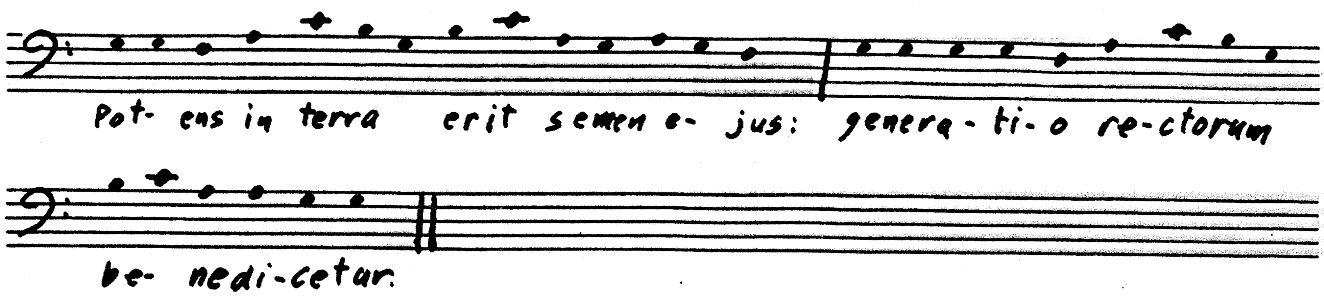


Example 2.3.6

This chant is in the fourth maneria; it consists of two members, each of which has a coda figure. It may be encoded in EUTERPE as follows:

```
OMNES:  PUSHJ$ M1
        R
        PUSHJ$ M2
        REPEAT 2 R
        START 2,POTENS
        CANCEL

M1:     PUSHJ$ FINAL
        ADDI TONOS,3
        PUSHJ$ CLIVIS
        SUBI TONOS,2
        PUSHJ$ SALI
        PUSHJ$ TORCUL+1
        SOS TONOS
COD:    PUSHJ$ TORCU
        PUSHJ$ CLIVIS+1
        POPJ$

M2:     PUSHJ$ FINAL
        REPEAT 4 XCT @TONOS
        SOS TONOS
        PUSHJ$ PES
        ADDI TONOS,2
        EXIT1 COD-1
        PUSHJ$ M1+2
COD1:   PUSHJ$ COD
        XCT @FINAL
        POPJ$

;                          EXAMPLE 2.3.7
```

Now consider the antiphon _Potens in terra_, reproduced by [Ferretti] from the Hartker Antiphonal and transcribed by the author below:



Example 2.3.8

The first member has a _prosthesis_, i.e. an introductory passage to accomodate a longer text. This prosthesis is such as to give the first member the same melodic character as the second, but cadencing on the coda figure of the first member. The second member, on the other hand, follows closely the model of the _Omnes de Saba venient_ chant, but with an abbreviated coda. Here is a program for _Potens in terra_ which is obtained entirely by referencing the program for _Omnes de Saba venient_:

```
POTENS: PUSHJ$ MM1
        R
        PUSHJ$ MM2
        FINE

MM1:    PUSHJ$ FINAL
        EXIT1 COD1
        PUSHJ$ M2+3
        TRA COD-1

MM2:    EXIT1 COD1
        PUSHJ$ M2
        XCT @TONOS
        NEST |(COD1)COD+1|
        POPJ$

;
```

EXAMPLE 2.3.9

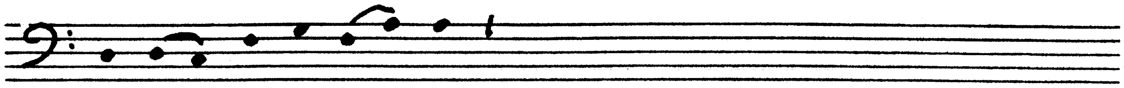
[Ferretti] makes the cautionary remark that the chronology of the roughly thirty antiphons composed on this one melody-type is unknown. We selected this particular order of "derivation" because the latter appeared to be an expansion of the former. Thus, it seemed more reasonable to compose the second to recover the first.

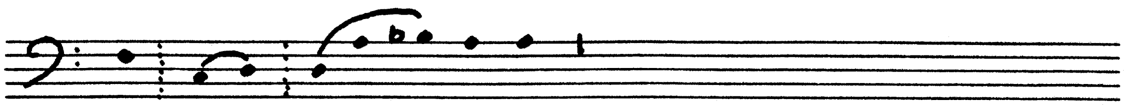
The process of centonization is closely related to our technique of melodic development through the excerpting of a broad repertoire of subroutines. Centonized melodies arise by piecing together fragments from other melodies. [Ferretti] describes the technique as follows:

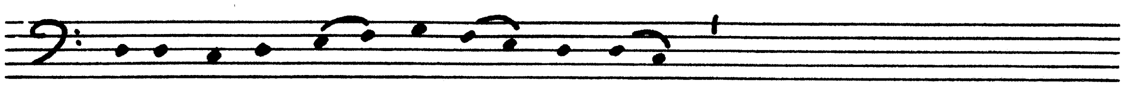
> In this process, one does not apply a traditional melody to alternative texts, but one takes a certain number of formulas belonging to a certain modal and melodic type from the traditional musical foundation, and one joins them, one links them, in order to make an organic, homogeneous, logical whole. At first glimpse, the melody thus obtained appears to be created at one stroke; one would call it original, in that it is natural and harmonious in all its parts; while in reality, it is a veritable mosaic. One may observe of this type of composition, as one did in the case of melody-types, that the expression has a value exclusively musical, not deriving directly from the text; and in each formula there are notes and groups of notes which demand the tonic accent. Furthermore, the different formulas are not at all united by chance; their joining is based on precise rules which one must indispensibly observe. Finally, one must know that these formulas have logical connections by dint of which they respond to each other. Thus, the artist does not play with absolute liberty; he is held by the logic and musical exigencies of the formulas.

[Ferretti] has compiled the following list of formulas for centonized antiphons in the first (dorian) mode; these are grouped into three classes: opening formulas, central formulas, and closing formulas:

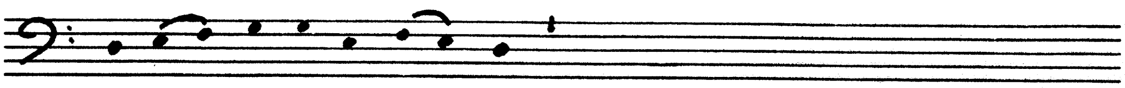
A) Opening formulas

1. 

2. 

3. 

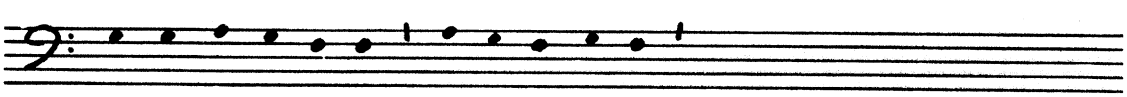
4. 

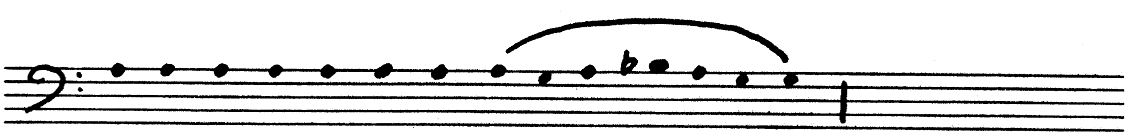
5. 

B) Central formulas

6. 

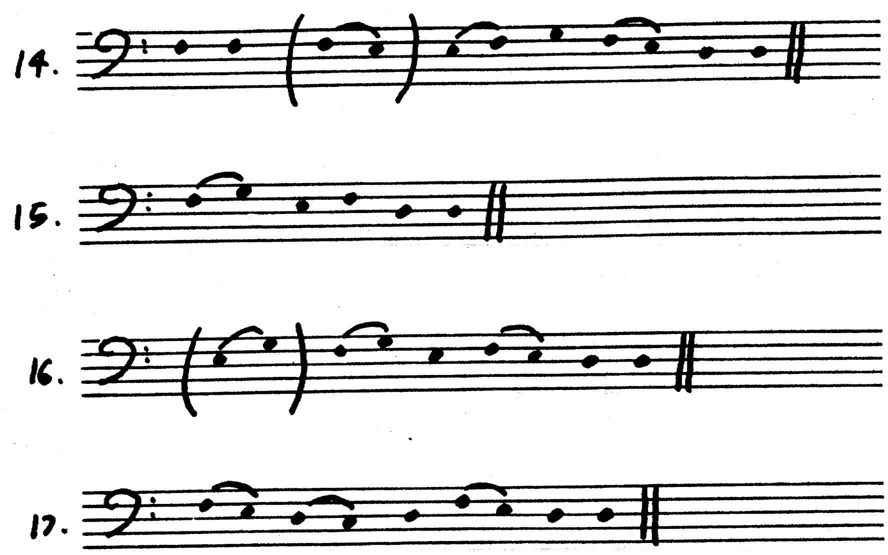
7. 

8. 

9. 


10. a.
b.
c.
11. a.
b.
12. a.
b.
13.

c) Closing formulas



Example 2.3.10

Since these are in the first mode, we need a template for the first maneria. This template is much like the others we have designed, except that the dorian mode has occasions in which the B might be flatted. Hence, there are routines, FLAT and UNFLAT, which alter the scale appropriately:

```
TITLE TEMPLATE FOR FIRST MANERIA
.INSRT NEUMES >           ;LOAD GAMUT AND NEUMES
FINAL:  MOVEI TONOS,DORIAN
        POPJ$
TENOR:  MOVEI TONOS,DORIAN+4
        POPJ$
FLAT:   MOVEI 16,I B FL
        MOVEM 16,HYPODO+1
        ADDI 16,I
        MOVEM 16,DORIAN+5
        POPJ$
UNFLAT: MOVEI 16,I B
        TRA FLAT+1
;                      EXAMPLE 2.3.11
```

We are now in a position to encode Ferretti's formulas. These can also be stored as a template which may then be referenced by the actual programs of centonized melodies:

```
TITLE CENTONIZATION FORMULAS FOR FIRST MODE
.INSRT PROTUS >          ;LOAD FORMULAS FOR FIRST MANERIA

A1:     PUSHJ$ FINAL
        XCT @TONOS
        PUSHJ$ CLIVIS
        ADDI TONOS,3
        PUSHJ$ TORCU
        PUSHJ$ PES+1
        TRA CLIVIS+2

A2:     PUSHJ$ FLAT
        XCT @DORIAN+2
        MOVEI TONOS,DORIAN-1
        PUSHJ$ PODAT
        PUSHJ$ FINAL
        XCT @TONOS
        ADDI TONOS,4
        PUSHJ$ TORCU
        TRA CLIVIS+2

A3:     EXIT A1+3
        PUSHJ$ A1
        UNEXIT A1+3
        AOS TONOS
        PUSHJ$ VISIO
        NEST |(REGIAS)CLIVIS+1|
        TRA CLIVIS

A4:     EXIT A5+2
        PUSHJ$ A5
        UNEXIT A5+2
        PUSHJ$ CLIVI+1
        NEST |(SALI)PES+1|
        PUSHJ$ TORMOD+1
        PUSHJ$ REGIAS+1
        PUSHJ$ TORCUL
        PUSHJ$ CLIVIS
        NEST |(TORCU)PODAT+1|
        PUSHJ$ PODAT+1
        TRA CLIVIS+2
```

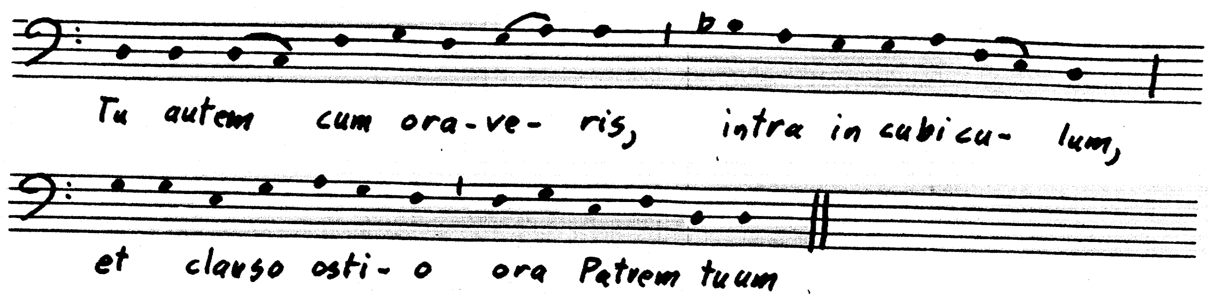
```
A5:     PUSHJ$ FINAL
        PUSHJ$ VISIO
        PUSHJ$ CLIVI
        NEST |(TORCU)PODAT+1|
        TRA CLIVIS+1

B6:     PUSHJ$ FLAT

        PUSHJ$ TENOR
        PUSHJ$ ELEGI
        PUSHJ$ CLIVIS+1
        TRA ELEGI

B7:     MOVEI TONOS,DORIAN+3
        REPEAT 2 XCT @TONOS
        SUBI TONOS,2
        PUSHJ$ SALI
        PUSHJ$ REGIAS+1
        TRA PODAT

B8:     EXIT B7+3
        PUSHJ$ B7
        UNEXIT B7+3
        EXIT B7+6
        NEST |(B7)SALI+1|
        XCT @TONOS
        R
        ADDI TONOS,2
        NEST |(B7)SALI+1|
        UNEXIT B7+6
        NEST |(TORCU)PODAT+1|
        POPJ$

B9:     PUSHJ$ FLAT
        PUSHJ$ TENOR
        REPEAT 8. XCT @TONOS
        SOS TONOS
        NEST |(VISIO)PODAT+2|
        PUSHJ$ REGIAS+1
        TRA CLIVIS+2

B10A:   PUSHJ$ TENOR
        EXIT VISIO+2
        PUSHJ$ B9+7
        UNEXIT VISIO+2
        PUSHJ$ CLIVI
        TRA REGIAS+1

B10B:   PUSHJ$ FLAT
        MOVEI TONOS,DORIAN+5
        NEST |(REGIAS)CLIVIS+2|
        EXIT PODAT+2
        PUSHJ$ PODAT
        UNEXIT PODAT+2
        TRA B10A+4

B10C:   MOVEI TONOS,DORIAN+2
        EXIT B9+13.
```

```
        PUSHJ$ B9+11.
        UNEXIT B9+13.
        AOS TONOS
        TRA B10B+3

B11A:   PUSHJ$ FINAL
        PUSHJ$ CLIVIS
        MOVEI TONOS,DORIAN+2
        PUSHJ$ B9+11.
        AOS TONOS
        TRA CLIVIS

B11B:   MOVEI TONOS,DORIAN+2
        PUSHJ$ MODTOR
        AOS TONOS
        NEST |(B10A)CLIVI+2|
        POPJ$

B12A:   MOVEI TONOS,DORIAN+2
        PUSHJ$ CLIVI
        ADDI TONOS,2
        EXIT A3+6
        NEST |(A3+4)PODAT+2|
        UNEXIT A3+6
        PUSHJ$ VISIO+1
        XCT @FINAL
        PUSHJ$ REGIAS+1
        PUSHJ$ VISIO+1
        TRA CLIVIS+2

B12B:   EXIT1 B10B+6
        PUSHJ$ B10C
        SUBI TONOS,4
        NEST |(VISIO)PODAT+2|
        POPJ$

B13:    MOVEI TONOS,DORIAN-1
        PUSHJ$ SALIC
        REPEAT 4 XCT @TONOS
        TRA TORCU

C14:    MOVEI TONOS,DORIAN+2
        REPEAT 2 XCT @TONOS
        PUSHJ$ CLIVIS
        MOVEI TONOS,DORIAN+1
        EXIT A3+6
        NEST |(A3+4)PODAT+2|
        TRA CLIVIS+2

C15:    MOVEI TONOS,DORIAN+2
```

```
        PUSHJ$ TORMOD
        NEST |(TORMOD)PODAT+1|
        TRA CLIVIS+2

C16:    MOVEI TONOS,DORIAN+1
        PUSHJ$ PES
        EXIT C15+2
        PUSHJ$ C15
        UNEXIT C15+2
        AOS TONOS
        NEST |(C14)A3+5|
        POPJ$

C17:    MOVEI TONOS,DORIAN+2
        PUSHJ$ REGIAS
        NEST |(SALIC)PODAT+1|
        NEST |(C14)REGIAS+1|
        POPJ$

;
```

EXAMPLE 2.3.12

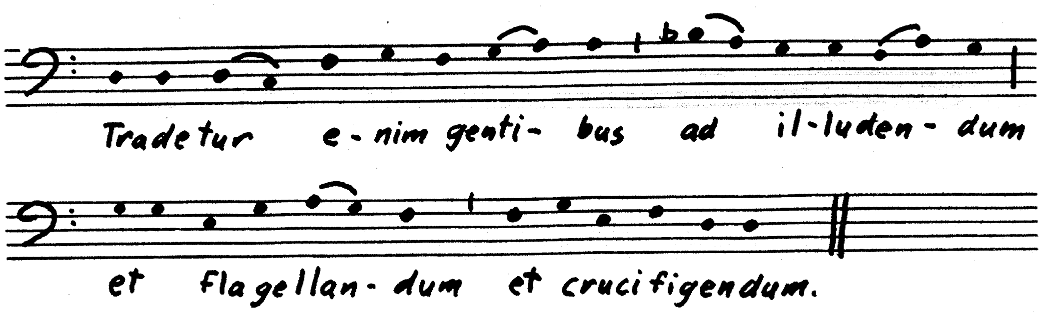
Given this template and the template for the first maneria, programs of centonized melodies are extremely simple in form. Here are three antiphons from the Antiphonale Romanum and their representative programs ([AR]):



