

THE UNIVERSITY OF ALBERTA

A COMPUTER-AIDED MUSIC FACILITY

by



Lorne Dale Fredlund

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

Spring, 1973

ABSTRACT

A survey of present computer applications in the field of music is presented and evaluated, and a proposal for a well-balanced computer-aided music facility is put forth. In the proposed system, special emphasis is placed on the purpose of the graphical display unit in meeting the needs of the traditional composer. The requirements of a traditional composer are investigated and several music transformations are proposed to aid in music composition. A partial implementation demonstrates the feasibility of aiding the traditional composer. Further extensions and improvements of the implementation are suggested.

ACKNOWLEDGEMENTS

I express my appreciation to Professor J. R. Sampson for his advice, criticism, and guidance throughout the duration of this research. In addition, I wish to express my gratitude to Professor B. Harris and Professor R. L. Ware for their support, encouragement, and continued interest in the project. Special thanks are extended to my wife Peggy for her assistance in the initial typing and proofreading of the thesis.

The financial assistance supplied by the Department of Computing Science and the National Research Council of Canada is gratefully acknowledged.

TABLE OF CONTENTS

	Page
CHAPTER I: COMPUTER APPLICATIONS IN MUSIC	
1.1 Introduction	1
1.2 Computer Composition	2
1.3 Computers in Music Research	3
1.4 The Computer as an Orchestra	5
1.5 Computer-Aided Music Composition	6
1.6 Evaluation	8
CHAPTER II: COMPONENTS OF A COMPUTER-AIDED MUSIC FACILITY	
2.1 Introduction	10
2.2 A well-balanced system	10
2.3 Graphical display unit (GDU)	19
2.3.1 GDU as a control unit	20
2.3.2 Visual representation of a music score	21
2.3.3 Initiation of transformations	23
2.4 Music transformations and arrangements	23
2.4.1 Arranging capability	24
2.4.2 Transposition	25
2.4.3 Copying	29
2.4.4 Inversion	30
2.4.5 Retrograde	32

2.4.6 Augmentation and diminution	32
2.5 Shortcomings of the GDU	34
2.6 Data structure	35
2.6.1 Introduction	35
2.6.2 Proposed data structure	37
2.6.2.1 Level 1	38
2.6.2.2 Level 2	42
CHAPTER III: IMPLEMENTATION	
3.1 Introduction	49
3.2 Commands	57
3.3 Data structure	68
3.3.1 Level 1	68
3.3.2 Level 2	68
3.3.3 Level 3	71
CHAPTER IV: CONCLUSIONS	
4.1 Evaluation of the proposed system	73
4.2 Extent of implementation	75
4.3 Direction of extensions	76
REFERENCES	78

LIST OF FIGURES

Figure	Page
1.1 Block diagram of the Pulfer system	7
2.1 Block diagram of the proposed system	11
2.2 A sample music plot	17
2.3 An array of intervals above and below any pitch	28
2.4 Order of the flat key signatures	29
2.5 Order of the sharp key signatures	29
2.6 A sample music staff	38
2.7 Example of a block of staves	39
2.8 Format of vector PTSS	40
2.9 Note or rest node	44
2.10 Structure of an M-note chord	45
2.11 Node in MKS	47
2.12 MKS in the data structure	47
3.1 Alphanumeric/function keyboard	52
3.2 Initial display on screen	53
3.3 A type I score	54
3.4 A type II score	55
3.5 A type III score	56
3.6 Transformations: Reproduce and Copy	65
3.7 Transformations: Invert and Retrograde	66
3.9 A one-note node	69

3.10	Format of information in the attribute field ..	69
3.11	A two-note node	70
3.12	Node in MKS	71

CHAPTER I

COMPUTER APPLICATIONS IN MUSIC

1.1 Introduction

The use of computers in the fine arts (art, architecture, music) has become an area of major interest. The primary incentive for scholars in the fine arts is the ability to manipulate rapidly large amounts of data for purposes of analysis and synthesis. But the problems facing computer application to the fine arts are extensive. Even after a formal structure and basic parameters have been established, an efficient interface with the computer is a principal difficulty.

The major and most significant computer applications in the fine arts have involved the field of music. This is primarily due to the well defined components, rules and basic mathematical structure of the field. The types of applications have been as wide and varied as the field of music itself. These applications will be considered in four major categories.

1.2 Computer Composition

The problems of music composition by computer are different from those of traditional music composition (i.e. composition by human composers). Computer composition could be defined as extracting order out of a chaotic multitude of available possibilities. This extraction is attained by applying certain mathematical operations derived from probability theory. On the other hand traditional composition cannot be so easily or concisely defined. It is sufficient to point out, however, that traditional music composition has been primarily melodic and more instinctive than reasoned. The essential difference between the two types of composition is that the computer composer is attempting to replace variety and creative elements in music with random generation governed by a predefined bias. The results have been interesting.

The first well-known computer composition was the Illiac Suite for String Quartet by Lejaren A. Hiller Jr. and Leonard M. Isaacson [4]. The score of this piece was published in 1957. It stimulated much interest and extensive comment. Since 1957 a miniature explosion of computer compositions [3] has occurred. Analysis of the computing techniques has been extensive and has resulted in many interesting discoveries. But discussion of these results is beyond the scope and purpose of this paper.

Proponents of computer composition are very excited about possible future developments--an excitement not widely shared by musicians and composers. In 1960, the Newsweek music critic commented on the Illiad Suite: "Premiered in Chicago, the stilted, lifeless performance aroused much curiosity but not much envy from flesh-and-blood composers."

[1] It is interesting to note that just prior to the concert in question, the quartet which was to play the Illiad Suite refused to do so. A tape recording previously made by the quartet was played for the audience.

Though opinions vary, it is important to note that current analysis of music composition techniques is providing new theory and possibly more useful parameters for the computer composer. It is evident that this field of research is not stagnant and that only time will reveal its possibilities.

1.3 Computers in Music Research

This area of application appears to enjoy general acceptance from musicologists. The primary reason for this acceptance concerns the unintrusive nature of the application. To the computing scientist, however, this area of application presents the basic problem of input to a

computing system. In a later chapter this problem will be considered in some detail.

The computer has been applied widely in the field of music research [6,7] (see [7] for a broad spectrum of articles referring to music research; especially relevant are articles by Suchoff pp. 193-206, Crane and Fiehler pp. 209-222, Gabura pp. 223-276, Karp pp. 293-295). One area of application concerns the general category of information storage and retrieval. Here the computer searches large scores for patterns which are being investigated for possible duplications, borrowings and identification of anonymous works.

A second use of the computer in music research involves harmonic, melodic, thematic and form analysis [6,7] (most relevant articles in [7] are by Fuller pp. 123-131, Jackson pp. 132-146, Gabura pp. 223-276). The computer operates primarily in a statistical role, providing the musicologist with a very detailed and powerful multidimensional view of the music under consideration. The results have been encouraging. The computer has increased the scope of research an individual scholar can undertake. No longer are the painstakingly slow processes of music sampling and the non-creative aspects of analysis any hindrance to the musicologist. Essentially the