Example 2.3.13

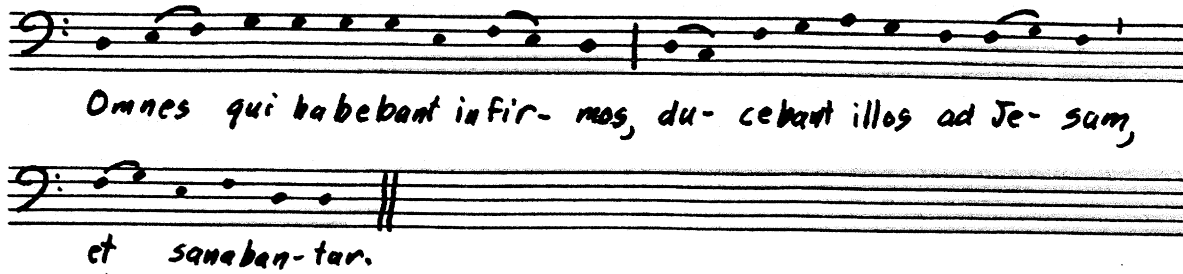
```
XCT QFINAL
EXIT1 A1+5
PUSHJ$ A1
AOS TONOS
PUSHJ$ PODAT
PUSHJ$ A1+6
R
PUSHJ$ B10B
R
EXIT1 B7+6
PUSHJ$ B7
R
PUSHJ$ C15
FINE
```

EXAMPLE 2.3.14




```
PUSHJ$ FINAL
XCT @TONOS
EXIT1 A1+5
PUSHJ$ A1+1
AOS TONOS
PUSHJ$ PODAT
PUSHJ$ A1+6
R
EXIT1 B6+2
PUSHJ$ B6
SOS TONOS
MEST |(B6)ELEQ|+1|
R
EXIT1 B7+6
PUSHJ$ B7
R
PUSHJ$ C15
FINE
```

EXAMPLE 2.3.10



Example 2.3.17

```
EXIT1 A5+2
PUSHJ$ A5
REPEAT 2 XCT GTONCS
PUSHJ$ A5+2
R
PUSHJ$ B11A
R
PUSHJ$ C15
FINE
```

; EXAMPLE 2.3.18

Centonization provides a foundation for some rather simple, but nevertheless effective, experiments in computer composition of music. Centonized antiphons are represented by an abstract model which postulates an opening formula, one or more central formulas, and a closing formula. In addition, [Ferretti] cites the following rules concerning the ordering of these formulas:

6 must follow 1

7 must follow 6 or 10b

8 must follow 10 or 12a

9 must follow 2

3 is followed by 11 and takes 15 as final

5 is followed by 11 and takes 15 as final

4 is followed by 11 and takes 14 as final

13 is followed by 14 or 17

It should now be possible to apply these rules to the model to obtain a procedure which would yield melodies similar in nature to Gregorian chant. This does not necessarily imply a "simulation" of Gregorian chant; such would be impossible in the absence of a stipulated text. Rather, the procedure is a compositional technique which is not tied down to the note-by-note approach used by [Hiller] and Isaacson in their "cantus firmus generator," but is similar in nature to Mozart's "dice composer" ([Mozart, 1941]) with greater flexibility in determining the lengths of its phrases.

The current version of this program is in an early stage of development. While it works within the limitations of the rules outlined above, it is currently handling formulas only in their entirety. However, as we have seen from the above examples, even in centonization, truncations by nesting and exiting play an important role. These will be implemented into this program in a future version.

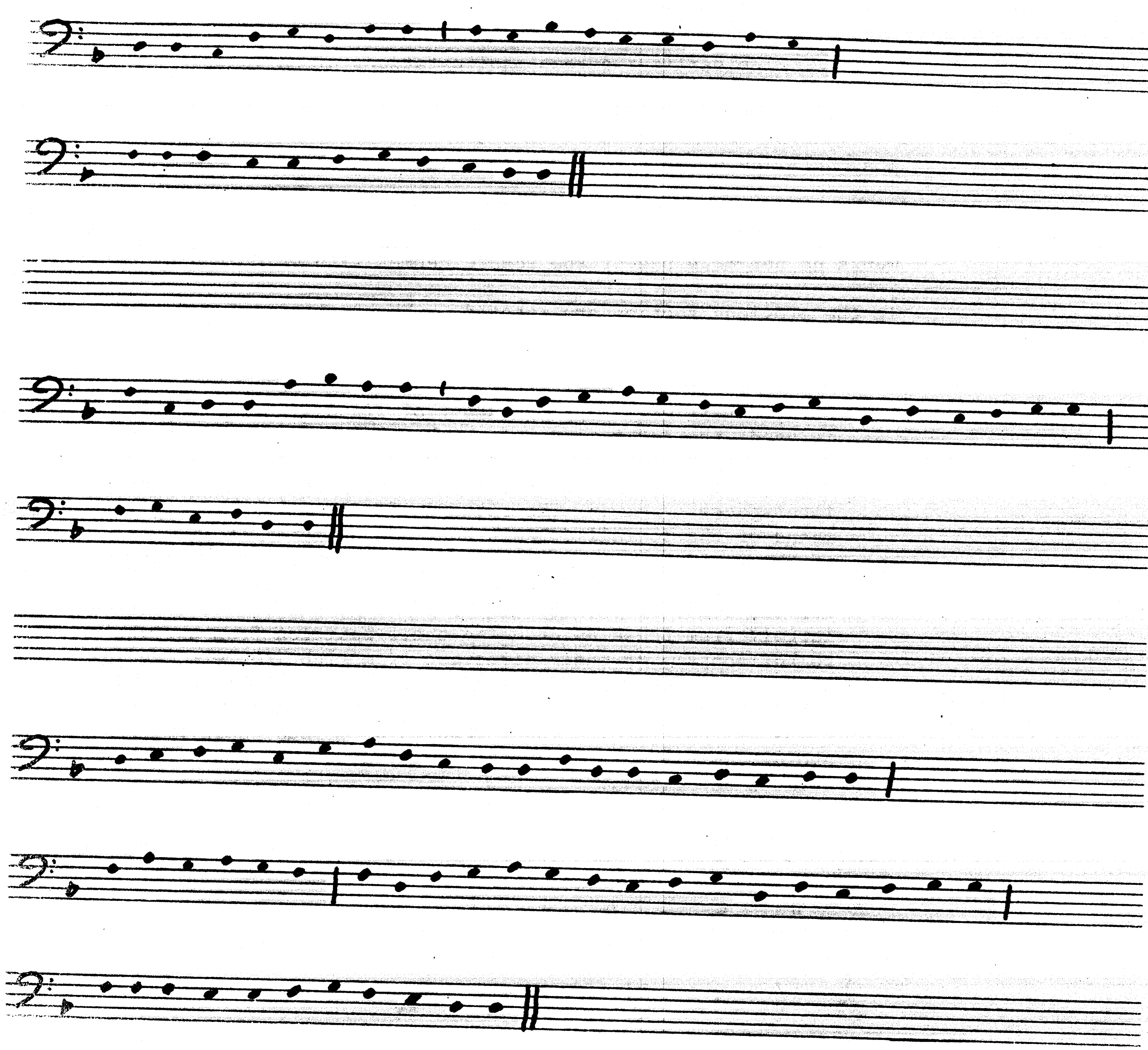
```
TITLE CENTONIZER

.INSRT EUTERP >
.INSRT CENTO1 >          ;LOAD CENTONIZATION FORMULAS

LOC VOICE1

OPEN:    R 4T
         SETZM FINSAV'
         MOVEI 1,STACK-1
         RAN 2
         ANDI 2,7
         JUMPE 2,.-2
         CAILE 2,2
         TRA .+4
         ADD 2,|@CFLIST|
         PUSH 1,2          ;PUSH A1 OR A2 AND GET CENTRAL FORMULA
         TRA CENTRA
         CAIE 2,4
         TRA FIN15
         MOVEI 3,14.       ;A4 TAKES C14 AS FINAL
         MOVEM 3,FINSAV
PUSHO:   ADD 2,|@CFLIST|
         PUSH 1,2          ;PUSH A3, A4, OR A5
         RAN 2   ;CHOOSE A B11
         ANDI 2,1
         AOS 2
         ADD 2,|@B11|
         PUSH 1,2
         RAN 2   ;ANOTHER CENTRAL FORMULA?
         ANDI 2,1
         JUMPE 2,CENTRA
         TRA CLOSE
FIN15:   CAILE 2,5
         TRA OPEN+2
         MOVEI 3,15.       ;A3 AND A5 TAKE C15 AS FINAL
         MOVEM 3,FINSAV
         TRA PUSHO
CENTRA:  RAN 2
         ANDI 2,7
         ADDI 2,6          ;NUMBER FROM 6 TO 13
         CAIE 2,13.
         TRA NOT13
         MOVE 3,FINSAV
         CAMN 3,15.        ;B13 WONT TAKE C15 AS FINAL
         TRA CENTRA
         ADD 2,|@CFLIST|
         PUSH 1,2          ;PUSH B13
         JUMPE 3,.+6       ;CHOOSE IF NO FINSAV
PUSHC:   ADD 3,|@CFLIST|
```

```
                PUSH 1,3
                SETZM 2 ;PUSH ZERO AT END OF STACK
                PUSH 1,2
                TRA PLAYIT
                RAN 2    ;CHOOSE BETWEEN C14 AND C17
                ANDI 2,1
                JUMPE 2,.+3
                MOVEI 3,14.
                TRA PUSHC
                MOVEI 3,17.
                TRA PUSHC
NOT13:          CAIGE 2,10.        ;B10, B11, AND B12 HAVE OPTIONS
                TRA NOP
                CAIN 2,11.
                TRA PUSHO+2        ;B11 ALREADY WORKED OUT
                CAIE 2,12.
                TRA .+5
                RAN 2
                ANDI 2,1           ;B12 ALSO BINARY CHOICE
                ADD 2,|QB12|
                TRA PUSHO+6
                RAN 2
                ANDI 2,3
                JUMPE 2,.-2        ;THREE CHOICES FOR B10
                ADD 2,|QB10|
                TRA PUSHO+6
NOP:            POP 1,3 ;HAVE TO CHECK PREDECESSOR
                PUSH 1,3
                CAIE 2,6
                TRA .+5
                CAME 3,|QCFLIST+1|        ;B6 MUST FOLLOW A1
                TRA CENTRA
                ADD 2,|QCFLIST|
                TRA PUSHO+6
                CAIE 2,9.
                TRA .+4
                CAME 3,|QCFLIST+2|        ;B9 MUST FOLLOW A2
                TRA CENTRA
                TRA NOP+6
                CAIE 2,7           ;B7 MUST FOLLOW B6 OR B10B
                TRA .+6
                CAMN 3,|QCFLIST+6|
                TRA NOP+6
                CAME 3,|QB10+2|
                TRA CENTRA
                TRA NOP+6
                CAMN 3,|QB12|      ;B8 MUST FOLLOW B12A OR ANY B10
                TRA NOP+6
                HRRZS 3
                CAIGE 3,B10
```

```
        TRA CENTRA
        CAIL 3,B11
        TRA CENTRA
        TRA NOP+6
CLOSE:  MOVE 3,FINSAV
        JUMPN 3,PUSHC     ;PUSH CLOSE IF PREDETERMINED
        RAN 3
        ANDI 3,3
        ADDI 3,14.        ;NUMBER FROM 14 TO 17
        TRA PUSHC
PLAYIT: SETZM 1
        SKIPN STACK(1)    ;ZERO MARKS END OF STACK
        FINE
        PUSHJ$ @STACK(1)
        R
        AOJA 1,PLAYIT+1
CFLIST: 0
        A1
        A2
        A3
        A4
        A5
        B6
        B7
        B8
        B9
        3,,B10
        2,,B11
        2,,B12
        B13
        C14
        C15
        C16
        C17
B10:    B10A
        B10B
        B10C
B11:    B11A
        B11B
B12:    B12A
        B12B

STACK:  BLOCK 20

END TUNE

;
```

EXAMPLE 2.3.19

Here are examples of three "antiphons" which this program produced:



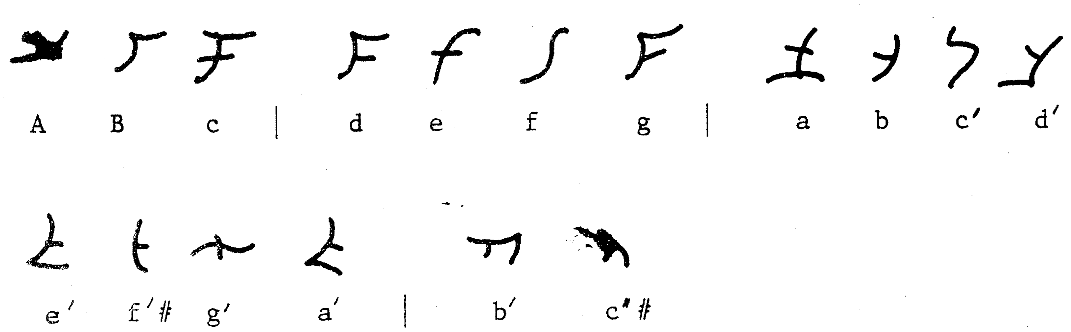
Example 2.3.20

Chapter 3 Early Polyphony

3.1. Unmeasured Polyphony

The earliest extant manifestations of polyphonic music are the organa which date from the ninth century ([Apel, 1969]). The most detailed description of this music is to be found in the Musics and Scholia Enchiriadis treatises ([Spiess]). These documents do not make use of the square neumes with which the examples in the previous section are notated. The system employed is described by Apel in The Notation of Polyphonic Music: 900-1600:

> Here a staff of a varying number of lines (four to eighteen) is used, the interspaces of which represent the successive degrees of the scale. Instead of using notes or similar signs, the syllables of the text are placed in the proper interspaces . . . . The pitch is further clarified by means of the so-called Dasia notation, written at the left of the staff. This system, which is a mediaeval imitation of the ancient Greek notation, utilizes four basic signs for the tones of the tetrachord d e f g, and others (derived largely from these by changing their position from upright to horizontal, or their direction from right to left, as in Greek notation) for one lower and two-and-half higher tetrachords which repeat the basic tetrachord in exact transpositions of the fifth. There results a curious scale which avoids diminished fifths but, as a consequence, includes augmented octaves as follows:



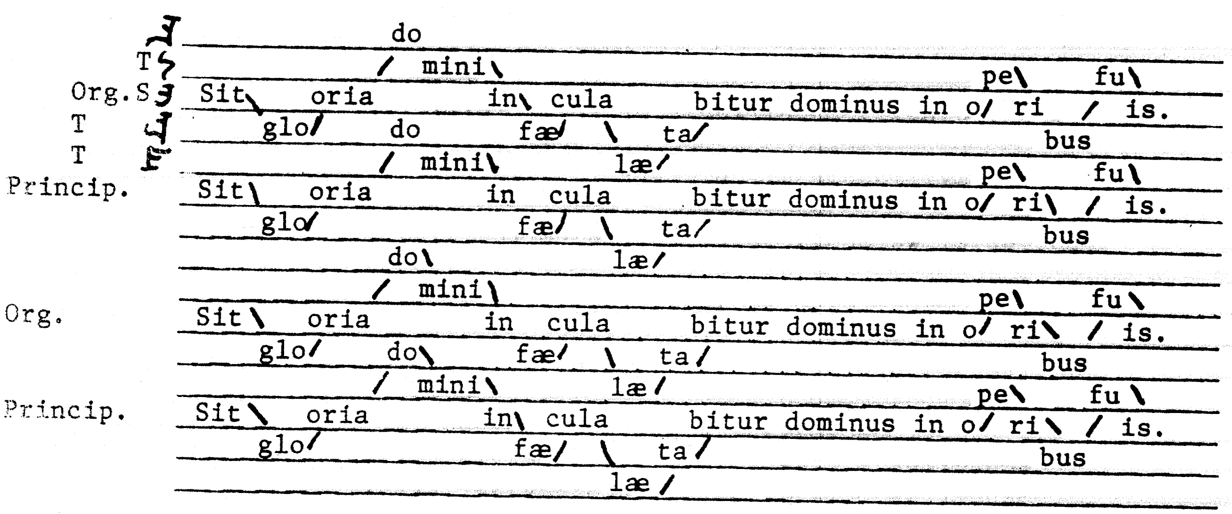
Example 3.1.1

The letters t (or t°) and s, indicating tonus and semitonus (whole-tone and semitone) are added in some of the examples as a further clarification of pitch. ([Apel, 1953]).

The practice of octave doubling, the most elementary form of polyphony, was not meant to include the augmented octaves of this particular gamut. L. B. [Speiss] cites the following passage from the Scholia Enchiriadis as evidence:

> But it must be realized that in this greatest symphonia (i.e. the symphonia of the octave) the voice which shall be added to the upper and (or?) lower voice at the octave does not follow the order of its own position but of that to which it shall respond consonantly.

This is further reinforced by the following example from the Musics Enchiriadis in which the intervallic specifications at the left of the staff alter the B-natural to a B-flat in the upper voice octave-doubling of the vox organalis ([Gerbeto]):



Example 3.1.2

Were we to program this example, the only information we would need to provide by way of a template would be the basic gamut. This may be represented in the same manner as Odo's gamut, i.e. in a linear array:

```
MEGAM:  I G
        I A
        I B FL
        J C
        J D
        J E
        J F
        J G
        J A
        J B
        K C
        K D
        K E
        K F SH
        K G
        K A
        K B
        L C SH
```

;

EXAMPLE 3.1.3

Like the Scholia Enchiriadis we shall refer to pitches of the gamut numerically ([Speiss]); however, while the Enchiriadis begins its count at e, we shall refer to the bottom of the gamut as zero and reference MEGAM with an index register, TONOS.

[Reese] provides the following analysis of Example 3.1.2:

> Let us imagine a group of voices (or instruments, or both) rendering a Gregorian melody, not in the beautiful flowing manner we believe within the correct tradition of plainsong style, but slowly and deliberately, as though the executants wished to make sure that another group, not performing in unison with them, would be prevented from going astray. (The author of the Musica Enchiriadis specifically recommends

> a slow tempo as suitable for organum.) Let us imagine the second group duplicating the church melody a fifth below throughout. The result would be the strict type of simple organum at the fifth or _diapente_. The group performing the plainsong would have the _vox principalis_; the other group, the _vox organalis_. If, in addition, the _vox principalis_ were doubled at the octave below and the _vox organalis_ at the octave above, we would have strict composite organum at the fifth.

In accordance with the principles of the _Scholia Enchiriadis_, EUTERPE executes the octave transpositions independently of the gamut, using the RELPIT instruction. However, the _vox organalis_ is obtained by referencing the address of the _vox principalis_ minus four, i.e. moving to the corresponding location on the next lower tetrachord. We define a special purpose macro, VOX, to set TONOS to the proper value and then sound the corresponding note. The execution of the processor is such that TONOS is set before the other voice programs are processed, so that they can work on the basis of its "current" value. The following program obtains.

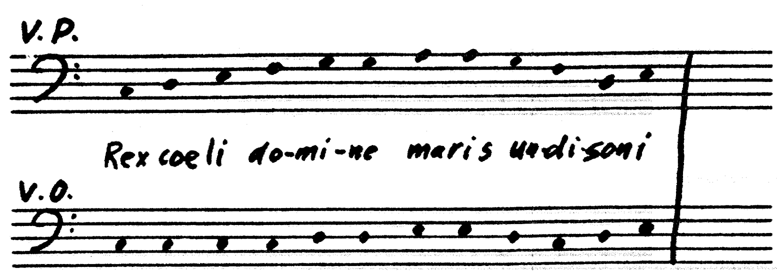
```
TITLE SIT GLORIA DOMINI

TONOS=1

DEFINE VOX X
        MOVEI TONOS,X
        XCT MEGAM(TONOS)
TERMIN

.INSRT EUTERP >
.INSRT GAMUT >  ;LOAD MEGAM

LOC VOICE1      ;VOX PRINCIPALIS

        R 4T
        VOX 6
        VOX 5
        REPEAT 2 VOX 6
        VOX 8.
        REPEAT 2 VOX 7
        VOX 6
        VOX 5
        REPEAT 2 VOX 6
        VOX 4
        VOX 5
        REPEAT 7 VOX 6
        VOX 7
        VOX 6
        VOX 5
        VOX 7
        VOX 6
        FINE

LOC VOICE2      ;VOX PRINCIPALIS DOUBLED AT THE OCTAVE BELOW

        RELPIT -1
        TRA VOICE1

LOC VOICE3      ;VOX ORGANALIS

        R 4T
        XCT MEGAM-4(TONOS)
        TRA .-1 ;EACH TIME XCT IS REPEATED TONOS WILL HAVE NEXT VALUE

LOC VOICE4      ;VOX ORGANALIS DOUBLED AT THE OCTAVE ABOVE

        RELPIT 1
        TRA VOICE3

END TUNE
;
```

EXAMPLE 3.1.4

While the gamut has been cleverly designed to avoid diminished fifths, augmented fourths arise consistently between the second note of a tetrachord and the third note of the next lower tetrachord. Consequently, the Musica Enchiriadis prescribes a set of rules for the use of oblique motion to avoid the tritone in organum at the fourth. These rules are somewhat flexible and tend to vary with different examples. However, one basic principle is summarized by [Reese] as follows:

> When the vox principalis begins in such a way that the vox organalis cannot accompany at the fourth without passing below the fourth degree of the lower tetrachord, the vox organalis has to begin in unison with the vox principalis and, unless the interval of a fourth is immediately thereafter reached, remain stationary until it is possible to parallel the vox principalis at that interval. Similarly, the vox organalis has to close in unison with the vox principalis if the ending does not admit of a duplication at the fourth.

This may be illustrated by the following example from the Musica Enchiriadis which we have transcribed into conventional notation ([Geberto]):



Example 3.1.5

The *vox organalis* begins in unison with the *vox principalis*, and does not move until the latter has risen a fourth above it. The two voices then proceed in parallel motion until they join in unison on the last two notes.

In representing this as a program, the chant melody is performed by voices (the *vox principalis*) and is stored as the subroutine REX. As TONOS is set by voice 1, voice 2 examines the index to determine whether or not organum at the fourth is possible (CAILE TONOS, 5). If the index is indeed too low (i.e. if the contents of TONOS is less than or equal to 5, then it sounds the opening pitch (indexed by accumulator 2) at the beginning of a phrase and doubles VOICE 1 at the end of the phrase.

```
TITLE FREE ORGANUM

TONOS=1

DEFINE VOX X
        MOVEI TONOS,X
        XCT MEGAM(TONOS)
TERMIN

.INSRT EUTERP >
.INSRT GAMUT >

LOC VOICE1        ;VOX PRINCIPALIS

        R 4T
        PUSHJ$ REX
        FINE

REX:    VOX 3
        VOX 4
        VOX 5
        VOX 6
        REPEAT 2 VOX 7
        REPEAT 2 VOX 8.
        VOX 7
        VOX 6
        VOX 4
        VOX 5
        POPJ$

LOC VOICE2        ;VOX ORGANALIS

        R 4T
        SETZM 5
        MOVE 2,TONOS      ;SAVE OPENING PITCH
FIND:   CAILE TONOS,5     ;NO PARALLEL MOTION UNTIL TONOS HIGH ENOUGH
        TRA .+4 ;IF TONOS GREATER THAN 5 SOUND ORGANUM AT FOURTH
SING:   JUMPN 5,SAVE      ;UNISON AT END
        XCT MEGAM(2)      ;HOLD PITCH AT BEGINNING
        TRA FIND
        XCT MEGAM-3(TONOS)        ;CONSTRUCT ORGANUM AT FOURTH
        SETOM 5
        TRA FIND
SAVE:   XCT MEGAM(TONOS)          ;UNISON WITH VOICE1
        TRA FIND

END TUNE

;
```

EXAMPLE 3.1.6

3.2. Rules of Mensuration

The earliest application of rhythm to melody involved the adaptation of a rhythmic mode to a melodic line. This consisted in a rhythmic pattern which would be strictly applied to a given melodic passage, although it was possible to change modes within a voice or to execute different modes in different voices. Towards the end of the 12th century, these modes were determined by a ligature notation which provided a set of subtle "cues" or "hints," rather than an explicit definition.

Pitch and rhythm are, thus, seen to be independent processes which, when combined, yield a rhythmic melodic progression. The MELD instruction in EUTERPE is particularly appropriate to such circumstances. By omitting all rhythmic specifications, a modal routine which simply iterates a rhythmic pattern may be melded onto any melodic pattern. The modes may then be stored in a template and accessed by any appropriate melodic program.

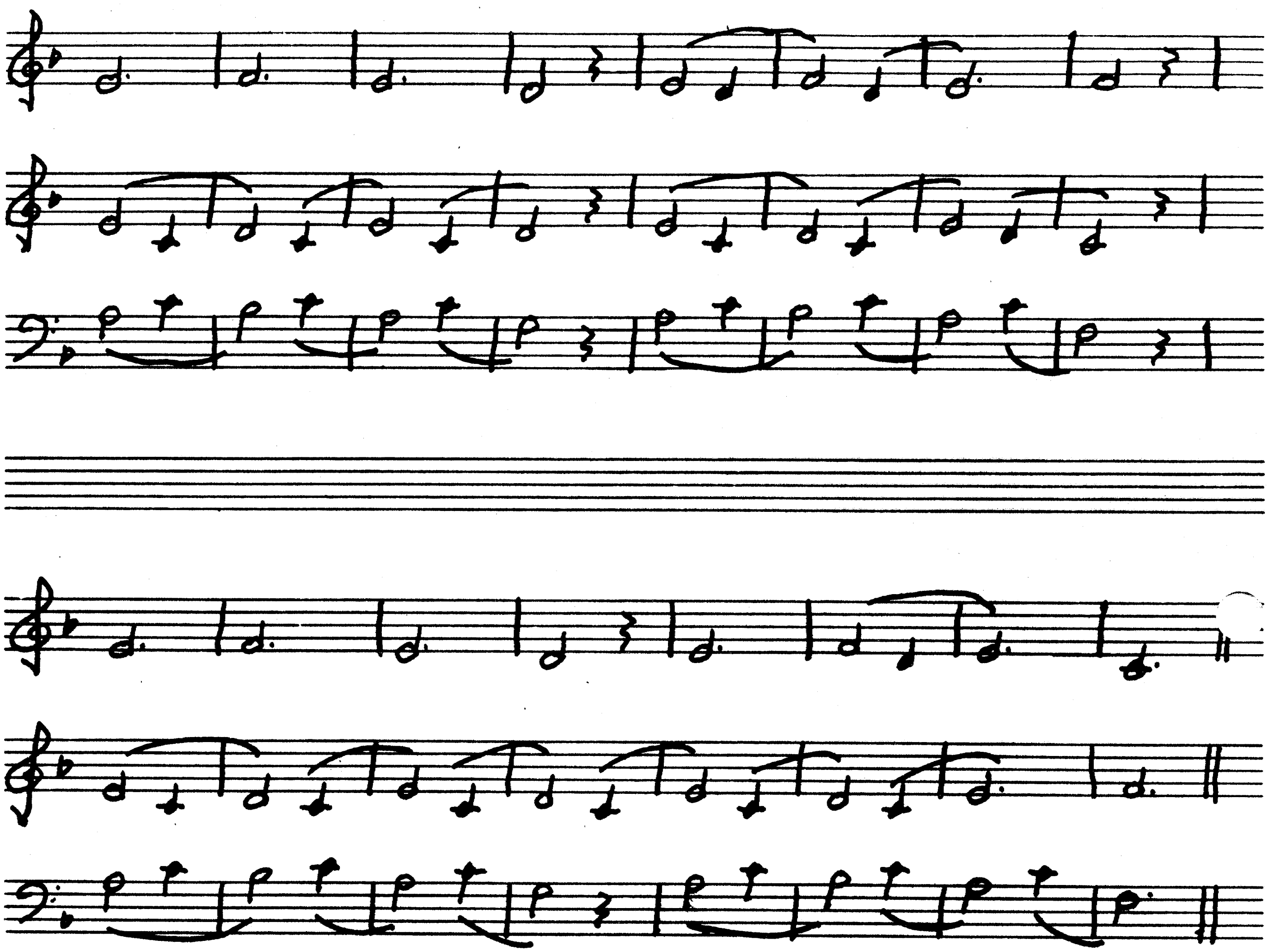
We use the customary representation of six rhythmic modes as may be found in the Harvard Dictionary of Music. The first mode is the repeated pattern of a half note followed by a quarter note; the second mode is the reverse, i.e. the pattern is a quarter note followed by a half. In the third mode the pattern consists of a dotted half note followed by a quarter note and a half note; in the fourth it is a quarter note followed by a half note and a dotted half note.

The fifth mode consists of repeated dotted half notes; and the sixth mode is formed from repeated quarter notes, generally grouped in threes ([Apel, 1969]). Here is the template for the rhythmic modes;

```
MODE1:  R 2T
        R 4T
        TRA MODE1

MODE2:  TRA MODE1+1

MODE3:  R 2D
        R 4T
        R 2T
        TRA MODE3

MODE4:  TRA MODE3+1

MODE5:  R 2D
        TRA MODE5

MODE6:  R 4T
        TRA MODE6

;
```

EXAMPLE 3.2.1

The following is a transcription by Johannes Wolf of modal polyphony ([Wolf, 1963]):



Example 3.2.2

While more recent research has shown Wolf's treatise of 1919 to be unreliable on many counts, we have chosen this example because it is such a straightforward demonstration of modal rhythms. The reality is seldom as simple as this example, and we use it only as an ideal case.

Our program representation involves melodic specification with melding into the proper rhythmic modes. The melody lines are stored by voice and by phrase. There is considerable interrelationship of thematic material, so that the only straightforward routines are the first phrases of each voice, while most of the remaining material is derivative. One particularly interesting example is the use of an index register in VIP2 to augment the theme in VIP1. The program follows:

```
TITLE VIDERUNT OMNES

.INSRT EUTERP >
.INSRT MODES >   ;LOAD TEMPLATE FOR RHYTHMIC MODES

IRP X,,|1,2,3|   ;SPECIAL MACRO TO HAVE THE THREE VOICES CALL THEIR
        LOC VOICE!X      ;RESPECTIVE PHRASES
                IRP Y,,|1,2,3,4|
                        PUSHJ$ V!X!P!Y
                TERMIN
                FINE
TERMIN

V1P1:   MELD MODE5       ;FIRST PHRASE FOR VOICE1
        K E
        K F
        K E
        UNMELD
        MELD MODE1
        TRA V2P1+7

V1P2:   MELD MODE1       ;SECOND PHRASE FOOR VOICE1
        MOVE 1,|-2,,1|
        XCT V1P1(1)      ;ELABORATION OF V1P1
        K D
        AOBJN 1,.-2
        UNMELD
        MELD MODE5
        EXIT1 V1P1+6
        PUSHJ$ V1P1+3
V1P2A:  K F
        TRA V2P1+8.

V1P3:   TRA V1P1         ;THIRD PHRASE FOR VOICE1 (SAME AS FIRST)

V1P4:   EXIT1 V1P1+2     ;FOURTH PHRASE FOR VOICE1
        PUSHJ$ V1P1
        UNMELD
        MELD MODE1
        MOVE 1,|-1,,2|
        EXIT1 V1P2A
        PUSHJ$ V1P2+2
        UNMELD
        MELD ALL,MODE5
        K C
        POPJ$

V2P1:   MELD MODE1       ;FIRST PHRASE FOR VOICE2
        K E
        K C
```

```
        K D
        K C
        K E
        K C
        K D
        0
        UNMELD
        POPJ$

V2P2:   EXIT1 V2P1+6      ;SECOND PHRASE FOR VOICE2
        PUSHJ$ V2P1       ;SAME AS V2P1 BUT WITH DIFFERENT ENDING
        K D
        K C
        TRA V2P1+8.

V2P3:   EXIT1 V2P1+8.     ;THIRD PHRASE FOR VOICE2
        PUSHJ$ V2P1       ;SAME AS V2P1 BUT WITH DIFFERENT ENDING
        K C
        UNMELD
        POPJ$

V2P4:   EXIT1 V2P1+5      ;FOURTH PHRASE FOR VOICE2
        PUSHJ$ V2P1       ;SAME AS V2P1 BUT WITH DIFFERENT ENDING
        UNMELD
        TRA V1P1

V3P1:   MELD MODE1        ;FIRST PHRASE FOR VOICE3
        J A
        K C
        J B FL
        K C
        J A
        K C
        J G
        0
        UNMELD
        POPJ$

V3P2:   EXIT1 V3P1+7      ;SECOND PHRASE FOR VOICE3
        PUSHJ$ V3P1       ;SAME AS V3P1 BUT WITH DIFFERENT ENDING
        J F
        TRA V3P1+8.

V3P3:   TRA V3P1          ;THIRD PHRASE FOR VOICE3 (SAME AS FIRST)

V3P4:   TRA V3P2          ;FOURTH PHRASE FOR VOICE3 (SAME AS SECOND)

END TUNE
;
```

EXAMPLE 3.2.3

While ligatures could represent rhythmic patterns associated with multiple-note neumes, the major contribution to the development of a mensural system was the assignment of rhythmic values to single notes. The anonymous treatise *Regles Sur L'Art de Dechanter* ([Coussemaker]) distinquishes between simple (*simplices*) and composite (*compositae*) notes. Three types of simple notes are specified: long (▉), breve (■), and semibreve (◆).

The interpretation of these symbols was ultimately codified by Franco of Cologne in the treatise *Ars Cantus Mensurabilis*. Each symbol admits of two possible interpretations. The breve usually assumes the duration of a single beat, or *tempus*; in this case, it is called a *recta brevis*. However, it may also endure two beats, in which case it is called an *altera brevis*. A long may occupy either three beats (*perfecta longa*) or two beats (*imperfecta longa*); and a semibreve is either minor, in which case it assumes one-third of a beat, or major, which represents two-thirds of a beat ([Franconis]).

The following motet, from manuscript H196 of the Faculte de Medecine de Montpellier, provides an example of Franconian notation ([Rokseth]):


Triplum
On parole de batre et de vanner Et de foïr et de hanner Mais ces deduis trop me desplaisent,
Car il n'est si bonne vie Que d'estre a aise De bon cler vin et de chapons Et d'estre avec bons
compaignons, Liés et joi- ans, Chantans, truffans et amorous, Et d'avoir quant c'on a mestier,
Pour so la cier, Be les dames a de vis: Et tout ce truev'on a Paris.
Duplum
A Paris soir et matin Truev'on bon pain et bon cler vin, Bonne char et bon poisson,
De toutes guises compaignons, Sens sou tie grant baudoir, Biaus joiaus dames d'onnour
Et si treu'on bien entre deus De menre feur pour homes de si teus.
Tenor
Frese nouvele! Muere france, muere muere france!

Example 3.2.4

While there are a few instances of composite notes (both ligatures and plicata) in this example, almost all the notes are simple (including the tenor -- a relatively rare case). It is easy enough to define special symbols for the rhythmic field of a note word which represent longs, breves, and semibreves; but something has to account for their alternative interpretations.

The solution is provided by a special-purpose subroutine which realizes Franco's rules. Certain subtleties have had to be programmed into this routine so that it can be accessed by all six voices (a bit of overkill since more than three voices are rarely needed). In addition, we have employed a useful heuristic which seems to have escaped documentation from Franco to the present. The key lies in the importance of the perfection, the unit of three beats. While the perfection does not quite introduce bar lines (there being no stress associated with its first beat), there is no tendency towards sustaining a note across one perfection into the next (just as semibreves do not permit syncopation of a note crossing from the middle of one beat into the next.)

This makes matters much simpler than they are in Apel's description. A routine to interpret Franconian notation needs only three basic states, corresponding to the three beats of the perfection. (As we have seen, semibreves form groups, each of which occupy a single beat; so we need not worry about subdividing the beat.) On the basis of its current state and its current input (i.e. note-shape), the routine can easily determine the associated duration according to the following algorithm:

```
On the first beat:
     If the note is a LONG:
          It is perfect when followed by
               a long or exactly two or three
               breves
          It is imperfect otherwise
     If the note is a BREVE:
          It is always recta
On the second beat:
     If the note is a LONG:
          It is always imperfect
     If the note is a BREVE:
          It is altered if it is followed
               by a long or a signum perfectionis
          It is recta otherwise
On the third beat:
     A LONG is impossible
     A BREVE is always recta
```

When counting breves after a long, the count is halted by a _signum perfectionis_ and pairs or triples of semibreves count as unit breves. Semibreves are handled on any beat by the following subroutine:

```
The first semibreve is always minor
The second semibreve is:
     minor if it is followed by exactly one
          semibreve
     major, otherwise
The third semibreve (if it exists) is always
     minor
```

This accounts for all possibilities within a beat.

The significance of this heuristic is that while each note sign has two possible interpretations, an ambiquity is possible only _once_ for each figure: on the first beat of the perfection for longs, on the second beat for breves, and on the second subdivision of a beat for semibreves. All other cases are strictly determined. Look-aheads are also considerably reduced. The breve decision only involves the successor, and the semibreve decision must check the next two notes; only the long decision may be forced to look four notes ahead.

In order that it may be processed by more than one voice, this subroutine keeps a set of registers for each voice which serve as a "program counter" through the program being processed, a pointer to the next beat in the perfection, and a pointer to the next semi-breve sub-beat within a beat. Finally, the note is stored in a register and played by an XCT instruction, so that the original program is unaltered.

A program using mensural symbols is properly interpreted using the macro MENS. MENS takes as argument the first address to be so interpreted, sets the appropriate program counter, and calls the mensural interpreter for the voice which executes the macro. The program is then processed until a non-note-word (conventionally POPJ$) is encountered, at which the interpreter is exited. Control returns to the address after the MENS macro.

It would not be difficult to write this interpreter into EUTERPE's processor, in which case MENS would become an instruction rather than a macro. However, Franconian notation is very much a special case; even the Montpelier codex has relatively few pieces so -otated. Such a feature would be an esoteric frill of little practical value to most users.

We present a listing of the interpreter with abundant comments relevant to the above discussion:

```
TITLE FRANCONIAN NOTATION

VN=17

LONG=2D         ;LONG NORMALLY DOTTED HALF
BREVE=4T        ;BREVE NORMALLY QUARTER
SEMIB=8T3       ;SEMIBREVE NOREMALLY TRIPLET EIGHTH

DIVISI=777000

DEFINE MENS ADR
        MOVEI 17,ADR
        LOAD 12,VP
        MOVEM 17,@12
        PUSHJ$ MENSUR
TERMIN

MENSUR: MOVE VN,VOICE    ;INITIALIZE UPON ENTRY
        SETZM SBPT1-1(VN)       ;SEMIBREVE COUNT AT BEINNING OF BEAT
        MOVEI 15,BEAT1   ;PERFECTION AT FIRST BEAT
        MOVEM 15,NXBT1-1(VN)
        TRA LD           ;LOAD FIRST NOTE
BEAT1:  CAIN 16,21       ;LONG?
        TRA .+6          ;JUMP IF SO
        MOVEI 15,BEAT2   ;IF NOT NEXT BEAT IS SECOND OF PERFECTION
TORY:   MOVEM 15,NXBT1-1(VN)
        CAIE 16,11       ;SEMIB?
        TRA PLAY         ;IF NOT PLAY BREVE AS QUARTER NOTE
        TRA SB1          ;IF SO BEGIN SEMIBREVE ROUTINE
        PUSHJ$ BREVEP    ;IS LONG FOLLOWED BY EXACTLY TWO OR THREE
        JUMPE 15,PERF           ;SEMIBREVES?
        TRZ 15,1
        JUMPN 15,PERF    ;JUMP IF SO
        MOVEI 15,BEAT3   ;IF NOT NEXT BEAT IS THIRD OF PERFECTION
        MOVEM 15,NXBT1-1(VN)
        MOVEI 16,5       ;HALF NOTE LONG IS PLAYED
        DPB 16,BYTPTR
        TRA PLAY
PERF:   MOVEI 15,BEAT1   ;NEXT BEAT IS FIRST OF PERFECTION
        MOVEM 15,NXBT1-1(VN)
        TRA PLAY         ;PLAY PERFECT LONG
BEAT2:  CAIE 16,21       ;LONG?
        TRA .+4          ;JUMP IF NOT
        MOVEI 15,BEAT1   ;NEXT BEAT IF THIRD OF PERFECTION
        MOVEM 15,NXBT1-1(VN)
        TRA PERF-3
        MOVEI 15,BEAT3
        MOVEM 15,NXBT1-1(VN)
        CAIN 16,11       ;SEMIB?
        TRA SB1          ;IF SO BEGIN SEMIBREVE ROUTINE
```

```
        PUSHJ$ LONGP      ;FOLLOWED BY LONG OR DIVISI?
        JUMPN 15,BEAT2+2          ;IF SO ALTER BREVE
        TRA PLAY          ;IF NOT PLAY IT RECTA
BEAT3:  MOVEI 15,BEAT1    ;NEXT BEAT IS FIRST OF PERFECTION
        TRA TORY          ;JUST LIKE FIRST BEAT BUT NO LONGS

SB1:    MOVEI 15,SB2      ;NEXT SEMIBREVE IS SECOND
        MOVEM 15,SBPT1-1(VN)
        TRA PLAY          ;PLAY SEMIBREVE AS TRIPLET EIGHTH
SB2:    PUSHJ$ SBP        ;IS THIS SEMIB FOLLOWED BY EXACTLY ONE MORE?
        JUMPE 13,SB3-2    ;IF NOT PLAY MAJOR SEMIBREVE
        MOVEI 15,SB3      ;IF NOT NEXT SEMIBREVE IS THIRD
        TRA SB1+1         ;PLAY MINOR SEMIBREVE
        MOVEI 16,12       ;MAJOR SEMIBREVE
        DPB 16,BYTPTR
SB3:    SETZM SBPT1-1(VN)         ;THIRD SEMIBREVE IS END OF BEAT
PLAY:   XCT NOTE1-1(VN) ;PLAY NOTE
        MOVE VN,VOICE
        AOS VP1-1(VN)     ;INCREMENT PROGRAM COUNTER
        LDB 16,INSPTR     ;NOTE WORD?
        SKIPE 16
        POPJ$             ;EXIT IF NOT
LD:     MOVE 16,@VP1-1(VN)        ;GET NOTE
        MOVEM 16,NOTE1-1(VN)
        SKIPE SBPT1-1(VN)         ;MIDDLE OF BEAT?
        TRA @SBPT1-1(VN)          ;IF SO GO TO SEMIBREVE ROUTINE
        LDB 16,BYTPTR     ;GET RHYTHM FIELD
        CAIN 16,777       ;DIVISI?
        TRA PLAY+2        ;GET NEXT NOTE IF SO
        TRA @NXBT1-1(VN)          ;GO TO APPROPRIATE BEAT IN
                                  ;PERFECTION IF NOT
SBP:    SETZM 13          ;ZERO IF NO SEMIBREVES
        AOS VP1-1(VN)     ;LOOK AT NEXT WORD
        LDB 14,BYTPTV
        CAIE 14,11        ;SEMIB?
        TRA SBP2          ;POP BACK IF NOT
        JUMPN 13,SBP1     ;JUMP IF SECOND SEMIBREVE
        AOS 13            ;SET TO ONE
        PUSHJ$ SBP+1      ;LOOK AT NEXT WORD
SBP2:   SOS VP1-1(VN)
        POPJ$
SBP1:   SETZM 13          ;SET TO ZERO
        TRA SBP2          ;POP BACK

BREVEP: SETZM 15          ;ZERO IF NO BREVES
        AOS VP1-1(VN)     ;LOOK AT NEXT WORD
        LDB 14,BYTPTV
        CAIE 14,4         ;BREVE?
        TRA CSB ;CHECK FOR SEMIBREVES IF NOT
        AOS 15  ;INCREMENT BREVE COUNT BY ONE
```

```
        CAILE 15,3        ;MORE THAN THREE BREVES?
        TRA BP  ;JUMP IF SO
        PUSHJ$ BREVEP+1 ;LOOK AT NEXT WORD IF NOT
        SOS VP1-1(VN)
        POPJ$
CSB:    CAIE 14,11        ;SEMIB?
        TRA .-3 ;POP BACK IF NOT
        AOS VP1-1(VN)    ;IF SO LOOK AT NEXT WORD
        LDB 14,BYTPTV
        CAIE 14,11        ;SEMIB?
        TRA SCB ;IF NOT CHECK FOR BREVE
        AOS 15  ;INCREMENT BREVE COUNT BY ONE
        CAILE 15,3        ;MORE THAN THREE BREVES?
        TRA BP1 ;JUMP IF SO
        PUSHJ$ BREVEP+1 ;LOOK AT NEXT WORD IF NOT
BP2:    REPEAT 2 SOS VP1-1(VN)
        POPJ$
SCB:    PUSHJ$ BREVEP+3
        SOS VP1-1(VN)
        POPJ$
BP:     MOVEI 15,1        ;SET TO ONE
        TRA CSB-2         ;POP BACK
BP1:    MOVEI 15,1        ;SET TO ONE
        TRA BP2           ;POP BACK

LONGP:  SETZM 15          ;SET TO ZERO
        AOS VP1-1(VN)     ;LOOK AT NEXT WORD
        LDB 14,BYTPTV
        CAIN 14,21
        SKIPA
        CAIN 14,777
        SETOM 15          ;SET TO ONES IF LONG OR DIVISI
        SOS VP1-1(VN)
        POPJ$

IRP Y,,|VP,NXBT,SBPT,NOTE|
        IRP X,,|1,2,3,4,5,6|
                Y!!X:   0
        TERMIN
TERMIN
BYTPTR: (111100)NOTE1-1(VN)
INSPTR: (331100)@VP1-1(VN)
BYTPTV: (111100)@VP1-1(VN)

;                         EXAMPLE 3.2.5
```

Using this mensural interpreter, we may code the motet presented above as follows:

```
TITLE ON PAROLE A PARIS FRESE NOUVELLE

.INSRT EUTERP >
.INSRT FRANCO >

LOC VOICE2        ;ON PAROLE

        MENS V1
        FINE

V1:     K F BREVE
        BREVE
        K C BREVE
        K D BREVE
        K E BREVE
        K F SEMIB
        K E SEMIB
        K D BREVE
        K E SEMIB
        K F SEMIB
        K G BREVE
        K A BREVE
        BREVE
        K G SEMIB
        K F SEMIB
        DIVISI
        K E SEMIB
        K F SEMIB
        K G LONG
        DIVISI  ;INSERTED TO JUSTIFY REST
        R BREVE
        R BREVE
        DIVISI
        K A BREVE
        BREVE
        K G SEMIB
        K F SEMIB
        K G BREVE
        K D BREVE
        K E BREVE
        K F LONG
        DIVISI
        K D BREVE
        K E BREVE
        K F BREVE
        K G BREVE
        K F SEMIB
        K E SEMIB
        K D BREVE
        K E BREVE
```

```
BREVE
K G SEMIB
K F SEMIB
DIVISI
K E SEMIB
K D SEMIB
K F LONG
DIVISI  ;INSERTED TO JUSTIFY REST
R BREVE
R BREVE
DIVISI
K A BREVE
SEMIB
K G SEMIB
DIVISI
K F SEMIB
K E SEMIB
K D BREVE
K G LONG
K F BREVE
K E SEMIB
K D SEMIB
K C BREVE
K D BREVE
K E BREVE
K F BREVE
K G BREVE
BREVE
K F BREVE
K E BREVE
SEMIB
K D SEMIB
K C SEMIB
K D BREVE
K E SEMIB
K F SEMIB
K G LONG
BREVE
K F BREVE
K E SEMIB
K D SEMIB
K E LONG
R LONG
DIVISI  ;INSERTED TO JUSTIFY REST
K F BREVE
LONG
K G BREVE
K A BREVE
BREVE
K G BREVE
```

```
K F SEMIB
K E SEMIB
K D BREVE
K C BREVE
K D LONG
K E SEMIB
K D SEMIB
K E BREVE
K F SEMIB
K G SEMIB
DIVISI
K F SEMIB
K E SEMIB
K D BREVE
K E SEMIB
K F SEMIB
DIVISI
K G SEMIB
K A SEMIB
BREVE
K G SEMIB
K F SEMIB
K G LONG
POPJ$
```

```
LOC VOICE1       ;A PARIS

        TEMPO ALL,(4)90.
        MENS V2P1
        J A 2T
        J B 8T   ;COMPOUND NEUME
        K C
        MENS V2P2
        FINE

V2P1:   K C LONG
        J B BREVE
        J A BREVE
        K D LONG
        DIVISI
        K E BREVE
        K D LONG
        K C SEMIB
        J B SEMIB
        K C LONG
        J A BREVE
        J B SEMIB
        K C SEMIB
        K D BREVE
        DIVISI  ;COMPOUND LONG
```

```
K E BREVE(SLUR)
K D BREVE
K C BREVE(LEGATO)
DIVISI
K D LONG
R BREVE
K C LONG
DIVISI
K D BREVE
K C BREVE
K D LONG
DIVISI
K C BREVE
J B LONG
J A SEMIB
J G SEMIB
J F LONG
J A BREVE
J B SEMIB
K C SEMIB
DIVISI
J A SEMIB
J G SEMIB
J A LONG
DIVISI
K D BREVE
BREVE
K F LONG
DIVISI
K D BREVE
K E BREVE
K D LONG
DIVISI
K E BREVE
K F LONG
K E SEMIB
K D SEMIB
K C LONG
BREVE
K D BREVE
K E LONG
R BREVE
K D BREVE(SLUR) ;PLICA
K C BREVE(LEGATO)
J B BREVE
K C LONG
K D BREVE
K C BREVE
K D LONG
DIVISI
```

```
        K C BREVE
        J B LONG
        J A SEMIB
        J G SEMIB
        J A LONG
        K C BREVE
        K D BREVE
        K C SEMIB
        J B SEMIB
        POPJ$

V2P2:   K D LONG
        POPJ$

LOC VOICE3        ;FRESE NOUVELLE

        MENS V3
        TRA VOICE3

V3:     J F LONG
        J G BREVE
        J A BREVE
        J G LONG
        DIVISI
        J A BREVE
        R LONG
        DIVISI
        J A SEMIB
        J B SEMIB
        K C LONG
        BREVE
        J B SEMIB
        J A SEMIB
        SEMIB
        J B SEMIB
        J A LONG
        J G LONG
        R BREVE
        POPJ$

END TUNE
```

; EXAMPLE 3.2.6

Special arrangements have been made for ligatures and <u>plicata</u>; but otherwise, the transcription is straightforward. Here is a transcription of the motet into modern notation:


On paro- le de batre et de vanner Et de foir et de hanner
A Paris soir et ma- tin Truev'on
Fre- se nou- ve- le! Muere fran-
Mais ces deduis trop me des- plai- sent, Car il n'est si bonne
bon pain et bon cler vin, Bon- ne char et
ce, muere muere fran- ce! Fre- se nou- ve-
vi-e Que d'estre a ai- se De bon cler vin et
bon pois- son, De tou- tes guises compai-gnons,
le! Muere fran-ce, muere muere fran-


de cha - pons, Et d'estre avec bons compaignons, Liés et joi - ans,
Sens sou - ti - e, grant bau - dour, Biaus
ce! Fre - se nou - ve - le!
Chantans, truf - fans et a - mo - rous, Et d'a - voir, quant c'on
joians da - mes d'ou - nour, Et si truev'a on bien
Muere fran - ce, muere, muere Fran - ce! Fre - se nou -
a mestier, Pour so - la - cier Be les dames a devis: Et tout
en - tredeus De meure feur pour homes
ve - le! Muere fran - ce, muere, muere

Example 3.2.7

Since we have already discredited the Franconian system for its esotericism, one might wonder why we undertook this study in the first place. The answer is that it has provided us with an opportunity to implement a rhythmic system markedly different from EUTERPE's normal rhythmic notation. Other systems could be similarly implemented -- if they are deterministic, by similar means; if not, with the aid of EUTERPE's random number generator. The fact of the matter is that the user of EUTERPE is free to exercise as much or as little control over the rhythmic process as he desires.

Chapter 4

Counterpoint and Harmony

4.1. Some Theoretical Aspects of Counterpoint and Harmony

The term "counterpoint" appears to have its origins in the early 14th century ([Apel, 1969]). The development of a mensural notation which represented each note by an individual symbol led to theoretical writings on polyphony as "nota contra notam," and various contrapuntal styles arose from composers who sought to take advantage of this notation. These included the canon, the palindrome and canon cancrizans (in which the notation was read both forward and backward), and the isorhythmic structure ([Reese]). This last technique may be regarded as an extension of rhythmic modality, but the rhythmic patterns which are repeated were longer and more complex.

Given a melodic line, i.e. cantus firmus, such techniques provided various transformations which could be applied to it to yield certain musical structures. Thus, the art of counterpoint was more than simply a study of the proper placement of "note against note." It was a more general study of the construction of an entire musical composition in terms of standardized techniques and original ideas. In fact, in his Introduction to [Fux]'s study of counterpoint, Gradus Ad Parnassum, Alfred Mann cites the Renaissance theorist, Gioseffo Zarlino, as using the word contrapuntizare for the description of "the whole composition at once."

Zarlino may also be recognized as the first theoretician of harmony, since his analyses were based on chordal constructions rather than simply on combinations of consonant and dissonant intervals between individual voices ([Apel, 1969]). However, a critical point of departure between harmony and counterpoint arose in the 17th century with the distinction between Palestrina counterpoint and Bach counterpoint. (This is the terminology Apel uses in the Harvard Dictionary of Music.) The former received codification in Gradus ad Parnassum, while the latter arose out of common practice in the 17th century. Its development followed the evolution of tonality, which, as Apel writes:

> ...superseded polyphonic modality and made possible an over-all conception of a piece of music from the harmonic point of view. Only when tonality was firmly established could the relative importance of chords built on the different degrees of the scale be determined in relation to a key center or tonic; only when this was accomplished could a logical departure from this tonic into other keys and return from those keys to it -- modulation -- be achieved.
> ([Apel, 1969])

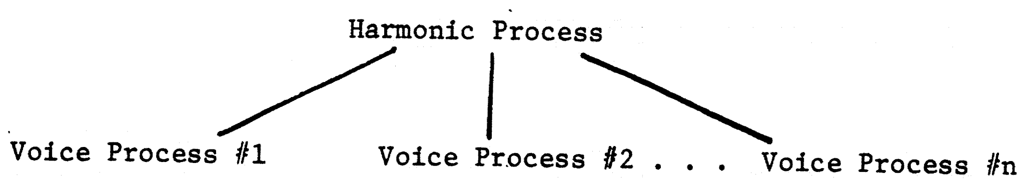
Bach counterpoint, then, was

> ...based on "tonal" (instead of "modal") harmonies, admitting six-four chords, seventh chords, and diminished seventh chords. After Bach the emphasis shifted to accompanied melody.
> ([Apel, 1969])

The distinction between Palestrina counterpoint and Bach counterpoint may also be expressed in terms of the underlying parallelism of the musical structures. Palestrina counterpoint entails a great deal of independence among the individual voices, and the underlying structure involves a parallelism consisting of as many processes as there are voices. Harmony, as such, is merely a consequence of this parallelism:

> Without question, all the composers of the sixteenth century were chord-conscious; but the music, particularly the sacred music, of a great number of them shows that the chordal aspect was secondary to the melodic in importance. Chords were not written for their own sakes; they were rather the result, the vertical aspect, of the combination of lines moving horizontally.
> ([Merritt])

In Bach counterpoint the processes of the individual voices still maintain a degree of independence similar to that of Palestrain counterpoint. However, there is now an additional process which determines the harmonic progressions. Hierarchically speaking the computation structure is organized roughly as follows:



Example 4.1.1

That is, there is a single harmonic process which dominates the overall structure of the piece within which the individual voice processes are essentially independent.

The consideration of harmony as a process unto itself is, of course, the discipline of <u>harmonic analysis</u>, which dates back to

Rameau's Traité d'harmonie (1722) ([Schenker]). Rameau's theories provoked a great deal of controversy and were viciously contested by musicians such as J. P. Kirnberger and C. P. E. Bach. However, the advent of a theory of harmony could not be reversed; and while certain aspects of Rameau's work were disputed, there were several factors which pervaded successive theories. One of these factors was the general approach to the subject in terms of what Felix [Salzer] calls chord grammar:

> Chord grammar denotes the usual type of analysis in which separate designations and labels are assigned to triads, seventh chords, etc. It is a purely descriptive means of registering and labeling each chord and relating it to different key centers. Chord grammar is the backbone of our present-day harmonic analysis, which is primarily concerned with recognition of the grammatical status of each chord in a musical work. It breaks up a phrase into a group of isolated chord entities.

The first theorist to provide an alternative to chord grammar was Heinrich Schenker. His approach is described by [Salzer] as the theory of chord significance:

> The study of chord significance, on the other hand, reveals the meaning of a chord and the specific role it plays in a phrase or section of a work, or in the work in its entirety. Chord significance, since it discloses the function of a chord, goes far beyond grammatical description by pointing out the special architectonic purpose of a chord within a phrase.
> As a first result of this distinction, Schenker found that the roles which

> chords play in a musical phrase or section are very diverse; even two grammatically identical chords appearing in the same phrase can fulfill totally different functions. Thus it follows that labelling chords according to their grammatical status never explains their functions or how they combine to create a unified whole.

Salzer's terminology of "chord grammar" and "chord significance" uncovers a fliration between music theory and linguistic theory which has been long outstanding. Recently, this flirtation has become less secretive and has been more prominent in public circles. Composers such as Luciano Berio, Arthur Berger, Elliot Carter and many others have expressed an interest in formal linguistics and particularly in mathematical linguistics. At this point, we might consider the importance of such a flirtation, as well as its chances of maturing into a full-blown affair.

A major point of view in this field is that taken by Leonard [Meyer] in his study, Emotion and Meaning in Music:

> The problem of musical meaning and communication is of particular interest for several reasons. Not only does music use no linguistic signs but, on one level at least, it operates as a closed system, that is, it employs no signs or symbols referring to the non-musical world of objects, concepts, and human desires. . . Unlike a closed, non-referential mathematical system, music is said to communicate emotional and aesthetic meanings as well as purely intellectual ones. This puzzling combination of abstractness with concrete emotional experience can, if understood correctly, perhaps yield useful insights into more general problems of meaning

> and communication, especially those involving aesthetic experience.

In generaly, the attempt to associate music with language almost immediately centers on this issue of meaning. Nothing could be further from our own approach. Taking music on a purely subjective level, we choose to regard musical "communication" as a personal relationship between the composer and the individual listener through the medium of performance. All "meaning" is purely within the imagination of the listener.

[Langer] asserts the theory that if the composer communicates anything, it is simply concepts of form. Thus, our own study has veered away from the Charybdis of "meaning" and aimed instead at an abstraction of form. In so doing, we have dismissed such matters of semantics and inference and taken an approach somewhat similar to Winograd's syntactic system:

> The parsing of a sentence indicates its detailed structure, but more important it abstracts the "features" of the linguistic components which are important for interpreting their meaning.
> ([Winograd, 1971])

Now let us again consider Salzer's dichotomy between chord grammar and chord significance. Our own approach is certainly sympathetic with the Schenkerian ideal. We attempt to regard every element of a composition in terms of its functional role in the entire composition, as a process. However, by taking a procedural approach to what Salzer calls "grammar," we are essentially avoiding the need for a dichotomy in the first place.

In the next two sections we shall try to apply our theories to examples of counterpoint and harmony. First we consider the mediaeval canon, "Sumer is Icumen In," whose form arises from some very simple procedures. Then we shall turn to one of Bach's two-part inventions which, while it is a piece of strict, two-voice counterpoint, is a powerful example of harmonic writing. We are intentionally avoiding the matter of figured bass in this study primarily because such a technique is notationally incompatible with the EUTERPE system. We intend to defer such studies until we have a better syntactic understanding of such an approach at hand.

4.2. "Sumer is Icumen In"

"Sumer is Icumen In" is one of the most familiar specimens of early counterpoint. There has been a great deal of dispute about both its origins and the manner in which it is to be performed. We shall base our program on a transcription by Manfred Bukofzer ([Greenberg]):

Example 4.2.1.

The canonic process for the upper four voices is very much the same as that of our model of "Frere Jacques," as we presented it in the Introduction. We have not stored any of the themes as subroutines in this example because it was not really necessary. Furthermore, we have programmed this, like our "Frere Jacques" model, as an "ideal canon" which never halts.

The other interesting feature of this canon is the process of <u>voice exchange</u> in the lower two parts. Voice exchange provides for a round "in the small" in which, in this particular example, two voices alternate in performing each other's parts. This gives rise to the following repeated pattern:

Example 4.2.2.

This formation is somewhat like a coroutine structure. Coroutines, as Knuth observes, exhibit a more symmetric relationship than do subroutines:

> In contrast to the unsymmetric relationship between a main routine and a subroutine, there is a complete symmetry between coroutines which call on each other. ([Knuth, 1968])

Thus, the A phrase is always followed by the B phrase which, in turn, is followed by the A phrase. Our program realization merely has one phrase in each of two voice programs and a TRA to the other voice at the end of each. Here is the program in its entirety:

```
TITLE SUMER IS ICUMEN IN

.INSRT EUTERP >

LOC VOICE1

        PITCH ALL,J C    ;CLEF SIGNS READ AN OCTAVE HIGHER THAN THEY
        L F 4T                    ;SOUND
        0
        L D
        L E

        L F
        0
        L E 8T(SLUR)
        L D
        L C 4T(LEGATO)

        K A
        0
        K B FL
        K G

        K A 2T
        R

        K F 4T
        K A
        K G
        K B FL

        K A
        0
        K G
        K F

        K A
        L C
        L D
        0

        L C 2T
        R

        L F
        L D

        L F
        R
```

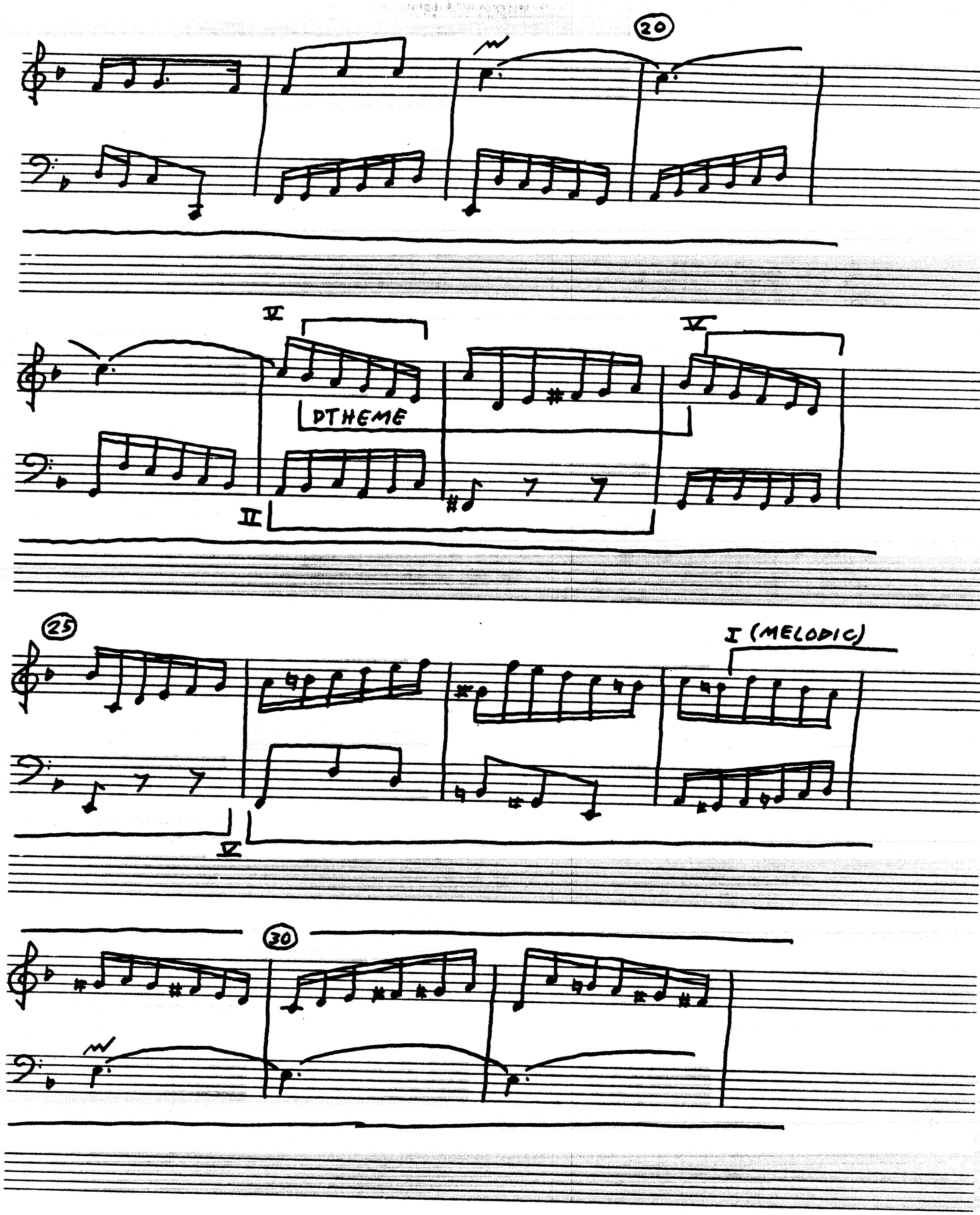
```
L C 4T
K A
K B FL
K G

K A
L C
K B FL
K A

K F
K A
K G
K E

K F 2T
R

K A 4T
0
K G
K B FL

L C
0
L D
L E

L C
0
L D
0

L C 2T
R

L F
L D

L F
L D

K F 4T
K A
K B FL
K G

K A
L C(SLUR)
K B FL(LEGATO)
```

```
        L C

        K A
        L C
        K G
        K E

        K F 2T
        R
        TRA VOICE1+1

LOC VOICE2

        REPEAT 2 R 1T     ;REST TWO MEASURES
        TRA VOICE1+1      ;START THEME

LOC VOICE3

        REPEAT 2 R 1T     ;REST TWO MEASURES
        TRA VOICE2        ;FOLLOW VOICE2

LOC VOICE4

        REPEAT 2 R 1T     ;REST TWO MEASURES
        TRA VOICE3        ;FOLLOW VOICE3

LOC VOICE5

        L C 2T            ;A PHRASE
        K B FL

        L C
        R
        TRA VOICE6        ;START B PHRASE

LOC VOICE6

        K F 2T            ;B PHRASE
        K G

        K F
        K G 4T(SLUR)
        K A(LEGATO)
        TRA VOICE5        ;START A PHRASE

END TUNE

;                         EXAMPLE 4.2.3
```

3. A Bach Invention

Canons are, by nature, rather simple in their formal structure. Let us now consider a more difficult example which entails certain aspects of contrapuntal development as well as some elementary problems in considerations of tonality. The example is the fourth of J. S. Bach's two-part inventions. We present a copy of the score with several notations regarding our EUTERPE code. In particular, we have labeled those passages corresponding to subroutines and provided a record of the scale parameter stack (using Roman numerals) for both voices ([Bach]):


THEME
SCALE : D MINOR (HARMONIC)
5
SEQ
10
III
15


20
V
D THEME
II
IV
25
I (MELODIC)
V
30


35
IV
40
I (NATURAL)
I (MELODIC)
45

Example 4.3.1.

There are two means by which one may specify notes for EUTERPE: directly and indirectly. The direct method is by simply writing out the note words, as was done in the preceding example. However, under certain circumstances, pitch and duration might be determined by separate, independent computation processes, so that EUTERPE is faced with the task of "compiling" its note words before executing them. This latter technique was particularly helpful in our representation of mediaeval music; we turn to it now for an approach to tonality.

Matters of tonality have always been a subject of violent controversy, and we shall not be so bold as to assert that the theory of modulation reduces to a study of push-down stacks. In our own treatment the scale array purports to be nothing more than an ordered subset of pitches, a subset which may be referenced with sufficient frequency to merit distinguished storage. When we change the scale parameter, we do this to allow for a change of materials, e.g. chromaticism. Whether or not this actually constitutes a modulation, as such, is a matter for further research.

These ideas bear some relationship to Schoenberg's concepts of monotonality and regions. [Schoenberg] explains monotonality as follows:

> Monotonality includes modulation -- movement towards another <u>mode</u> and even establishment of that mode. But it considers these deviations as regions of the tonality, subordinate to the central power of a tonic. Thus comprehension of the harmonic unity within

a piece is achieved.

Thus, the regions involve applications of tones foreign to the fundamental tonality but relevant to a secondary tonality which is considered as a subordinate region of the fundamental tonality. For example, when the second theme of a sonata enters "in the dominant," Schoenberg would interpret it as being derived from the materials of the <u>dominant region</u> of the principal tonality. EUTERPE's approach, however, differs from Schoenberg's on two significant points.

The first of these points derives from the use of the push-down list. Monotonality relates all regions exclusively to the tonic. While these regions are classified in terms of their distance from the tonic, there is no attempt to account for relationships among regions which do not involve the tonic. In EUTERPE a change of region is established by pushing down one level on a stack. A subsequent change of region is now expressed in terms of the current region, rather than in terms of the tonic, which is no longer at the top of the stack. This allows for a functional approach to such concepts as that of secondary dominants. If, indeed, a composition is moving to another region with respect to the tonic, this is accomplished by popping back to the tonic and pushing to that new region. Such an approach reflects Schenker's ideas concerning harmonic analysis, and more recently, examples have been provided by [Salzer], [Forte], and, in an a study of harmonic analysis by computer, [Winograd, 1968].

Secondly, we have not developed a theory of what Schoenberg calls "substitute tones" or "transformations" to account for chromaticisms. The only substitute tones which exist in EUTERPE are those provided by the harmonic and melodic minor scales. No further allowances are made for chromaticisms in the scale arrays. Such chromaticisms must be taken into account either by altering the scale array or by ignoring it in favor of direct usage of note words.

The Roman numerals and their associated brackets across the score are simply a record of the state of the scale array as affected by RELKEY instructions. Nesting of the brackets indicates the status of the push-down stack. The Roman numerals are the arguments of RELKEY; that is, they define the degree of the current scale upon which the new scale is formed. Unless otherwise specified, the mode is determined by examining the interval of the third in the old scale as formed on the note which will form the new "tonic." The scale is major or minor if this interval is major or minor, respectively. Also, unless otherwise specified, the minor scale is assumed to be natural.

The other brackets refer to thematic material defined by subroutines; these brackets are labeled with the names of their associated routines. These routines are all defined by indirect processes. Pitch parameters can be defined in terms of displacement by a given number of semitones or in terms of displacement by a given number of scale degrees. THEME is defined in precisely this manner, defining the ascending pattern by placing successive increments of one scale degree on a push-down stack, transposing the top note down an octave, and obtaining the descending pattern by popping these increments back -- all with respect to the harmonic minor scale. Consequently, the sequential descent of the thematic statements starting at measure five may be obtained by initializing the transposition parameter with the appropriate SETRAN and subsequently building the same attack on a different base.

It is in circumstances such as this one that the processes of nesting and exiting are invaluable; for throughout the development of this invention, the theme is stated in a variety of truncated forms. One example is the codetta in the upper voice at measure sixteen. This is realized by a PNEST instruction which enters the subroutine THEME, refraining execution until the address THEME1. An even more vivid example is the subroutine DTHEME, which produces the inverted pattern at measure 22. This pattern is essentially constructed by swapping the ascending and descending portions of THEME; and this, in turn, is realized by the proper insertion of nests and exits. Because thematic material is defined in terms of parametric alterations, it is, of course, necessary to use the instruction PNEST for nesting. Likewise, the subroutines exit with the instruction MOP TRP which, in the absence of any other mark on the transposition push-down list, restores it to its top level.

Trills are realized according to [Bach]'s own rules of ornamentation -- alternating thirty-second notes beginning with the <u>upper</u>

tone and ending with a thirty-second note and a dotted sixteenth note. This ending is handled by the subroutine TRICOD, while the body of the trill is taken care of by the subroutine TRILER.

Since all pitch computations are relative, some appropriate base point must be established. The instruction MELD is used to attach a program consisting of a single note word specifying that tone which is the "tonic" of the current scale sounded as a sixteenth note with legato articulation. This one note is run as a loop, so that all relative pitch computations relate back to this tonic. It is necessary to have one tonic for each voice because of certain subtleties in the score; and the assembly macro TONSET sets the tonic to the proper value, also assigning an octave specification.

We have tried to supply sufficient comments so that the reader may coordinate the EUTERPE program with the score. We have also provided a running account of the state of the scale array. With the exception of the initialization of the scale at the beginning of VOICE1, all modifications are performed by VOICE2 with the argument ALL, so that they also affect VOICE1. This emphasizes the fact that there is a single harmonic process which controls both voices. The primary exception occurs at measures 22 and 24, where VOICE1 does an extra RELKEY 5 which is popped off within DTHEME.

In all honesty it must be remarked that this particular coding was no easy matter. The fact that it did not boil down to trivial formulas is due tribute to the "inventiveness" of this composition.

Likewise, the manipulations of the scale parameter to account for the chromaticisms offer enlightening evidence of Bach's harmonic skill. Indeed, the difficulties in encoding this score have given the author an appreciation and a reverence for Bach's compositional technique. In preparing a program for this invention, one could readily acquire that "strong foretaste of composition" which Bach promised to those "lovers of the clavier" on the title page of these inventions([David]).

```
TITLE TWO-PART INVENTION IV

.INSRT EUTERP >

DEFINE TONSET N,OCTAVE
        MOVE 1,|16T(LEGATO)|
        ADDI 1,OCTAVE
        ADD 1,SCALE
        MOVEM 1,TONIC!N
TERMIN

LOC VOICE1

        TEMPO ALL,(8.)180.
        KEY D(HARM)      ;SETS KEY TO D MINOR
        GAMTRN
        TONSET 1,K
        MELD TONIC1
        MARK
        PUSHJ$ THEME     ;FIRST STATEMENT OF THEME AT MEASURE 1
        UNMARK
        K F      ;ARPEGGIATED COUNTERSUBJECT AT MEASURE 3
        K A 8T
        L D 8T
IMITAT: K G 8T
        L C SH 8T
        L E 8T
        RELPIT I
        PUSHJ$ THEME     ;DESCENDING SEQUENTIAL STATEMENT OF THEME AT
        SETRAN -1(AD)            ;MEASURE 5
        PUSHJ$ THEME+1
        TONSET 1,K       ;KEY IS PUSHED TO F MAJOR
        UNREL PIT
        SETRAN 3(AD)
        PUSHJ$ THEME+1
        SETRAN 0(AD)
        PNEST |(FIG)FIG1+1|      ;SECONDARY FIGURE AT MEASURE 11
        SETRAN -1(ASC)                   ;ALSO IN DESCENDING SEQUENCE
        EXIT1 LREST(TRP)
        PUSHJ$ FIG
        SETRAN 0(AD)
        K B FL 4T        ;CODETTA FOR FIRST SECTION AT MEASURE 15
        K A 8T
        K G 8T
        L C 16T
        SETRAN -2(AD)
        PNEST |(THEME)THEME1|    ;DESCENDING HALF OF THEME AT MEASURE 16
        SETRAN 0(ASC)
        K G      ;END OF FIRST SECTION AT MEASURE 17
        K G 8D
```

```
        K F
        UNMELD
        8T
        L C
        0
        MOVEI 1,17.
        MOVEM 1,TRILCT'
        PUSHJ$ TRILER     ;SUBROUTINE FOR TRILL AT MEASURE 19
        PUSHJ$ TRICOD     ;KEY IS PUSHED TO G MINOR
        MELD TONIC1
        RELKEY 5(NAT)     ;PUSHES KEY TO D MINOR
        TONSET 1,K        ;SUBROUTINE DTHEME HANDLES INVERSION AT MEASURE 22
                                   ;ANOTHER DESCENDING SEQUENCE
        PUSHJ$ DTHEME     ;POPS KEY TO G MINOR AFTER WHICH IT IS POPPED TO
                                   ;F MAJOR
        RELKEY 5          ;PUSHES KEY TO C MAJOR
        TONSET 1,K
        PUSHJ$ DTHEME     ;POPS KEY TO F MAJOR AFTER WHICH IT IS POPPED TO
        TONSET 1,K                 ;D MINOR AND PUSHED TO A MINOR
        EXIT1 THEME2+2
        PUSHJ$ THEME+1    ;STATEMENT OF THEME AT MEASURE 26
        UNREL TRP
        0
        RELKEY (MELO)     ;PUSHES HARMONIC MINOR TO MELODIC MINOR
        MUNREL TRP
        SETRAN -2(ASC)
        PNEST |(THEME)THEME1|      ;FRAGMENTS OF THEME USED AT MEASURE 28
        RELTRN -4                          ;AS BRIDGE TO NEXT DEVELOPMENT
        EXIT1 THEME2-1
        PNEST |(THEME)THEME1|
        EXIT THEME2
SEQ1:   EXIT1 THEME1-3(TRP)        ;ASCENDING SEQUENTIAL STATEMENT OF THEME
        PUSHJ$ THEME                       ;AT MEASURE 30
        RELTRN 2
        SKIPG TRILCT
        TRA LAST          ;SEQUENCE BREAKS WHEN VOICE2 STOPS TRILL
        PNEST |(THEME)THEME1-3|
        UNREL TRP
        TRA SEQ1
LAST:   EXIT1 THEME1(TRP)
        PNEST |(THEME)THEME1-3|
        UNEXIT THEME2
        MUNREL TRP
        SETRAN 0(DES)
        EXIT THEME2(TRP)
        PNEST |(THEME)THEME1|
        RELTRN 3(ASC)
        PNEST |(THEME)THEME1+2|
        UNEXIT THEME2
        UNREL KY          ;POPS BACK TO HARMONIC MINOR
```

```
L A 8D
L D
K B 8D
K A
K A 8D  ;KEY IS POPPED TO D MINOR AND PUSHED TO G MINOR
TONSET 1,K
RELTRN -4
EXIT1 LREST-1(TRP)
PNEST |(FIG)FIG+4|        ;FRAGMENT OF SEQQENTIAL FIGURE USED
MUNREL TRP
K F SH 8T                          ;AS BRIDGE AT MEASURE 38
K A 8T
K B FL 16T        ;KEY IS POPPED TO D HARMONIC MINOR AND PUSHED TO
TONSET 1,K                 ;NATURAL MINOR
RELTRN 2
PUSHJ$ THEME+1    ;STATEMENT OF THEME AT MEASURE 40
K A(LEGATO)       ;KEY IS POPPED BACK TO HARMONIC MINOR
L F
L E
L F 8T
K G 8T
L E 8T
R 8T
RELPIT I
PUSHJ$ THEME      ;RECAPITULATION AT MEASURE 44
K F(LEGATO)
UNREL PIT
L D 8T
K G 8D
L D
L C SH
L E
K A
L C SH
L D
K B
L C SH 8D
L D
L D
RELKEY (NAT)
RELTRN 1
EXIT1 DTHEME+2
PUSHJ$ DTHEME     ;THEME INVERSION USED AS CODA AT MEASURE 49
RGMTRN -3
PUSHJ$ DTHEM1
RELPIT I
0
UNREL PIT
K F 8T
K E
```

```
         0
         4D
         FINE

TONIC1:  K D 16T(LEGATO)
         TRA .-1

TONIC2:  K D 16T(LEGATO)
         TRA .-1

THEME:   REPEAT 6 |0
                   RGHTRN 1
|
         RELPIT -I
         0
         UNREL PIT
THEME1:  REPEAT 5 |UNREL TRP
                   0
|
THEME2:  RGHTRN 1
         (SLUR)
         MOP TRP

DTHEME:  EXIT1 THEME2(TRP)
         PNEST |(THEME)THEME1|
         UNREL KY
         TONSET 1,K
         RGHTRN -4
DTHEM1:  RELPIT I
         0
         UNREL PIT
         EXIT1 THEME1-3
         PUSHJ$ THEME+1
         0
         MOP TRP
```

```
FIG:      MARPAR TRP
          RGMTRN 5
          MARK TRP
          PUSHJ$ FIG1
          UNMARP TRP
          PUSHJ$ FIG1
          UNMARK TRP
          RGMTRN 1
          8T
          R 8T
LREST:    R 8T
          MOP TRP

FIG1:     0
```

```
            RGMTRN 1
            0
            RGMTRN 1
            0
            MOP TRP

TRILER:     RGMTRN 1
            32T
            UNREL TRP
            32T
            SOSLE TRILCT
            TRA .-5
            POPJ$

TRICOD:     RGMTRN 1
            32T
            UNREL TRP
            16D
            POPJ$

LOC VOICE2

            WAVE SQUARE
            R 2D
            RELPIT -1
            EXIT1 IMITAT     ;IMITATION OF VOICE1 AT MEASURE 3
            PUSHJ$ VOICE1+1 ;SETS KEY TO D MINOR
            K E 8T
            K G 8T
            L C SH 8T
SEQST:      PUSHJ$ SEQ       ;DESCENDING SEQUENTIAL BASS PATTERN AT MEASURE 7
            UNMELD
            RELKEY ALL,3     ;PUSHES KEY TO F MAJOR
            TONSET 2,K
            MELD TONIC2
            SETRAN -3(ASC)
            EXIT1 SEX
            PUSHJ$ SEQ
            PUSHJ$ THEME2
            SETRAN 0(AD)
            PUSHJ$ THEME+1   ;DESCENDING SEQUENTIAL STATEMENT OF THEME
            SETRAN -1(AD)            ;AT MEASURE 11
            EXIT1 THEME2
            PUSHJ$ THEME+1
            UNREL TRP
            0
            SETRAN -4(AD)
            PUSHJ$ THEME+1
            0
            RGMTRN 1
```

```
        8T
        RELPIT -I
        8T
        UNREL TRP
        SETRAN 0(AD)
        PUSHJ$ THEME     ;ASCENDING SEQUENTIAL STATEMENT OF THEME
        SETRAN 2(AD)             ;AT MEASURE 18
        EXIT1 THEME2(TRP)
        PUSHJ$ THEME+1
        RELKEY ALL,2(MELO)       ;PUSHES KEY TO G MINOR
        TONSET 2,K

        SETRAN -4(ASC)
        PUSHJ$ FIG       ;DESCENDING SEQUENCE OF SECONDARY FIGURE
                                 ;AT MEASURE 22
        UNREL ALL,KY     ;POPS KEY TO F MAJOR
        TONSET 2,K
        PUSHJ$ FIG
        UNREL ALL,KY     ;POPS KEY TO D MINOR
        RELKEY ALL,5(HARM)       ;PUSHES KEY TO A MINOR
        TONSET 2,K
        SETRAN 0(AD)
        K F 8T  ;ARPEGGIATED BRIDGE AT MEASURE 26
        UNMELD
        RELPIT I
        REPEAT 5 |0
                RGMTRN -2
  |
        MUNREL TRP
        UNREL PIT
        MELD TONIC2
        0
        RGMTRN -1
        EXIT1 THEME+12
        PUSHJ$ THEME     ;FRAGMENT OF THEME AT MEASURE 28 PRECEDING TRILL
        MOVEI 1,30.
        MOVEM 1,TRILCT

        PUSHJ$ TRILER    ;SUBROUTINE FOR TRILL AT MEASURE 29
MOUT:   PUSHJ$ TRICOD
        UNREL PIT
        REPEAT 5 |8T
                UNREL TRP
  |
        MUNREL TRP
        UNREL PIT
        K D 8T
        K E 8T
        K F 8T
        K D 8T
```

```
         K E 8T
         RELPIT -I
         K E 8T
         K A 16T
         RELPIT -I
         UNREL ALL,KY     ;POPS KEY TO D MINOR
         RELKEY ALL,4(HARM)       ;PUSHES KEY TO G MINOR
         TONSET 2,K
         EXIT THEME2(TRP)
         PUSHJ$ THEME+1  ;STATEMENT OF THEME AT MEASURE 38
         UNREL ALL,KY    ;POPS KEY TO D MINOR
         RELKEY ALL,(NAT)         ;PUSHES HARMONIC MINOR TO NATURAL MINOR
         8D
         TONSET 2,K
         RELTRN -2
         EXIT1 LREST-1(TRP)
         PNEST |(FIG)FIG+4|       ;FRAGMENT OF SECONDARY FIGURE AS BRIDGE
         MUNREL TRP
         K G 8T                           ;AT MEASURE 40
         L C 8T
         RELPIT I
         UNREL ALL,KY    ;POPS BACK TO HARMONIC MINOR
         RELKEY (MELO)   ;PUSH HARMONIC MINOR TO MELODIC MINOR
         SETRAN 2(ASC)
         PUSHJ$ THEME    ;STATEMENT OF THEME AT MEASURE 42
         UNREL KY        ;POP BACK TO HARMONIC MINOR
         SETRAN 0(AD)
         UNEXIT THEME2
         EXIT1 SEQST
         K F 16T(SLUR)
         NEST |(VOICE2)IMITAT-3| ;RECAPITULATION OF ARPEGGIATION
                                         ;AT MEASURE 44
CODA:    MARK
         PUSHJ$ THEME    ;RECAPITULATION OF THEME AT MEASURE 46
         UNMARK
         K G
         K A 8T
         J A 8T
CODA1:   J B FL 8D
         K C
         J B FL
         J A
         J G
         EXIT1 CODA1
         PNEST |(CODA)THEME1+1|  ;REPETITION OF CODA FIGURE AT MEASURE 50
         J D 4D
         FINE

SEQ:     8T
         RELPIT I
```

```
            8T
            UNREL PIT
            RGMTRN 1
            REPEAT 4 |RGMTRN 1
                         8T
|
SEX:        MOP TRP

END TUNE

;                                    EXAMPLE 4.3.2
```

Chapter 5

A Brief Look at Sonata Form

Thus far, most of our studies have been involved with musical examples from the mediaeval and the baroque periods. We have chosen these examples for two basic reasons: clarity and brevity. While neither of these qualities is necessarily an aesthetic desideratum, both are indeed beneficial in the demonstration of new tools and ideas.

When we come to the classical period, brevity remains the soul of wit and composers such as Mozart are praised for an art form "whose surface clarity and simplicity hold in delicate balance brilliance of technique and profundity of emotion" ([Burkhart]). However, the increasing sophistication of the musical audience allowed such terms as "simplicity" and "surface clarity" to be used on a much larger scale; and unfortunately, we are faced with a problem of memory shortage in the current EUTERPE system. For all its surface simplicity, there is still a great deal of information which needs to be coded in a Mozart sonata movement. There is also an aspect of performance which becomes much more evident in a machine interpretation of Mozart than in one of Bach. Although Rudolf Gerber cites Mozart's "Haydn" quartets as realizations of "the absolute equal value of the four stringed instruments," ([Mozart, 1930]) there is nothing like a presentation of these four parts on an equal plane to make the listener realize that this so-called equal value was attained by an alternation in assuming key roles rather than in a "perfect democracy."

So that we could at least scratch the surface of a study of sonata form, we undertook an investigation of the exposition from the fourth movement of K. 458, the B-flat major quartet, subtitled "The Hunt." Our program never got beyond the roughest of forms, and there is no need to present it in as great a detail as we allotted Example 4.3.2. However, we may at least review the salient points of our experiment.

In its most skeletal form this exposition is built on three key sources of material: a "first theme," a "second theme," and a codetta. These are all four-voice passages, but a great deal of material is shared among the four parts. In particular, there are many instances in the score where the counterpoint is reinforced by doubling at the octave or the tenth. Furthermore, there is a great deal of cross referencing of fragments, in the same manner as that of the invention program.

In this score the chromaticisms are more abundant than in the Bach invention, and they may not be taken into account as readily by shifts of the scale parameter. Consequently, this material is better represented by note words than by the indirect processes of the Bach program. Nevertheless, tonal transposition plays a key role in associating different sections of the score; and one must account for a shift of the scale parameter from B-flat major to F major.

Of course, this is only the most preliminary of sketches. While the exposition is the major element of a movement in sonata form, it is the development and recapitulation which put the exposition in its proper perspective. Nevertheless, the exposition exhibits a basic repertoire of developmental processes in those bridge passages which link the key elements; and it is generally this material which is expanded and worked out to a much greater extent in the development section. Thus, while the limitations of a particular system may restrict one to a study of the exposition section, such a partial study is certainly not without value.

Chapter 6

APPLICATIONS

6.1. Note Word Transcriptions

Like a first sight-reading, a literal transcription of a score into note words has little to offer other than a basic familiarity with the material. Its only desirable characteristic is that it may enable the user to hear musical passages of such complexity that he may be unable to hear them in his head or realize them at a piano. Such complexities fall into two categories: microtonal and rhythmical.

The most ambitious transcription of a score into note words was undertaken for both these reasons. In the spring of 1971, the composer Ezra Sims was invited to write an oboe quartet (i.e. oboe, violin, viola, violoncello) for an upcoming recital by Burt Lucarelli. Prior works by the composer (including the Octet for Strings, 1964, and the Third Quartet, 1964, the latter recorded by the Lenox Quartet for CRI) had employed both quarter tones and sixth tones ([Apel, 1969]). By this time, Sims was in the process of experimenting with "tonal" aspects of microtonality. He had derived a new scale in both diatonic and chromatic forms which we shall discuss in greater detail in Section 6.3, and he was evolving a theory of modulation among the tonalities defined by this diatonic scale.

Nevertheless, these theories were all at the experimental stage; and Sims was without a suitable keyboard with which he could objectively test his work. Consequently, with the author's assistance, Sims' composition was transcribed, as it was being written, into EUTERPE. Thus far, this transcription has encompassed most of the third movement of the quartet.

Sims' score also involved some complex applications of groupettes. These were realized by the RELTEM instruction and gave an accurate representation of the rhythmic textures of the composition. The resulting machine performances have been recorded, and the tapes are to be forwarded to the four musicians to assist their preparation of the live performance.

A similar experiment has been performed on a smaller scale by the author, again simply using note word transcription to get the "feel" of a musical passage. In his essay on quarter tones, Charles [Ives] discusses four possible ways of constructing chords given a quarter tone gamut. Ives discovered these chords empirically, having had a special quarter tone piano at his disposal. The author composed the following, four-voice fanfare to explore the sounds of these chords all guilt on the same root. This was then transcribed to EUTERPE purely for listening purposes. The score and program follow below. (The notation is again Sims'.)

Example 6.1.1

FARE FOR FANS

```
TITLE FARE FOR FANS

.INSRT EUTERP >

LOC VOICE1

        TEMPO ALL,(4)90.
        K C 1T

        K C 1T

        4T
        K G
        L C 2T

        R 8T
        L C 16T
        0
        4T
        R 8T
        L C 16T
        0
        8T3
        0
        0

        L E 8T
        L C 16T
        0
        K G Q 8T
        K E 16T
        0
        K C 8D
        16T
        8T
        K E

        K G 1T
        FINE

LOC VOICE2

        R 4T
        K G 2D

        1T

        4T
        K C
        K E Q
```

```
          K B Q

          K B P 2T
          K G

          K B P
          L C 16T
          K B P
          L C
          K B P
          L C 4T

          1T

LOC VOICE3

          R 2T
          8T
          K E
          K E FL
          0

          K E
          K E P 4T
          0
          0
          8T

          K E Q 2T
          K G 4T
          K E P

          K G 2T
          K B P

          K G Q 4T
          L C
          L E
          L G Q

          L C 1T

LOC VOICE4

          R 1T

          4T
          K B P(STACO)
          0
          16T(LEGATO)
```

```
        L C
        K B P
        L C

        L C 8T
        K B Q 4T
        8T
        K C 8T3
        L C
        K B Q
        K G 8D
        K E P 16T

        1T

        K C 8T
        K E 16T
        0
        K C 4T
        K G Q 8T
        16T
        0
        K E 8T
        K C

        1T

END TUNE

;                              EXAMPLE 6.1.2
```

6.2. "Fantasy on an English Carol"

By its very nature, EUTERPE is more oriented toward development than invention. Therefore, in matters of original composition, it is most useful in writing fantasies or variations on pre-specified material. By way of demonstration, we used EUTERPE to prepare a polyphonic fantasy in six voices on the mediaeval English carol, "Nova, Nova." Here is the carol in its original monophonic setting ([Greenberg]):

Example 6.2.1

Here is the code for the author's "Fantasy on an English Carol." All six voices are employed, although only the first voice starts at the top of its program space (COICE1). The second through fifth voices all start one word after their first word in program space, since this first word is initialized to CANCEL. These voices are initially silent and are cued in by the instruction TRA at the appropriate moments. VOICE6 doesn't use its own program space because it simply doubles VOICE3 three octaves higher.

```
TITLE FANTASY ON AN ENGLISH CAROL

.INSRT EUTERP >
.INSRT MODES >  ;LOAD FILE CONTAINING RHYTHMIC MODES

LOC VOICE1

         SETON 1                 ;FLAG TO KEEP TRACK OF VOICE5 ENTRY
         SETON 3                 ;FLAG TO KEEP TRACK OF VOICE4 ENTRY
         TEMPO ALL,(2)200.
         PITCH J C
         PUSHJ$ CAROL
         TRA 2,VOICE2+1
CAROL1:  REPEAT 2 PUSHJ$ CAROL+1
         TRA 3,VOICE3+1
         REPEAT 2 PUSHJ$ CAROL+1
         CANCEL 3,
         CANCEL 6,
         PUSHJ$ CAROL+1
         CANCEL 2,
         CANCEL 4,
         PUSHJ$ CAROL+1
         FINE

CHOR1:   MELD MODE5       ;FIRST PHRASE OF CHORUS
         L C
         K A
         L C
         K G
         UNMELD
         POPJ$

CHOR2:   MELD MODE1       ;SECOND PHRASE OF CHORUS
         K A
         K B
         L C
         K B
         UNMELD
         MELD MODE5
         K A
         TRA CHOR1+4

VERSE:   MELD MODE1       ;ENTIRE VERSE
         L F
         L F
         L E
         L D
         L F
         L D
         UNMELD
```

```
          MELD MODE5
           L C
           K A 2T
           K B 4T
           L C
          UNMELD
          MELD MODE1
           L C
           L C
           L D
           L D
          UNMELD
          MELD MODE5
           L C
           L C
           L F 2T
           L E 4T
           L D
           L C
           L C 2T
           K B 4T
           K A
          KELTEM 1(2)
          PUSHJ$ CHOR1+1    ;AUGMENTATION OF FIRST PHRASE OF CHORUS
          POPJ$ TEM

CHORUS:   PUSHJ$ CHOR1      ;ENTIRE CHORUS
          TRA CHOR2

CAROL:    PUSHJ$ CHORUS     ;BEGIN WITH CHORUS
          PUSHJ$ VERSE      ;ENTRY POINT AFTER INITIAL STATEMENT
          JUMPE 1,CHORUS            ;SKIPS START AFTER VOICE5 IS STARTED
          TRA 5,VOICE5+1
          TRA CHORUS        ;END WITH CHORUS

LOC VOICE2+1

          PUSHJ$ VA1        ;ACCOMPANY VERSE WITH REPETITIONS OF CHORUS
          PUSHJ$ CHOR1
          SKIPE 3                   ;SKIPS START AFTER VOICE4 IS STARTED
          TRA 4,VOICE4+1
          PUSHJ$ CHOR2
          TRA VOICE2+1      ;SIMPLE LOOP

LONG1:    EXIT1 CHOR1+5     ;RHYTHMIC ADJUSTMENTS FOR PHRASES OF CHORUS
          PUSHJ$ CHOR1
          TRA CHOR1+4

LONG2:    EXIT1 CHOR1+5
          PUSHJ$ CHOR2
```

```
        TRA CHOR1+4

SHA1:   EXIT1 CHOR1+3
        PUSHJ$ CHOR1
        TRA CHOR1+4

SHA2:   MELD MODE1
        TRA CHOR2+3

SHB1:   EXIT1 CHOR1+4
        PUSHJ$ CHOR1
        UNMELD
        POPJ$

SHB2:   EXIT1 CHOR1+4
        PUSHJ$ CHOR2
        UNMELD
        POPJ$

VA1:    PUSHJ$ CHOR1        ;ACCOMPANIMENT FOR VOICE2
        PUSHJ$ LONG2
        PUSHJ$ SHA1
        PUSHJ$ SHB2
        TRA CHORUS

VA2:    PUSHJ$ CHOR2        ;ACCOMPANIMENT FOR VOICE4
        PUSHJ$ LONG1
        PUSHJ$ SHA2
        PUSHJ$ SHB1
        PUSHJ$ CHOR2
        TRA CHOR1

LOC VOICE3+1

        PITCH I C
        PITCH 6,L C
        TRA 6,.+1
        RELTEM 1(4)         ;CHORUS IN AUGMENTATION
GROUND: MELD MODE5
        L C
        UNREL TEM
        RELTEM 1(5)
        K A
        UNREL TEM
        RELTEM 1(3)
        PUSHJ$ CHOR1+3
        UNREL TEM
        RELTEM 1(2)
        PUSHJ$ CHOR2
        UNMELD C,
```

```
        TRA 6,CHOR2
        PUSHJ$ CHOR1
        UNMELD 6,
        POPJ$ 6,TEM
        TRA 6,GROUND-1
        UNREL TEM
        TRA GROUND-1

LOC VOICE4+1

        SETZM 3             ;FLAG IS SET TO PREVENT RESTARTING
        PITCH J C
        PUSHJ$ CHOR1        ;VOICE EXCHANGE WITH VOICE2
        PUSHJ$ VA2
        PUSHJ$ VA2+4
        TRA .-2

LOC VOICE5+1

        SETZM 1             ;FLAG IS SET TO PREVENT RESTARTING
        PUSHJ$ CHORUS
        TRA CAROL1          ;DOUBLE VOICE1

END TUNE

;                           EXAMPLE 6.2.2
```

The score as given is executed by the subroutine CAROL. This subroutine calls the subroutine CHORUS, followed by the subroutine VERSE, followed by a transfer into CHORUS which eventually provides the subroutine exit. This carol has modal rhythms, so the file containing the rhythmic modes is loaded, and these modes are melded to a program of pitch specifications.

The chorus is in eight measures, and it is expressed as a sequence of two four-measure phrases. These phrases are the routines CHOR1 and CHOR2. This division into periods is not just a demonstration of symmetry. In the development section, we shall actually handle these two phrases independently.

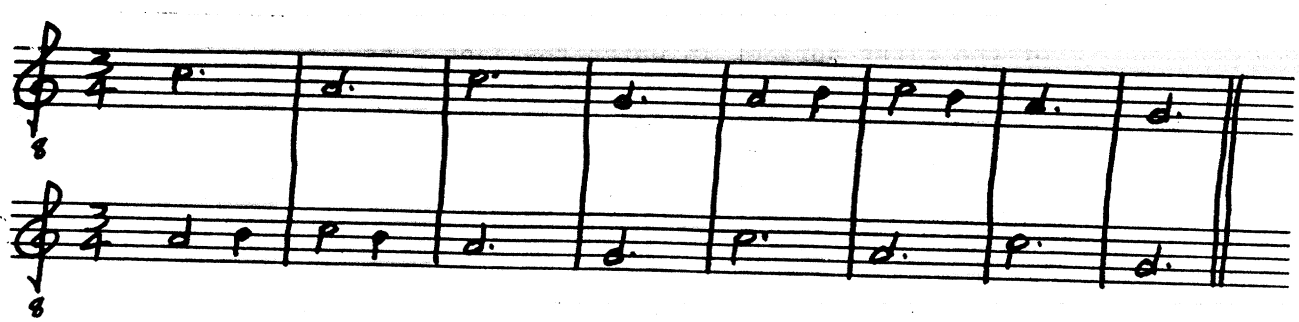
Observe that the verse section of this carol ends with an augmentation of the first four measures of the chorus. The basic idea which we decided to pursue was that of using the chorus contrapuntally against the verse. The verse, however, does not break down into a balanced set of four-measure phrases, so that were the chorus to be simply repeated successively against the statement of the verse, it would be "out of phrase" when the verse comes to the repetition of the chorus.

Therefore, we must determine the actual phrase structure of the verse and be prepared to transform the chorus to conform to this phrase structure. The portion of the verse preceding the augmentation of the chorus can be arranged into four phrases consisting of four measures, five measures, three measures, and three measures, respectively. The augmentation takes eight measures and may therefore accomodate two four-measure phrases.

The problem is now reduced to expressing the four-measure phrases of the chorus as either five- or three-measure statements. The easiest way to obtain a five-measure version is to simply repeat the last measure (i.e. note) of the phrase. The routines LONG1 and LONG2 accomplish this for each of the phrases of the chorus. For three-measure statements the easiest approach is to delete a measure. Since the last measure of the phrase establishes a sense of cadence, it is not always the most desirable for deletion. Consequently, we have made arrangements not only for the deletion of the last measure, but also for deletion of either the first or third measure of a phrase, as the case may be. The actual decision of which measure gets deleted is made on the basis of the intervals which are formed with the pitches in the verse. Finally, we can accompany the augmentation with a statement of the chorus in its entirety.

After the carol plays through once, VOICE2 is started with its contrapuntal line. This routine is stored at VA1; it consists of a four-measure statement of the first phrase, the five-measure prolongation of the second phrase, a three-measure statement of the first phrase, obtained by deleting the third measure (SHA1), a three measure statement of the second phrase obtained by deleting the last measure (SHB2), and a statement of the chorus against the augmentation.

Our next observation is that the symmetry of structure in the chorus makes it amenable to the technique of voice exchange. This is simply the technique of restating a given contrapuntal passage but interchanging the voices which we demonstrated in Section 4.2. Applieed to the chorus of "Nova, Nova," we obtain the following:



Example 6.2.3

The first four measures of the upper voice become the last four measures of the lower voice, and vice versa. Voice exchange arises naturally in rounds, as it is essentially a description of what happens once all the voices have entered.

Voice exchange was a prominent structural device used by English composers in the late thirteenth century ([Apel, 1969]), so it is not an unreasonable technique to apply to a carol from the fifteenth century. In this particular program voice exchange is conducted between VOICE2 and VOICE4. We have already assigned VOICE2 the task of accompanying the verse with repetitions of the chorus. In order to keep the structure in phase, statements of the chorus by VOICE4 must also be adjusted to the proper phrase lengths.

VOICE4 begins its first statement of the chorus four measures after the end of the second verse, and proceeds to follow essentially "behind" VOICE2. Its contrapuntal line is assigned in the routine VA2 and may be regarded as the "dual" of VA1. Thus, it consists of a four-measure statement of the second phrase, a five-measure statement of the first, a three-measure statement of the second, in which the first measure is omitted (SHA2), a three-measure statement of the first phrase, in which the last measure is omitted (SHB1), and a straight voice-exchanged statement of the chorus, as in the bottom line of Example 6.2.3, against the augmentation.

The use of augmentation in the statement of the verse motivated its application as a ground over which the rest of the counterpoint is built. An augmented statement of the entire chorus, of course, would have to be suitably adjusted to the phrase lengths in the verse. However, this can be accomplished by the use of the instruction RELTEM. In the augmentation the first measure of the chorus accompanies the first four measures of the verse, the next accompanies five measures, and the next two each accompany three. This brings us to the augmentation in the verse; so if we play the remaining four measures in the space of eight, this will sound as voice exchange in augmentation. The augmented line is played by two voices (VOICE3 is a very low register, VOICE6 in a very high one), we further reinforce the idea of voice exchange by having VOICE3 play the first phrase of the chorus and VOICE 6 play the second, both in augmentation, against the contrapuntal statement of the chorus in the other voices.

The only other voice which remains is VOICE5. This voice simply reinforces VOICE1 at the octave from the beginning of the second statement of the chorus to the end of the piece. After all the voices are sounding, they are gradually taken away. First the augmentations drop out, then the contrapuntal line provided by the chorus. Finally, the carol is stated once more, in octaves, by VOICE1 and VOICE5. Here is the resulting score which arises from this program:

FANTASY ON AN ENGLISH CAROL


S.
T.I.
A.


S.
A.
T.I.
S.
A.
T.I
T.II


S.
A.
T. I
T. II
S.
A.
T. I.
T. II.


S.
A.
T.I
T.II
S.
A.
T.I
T.II
B.


S'
S.
A.
T. I.
T. II.
B.


S.¹
S.
A.
T.I.
T.II.
B.


S.
S.
A.
T.I.
T.II.
B.
S.
A.
T.I.
T.II.


S.
A.
T. I.
T. II.

Example 6.2.4

6.3. Studies in Microtonality

The decision to temper EUTERPE's octave into 72 equal divisions was provoked by Ezra Sims, whose repertoire had included both quarter-tone and sixth-tone inflections. The principal motivation behind the use of microtones was the search for a better approximation to the pitches of the physical overtone series. The twelve-tone temperament approximates the third harmonic (i.e. the interval of a fifth) to within two cents (a cent is one hundredth of a semitone). However, it differs from the fifth harmonic (a major third) by 13.7 cents and from the seventh harmonic by 31.2 cents.

We have not labeled the interval associated with the seventh harmonic because it is so incompatible with the twelve-tone temperament. The interval of a minor seventh is 31.2 cents too high, and Sims refers to it as a "low" seventh. Nevertheless, the interval was recognized by Giuseppe Tartini in his Trattato di Musica, and he even accorded the interval a special chromatic notation, ([Fokker]).

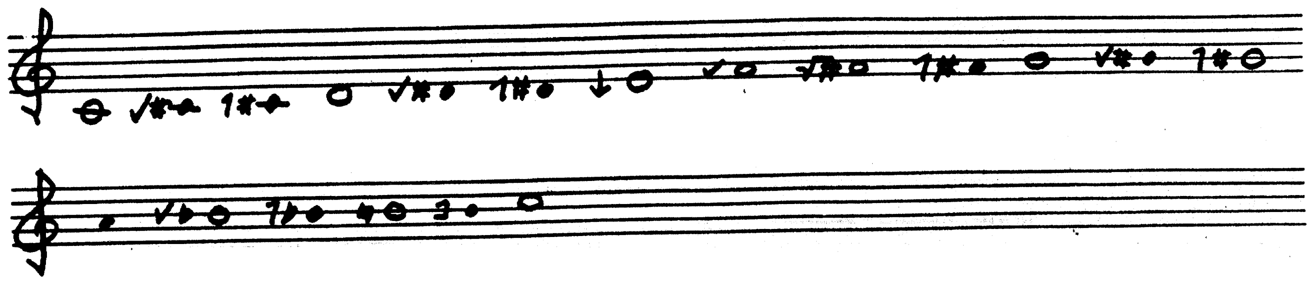
It was primarily the desire for a better approximation to the seventh harmonic which brought about attempts at other temperaments of the octave. Fokker cites two of these for the goodness of their approximation to the seventh and lower harmonics: the 31-tone temperament of Huygens and the 53-tone temperament of Mercator. These having the following differences (in cents) for the actual frequencies of the harmonic series:

	3rd harmonic	5th harmonic	7th harmonic
31-tone	5.2	0.8	1.1
53-tone	0.1	1.4	4.8

All of these differences are preferable to those of the twelve-tone temperament; however, these scales have the disadvantage of being incompatible with conventional chromatic music.

Because we wanted this compatibility, we turned, instinctively, to the 72-note temperament. It turns out, however, that the differences in this system are not that unfavorable. The third harmonic is still 2.0 cents off; exactly as in the chromatic gamut. Observe that this is preferable to Huygens' 31-tone temperament but inferior to the 53-tone system. The fifth harmonic is now 3.0 cents off, and the seventh harmonic is 2.2 cents off. While the 31-tone system has better fits for both these intervals, the 72-tone system has the advantage that its maximum error over all three intervals (3.0) is less than the same maximum in either of the above two systems. Hence, this temperament is certainly a useful one.

In his own experiments, Sims did not use the 72-tone gamut in its entirety. Rather, he extracted an unequally-tempered "chromatic scale" of eighteen pitches in which was embedded a nine-tone "diatonic scale." Here is Sims' gamut; the white notes are the diatonic tones:



Example 6.3.1

To familiarize the ear with this gamut, Sims wrote a series of etudes to be performed by EUTERPE. These outlined the structures of the two scale forms as well as the various possible intervallic combinations. Here is the code for these etudes:

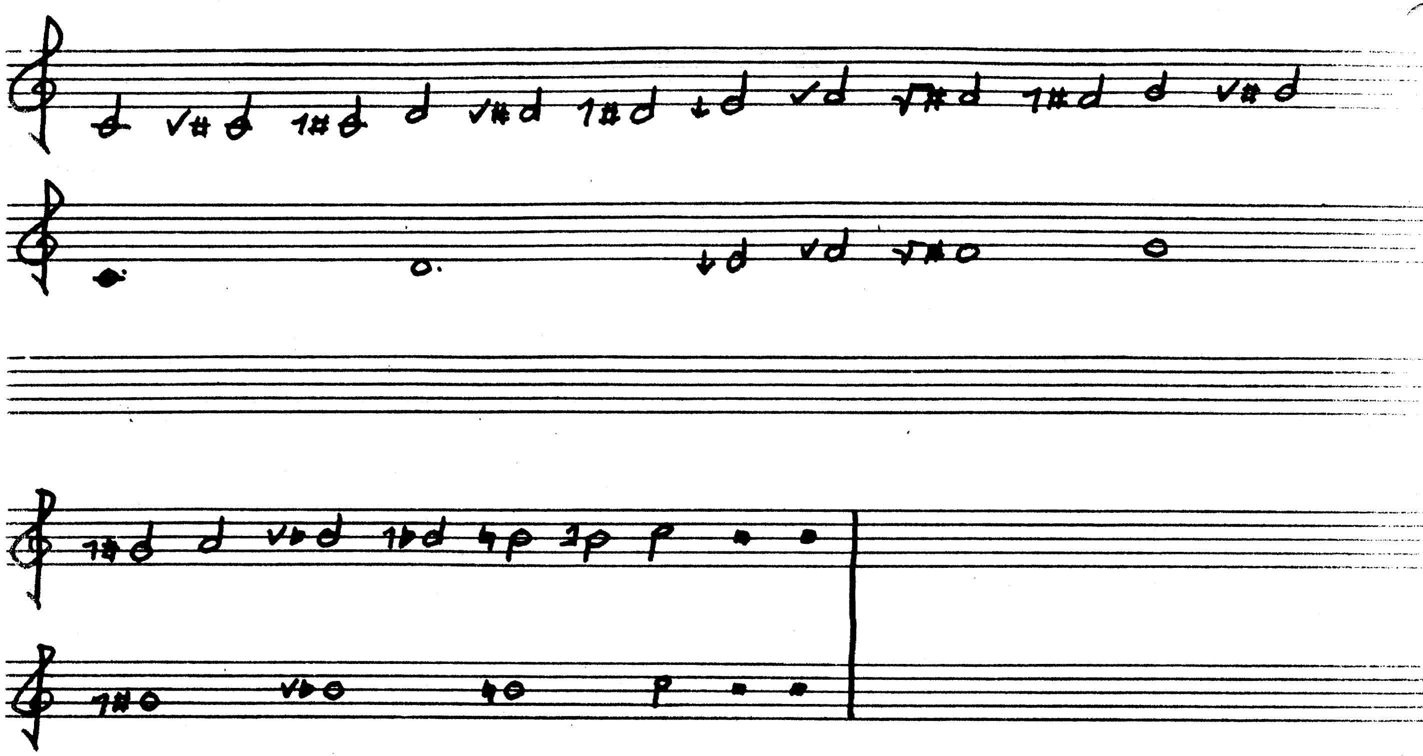
```
TITLE EXERCIZES FOR BURT

.INSRT EUTERP >

LOC VOICE1

          TEMPO ALL,(4)60.
          MOVEI 1,11.         ;COUNTER FOR FIRST TWO ETUDES
          MOVEI 2,0
          PUSHJ$ CHROM1
          ADDI 2,1
          TRA 2,CHR12
CHR11:    MOVEI 1,21.         ;COUNTER FOR THIRD ETUDE
          TEMPO (4)120.
          PUSHJ$ CHROM1
CHR21:    ARTIC ALL,STACO
          MOVEI 1,21.         ;COUNTER FOR FOURTH ETUDE
          MOVEI 2,3
          TRA 2,CHR22
          PUSHJ$ CHROM1
          TRA ALL,BLAT        ;LAST NOTE
CHR12:    UNREL TEM           ;VOICE2 FOR THIRD ETUDE
          TRA CHROM1
CHR22:    TEMPO ALL,(4)240.           ;VOICE2 FOR FOURTH ETUDE
          MOVEI 6,9.
          R 4T
          MOVEI 4,19.
          MOVEI 5,0
CHROM2:   XCT CHROM1(5)
          JUMPN 6,AOS5
          R 1T
          MOVEI 6,9.
          AOJA 5,AOS5+3
AOS5:     XCT CHROM1(5)
          AOS 5
          SOS 6
          SOJG 4,CHROM2
          TRA CHROM2-2
BLAT:     TEMPO ALL,(4)60.
          K C 1T(SLUR)
          8T
          FINE

CHROM1:   K C 2T              ;CHROMATIC GAMUT
          K C SH % 2T
          K C SH $ 2T
          K D 2T
          K D SH % 2T
          K D SH $ 2T
          K E U 2T
```

```
        K F % 2T
        K F SH P 2T
        K F SH $ 2T
        K G 2T
        K G SH % 2T
        K G SH $ 2T
        K A 2T
        K B FL % 2T
        K B FL $ 2T
        K B 2T
        K B Q 2T
LC:     L C 2T
        CAILE 2,2
        TRA LOUP
        MOVE 3,VOICE
        CAIN 3,2            ;VOICE2 SKIPS REST INTERRUPT
        TRA CHROM1
        PUSHJ$ 2,.+1
        R 1T
        POPJ$ 2,
        SOJG 1,CHROM1
        POPJ$
LOUP:   R 1T
        TRA .-3
```

```
LOC VOICE2

DIATO:  SKIPN 2
        RELTEM (3)
        K C 2T(SLUR)        ;DIATONIC GAMUT
        K D 2T
        SKIPN 2
        UNREL TEM
        K E U 2T
        K F % 2T
        SKIPN 2
        RELTEM (2)
        K F SH P 2T
        K G 2T
        K G SH $ 2T
        K B FL % 2T
        K B 2T
        SKIPN 2
        UNREL TEM
        L C 2T
        ADDI 2,1
        RELTEM (19.)
        TRA DIATO

END TUNE
;
```

EXAMPLE 6.3.2

CHROM1 is stored as a subroutine while DIATO is the first word of code for VOICE2. The first exercise is simply the simultaneous soundings of the scale in its chromatic (VOICE1) and diatonic (VOICE2) versions. The diatonic tones are sustained against the chromatic tones until the chromatic version ascends to the next diatonic tone. The score would appear as follows:



Example 6.3.3

The different durations of the diatonic tones are realized by RELTEM instructions. Accumulator 2 is used as a flag to determine whether or not these instructions are executed. Accumulator 1 is used as a counter to determine the number of iterations of the chromatic scale (once for this passage and ten times for the next exercise). VOICE1 handles all initialization, so that when it enters CHROM1, VOICE2 is ready to process DIATO. After the scales are completed, both voices execute a whole rest according to the following piece of code in CHROM1:

```
PUSHJ$ 2,.+1
R 1T
POPJ$ 2,
```

Example 6.3.4

These instructions make full use of the interrupt capabilities of intervoice control which we described in Section 1.2.3.

The next set of exercises consists of the sounding of the chromatic scale in its entirety against each tone of the diatonic scale. (This accounts for the remaining ten repetitions through CHROM1.) When VOICE2 is finished with its rest, is alters the flag in accumulator 2 and does a RELTEM (19). Now each tone of the diatonic scale will endure for an entire statement of the chromatic scale. The interrupt capabilities allow for the insertion of a rest at the end of each iteration of the chromatic scale.

When this is completed, VOICE1 pops out of CHROM1, readjusts the flag in accumulator 2 and begins the section at CHR11, transfering VOICE2 to the corresponding CHR12. Tempo adjustments give VOICE1 a tempo of 120 quarter notes per minute, while VOICE2 returns to 60 quarter notes per minute. Both voices enter CHROM1, so that they both play the chromatic scale, with VOICE1 playing twice as fast as VOICE2. Whenever VOICE1 hits the top of the scale, it interrupts VOICE2 with a rest; but when VOICE2 hits the top, there is no interrupt. VOICE1 cycles the scale 21 times, so that it ends on a high C and VOICE2 ends on the low C.

Next, the whole routine is repeated, this time with stacatto articulation at a tempo of 240 quarter notes per minute. This time around, rather than playing twice as slow, VOICE2 plays each of its pitches twice. This is achieved by indexing through CHROM1 and using the XCT instruction. At the end of this passage, all six voices sound middle C together -- a sort of final confirmation of the tonality.

These etudes provided both a fundamental exercise in programming and a composition for ear training. When the program was debugged, the final version was recorded for the musicians who will be playing Sims' Oboe Quartet, so that they may familiarize themselves with the underlying tonalities. The programming is based on a few simple contrapuntal concepts, all of which were realized with little difficulty.

BIBLIOGRAPHY

[AR] Antiphonale Sacrocanctae Romane Ecclesiae Pro Diurnis Horis, Desclee & Co., Tournai, Belgium, 1949.

[Apel, 1953] W. Apel, The Notation of Polyphonic Music: 900-1600, The Mediaeval Academy of America, Cambridge, Massachusetts, 1953.

[Apel, 1958] W. Apel, Gregorian Chant, Indiana University Press, Bloomington, Indiana, 1958.

[Apel, 1969] W. Apel, Harvard Dictionary of Music, The Belknap Press, of Harvard University Press, Cambridge, Massachusetts, 1969.

[Babbitt] M. Babbitt, Twelve-Tone Invariants as Compositional Determinants in Problems of Modern Music, P. H. Lang, ed., W. W. Norton & Co., Inc., New York, New York, 1962.

[Bach] J. S. Bach, Fifteen Two-Part Inventions; Fifteen Three-Part Inventions; Italian Concerto; Goldberg Variations, Lea Pocket Scores, New York, New York.

[Bauer-Mengelberg] S. Bauer-Mengelberg and M. Ferentz, On Eleven-Interval Twelve-Tone Rows, Perspectives of New Music, 3 (1965), pp.93-103.

[Burkhart] C. Burkhart, Anthology for Musical Analysis, Holt, Reinhart & Winston, New York, New York, 1964.

[Coussemaker] E. de Coussemaker, Histoire de l'Harmonie au Moyen Age, Librairie Archeologique de Victor Didron, Paris, France, 1852.

[David] H. T. David and A. Mendel, eds., The Bach Reader, W. W. Norton & Company, Inc., New York, New York, 1966.

[Davison] A. T. Davison and W. Apel, Historical Anthology of Music: Oriental, Medieval and Renaissance Music (Vol. I), Harvard University Press, Cambridge, Massachusetts, 1968.

[Dinwiddie] J. Dinwiddie, Mewantemooseicday: John Cage in David, 1969, Source, 7 (1970), pp.21-26.

[Ferretti] P. Ferretti, Esthetique Gregorienne, Vol. I, French translation by A. Agaësse, Desclee & Co., Tournai, Belgium, 1938. (English translations by the author.)

[Fokker] A. D. Fokker, Les Mathématiques et la Musique, Archives du Musée Teyler, 10, (1947), pp.1-32.

[Franconis] M. Franconis, Ars Cantus Mensurabilis, in Scriptorum de Musica Medii Aevi, Vol. I, E. de Coussemaker, editor, Georg Olms Verlagsbuchhandlung, Hildesheim, Germany, 1963. (English translation in O. Strunk, Source Readings in Music History: Antiquity and the Middle Ages (Vol. I), W. W. Norton & Co., Inc., New York, New York, 1965.

[Forte] A. Forte, Tonal Harmony in Concept and Practice, Holt, Rinehart & Winston, New York, New York, 1962.

[Fux] J. J. Fux, The Study of Counterpoint, A. Mann, translator and editor, W. W. Norton & Co., Inc., New York, New York, 1965.

[Galler] B. A. Galler, The Language of Computers, McGraw-Hill Book Company, Inc., Garden City, New York, 1963.

[Hewitt] C. Hewitt, PLANNER: A Language for Manipulating Models and Proving Theorems in a Robot, Artificial Intelligence Memo No. 168, Memorandum MAC-M-386, Project MAC, Massachusetts Institute of Technology, August, 1970.

[Hiller] L. A. Hiller, Jr. and L. M. Isaacson, Experimental Music, McGraw-Hill Book Company, Inc., New York, New York, 1959.

[Hymnal] The Hymnal, 1940.

[Ives] C. Ives, Essays Before a Sonata, The Majority and Other Writings, H. Boatwright, editor, W. W. Norton & Co., Inc., New York, New York, 1970.

[Knuth, 1968] D. E. Knuth, The Art of Computer Programming, Vol. I, Fundamental Algorithms, Addison-Wesley Publishing Co., Reading, Massachusetts, 1968.

[Knuth, 1969] D. E. Knuth, The Art of Computer Programming, Vol. II, Seminumerical Algorithms, Addison-Wesley Publishing Co., Reading, Massachusetts, 1969.

[Langer] S. K. Langer, Philosophy in a New Key, The New American Library, New York, New York, 1951.

[LaRue] J. LaRue, Guidelines for Style Analysis, W. W. Norton & Co., Inc., New York, New York, 1970.

[Liber] Liber Usualis, Desclee & Co., Tournai, Belgium, 1963.

[Ligeti] G. Ligeti, Pierre Boulez, Die Reihe, 4, (1960), pp. 36-62.

[Mason] D. G. Mason, Beethoven and his Forerunners, The Macmillan Company, New York, New York, 1918.

[Merritt] A. T. Merritt, Sixteenth-Century Polyphony, Harvard University Press, Cambridge, Massachusetts, 1946.

[Meyer] L. B. Meyer, Emotion and Meaning in Music, The University of Chicago Press, Chicago, Illinois, 1956.

[Minsky] M. Minsky, "Matter, Mind and Models," Artificial Intelligence Project Memo No. 77, Memorandum MAC-M-230, Project MAC, Massachusetts Institute of Technology, March, 1965.

[Mozart, 1930] W. A. Mozart, String Quartet (The Hunt) K.-V. No.458, R. Gerber, editor, Ernst Eulenberg, Ltd., London, England, 1930.

[Mozart, 1941] W. A. Mozart, The Dice Composer, A. Lazlo, arranger, Guild Publications of Art and Music, New York, New York, 1941.

[PDP-6] Programmed Data Processor-6 Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1964.

[Reese] G. Reese, Music in the Middle Ages, W. W. Norton & Co., Inc., New York, New York, 1940.

[Robertson] A. Robertson and D. Stevens, editors, The Pelican History of Music, Vol. I, Ancient Forms to Polyphony, Penguin Books, Baltimore, Maryland, 1960.

[Rokseth] Y. Rokseth, Polyphonies du XIIIe Siècle, Louise B. M. Dyer, Paris, France, 1935.

[Salzer] F. Salzer, Structural Hearing, Dover Publications, Inc., New York, New York, 1962.

[Schenker] H. Schenker, Harmony, O. Jonas, editor, E. M. Borgese, translator, The University of Chicago Press, Chicago, Illinois, 1954.

[Schoenberg] A. Schoenberg, Structural Functions of Harmony, L. Stein, editor, W. W. Norton & Co., Inc., New York, New York, 1969.

[Sims] E. Sims, Personal communication.

[Smoliar] S. Smoliar, EUTERPE: A Computer Language for the Expression of Musical Ideas, Artificial Intelligence Memo No. 129, Project MAC, Massachusetts Institute of Technology, April, 1967.

[Spiess] L. B. Spiess, Polyphony in Theory and Practice from the Ninth Century to the Close of the Thirteenth Century, Ph.D. Dissertation, Harvard University, 1947.

[Winograd, 1968] T. Winograd, Linguistics and the Computer Analysis of Tonal Harmony, Journal of Music Theory, 12, (1968), pp.2-49.

[Winograd, 1971] T. Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language MAC TR-84, Project MAC, Massachusetts Institute of Technology, February, 1971.

[Wolf, 1904] J. Wolf, Geschichte der Mensural-Notation von 1250-1460, Breitkopf & Härtel, Leipzig, Germany, 1904.

[Wolf, 1963] J. Wolf, Handbuch der Notationskunde, Vol. I, Georg Olms Verlagsbuchhandlung, Hildesheim, Germany, 1963.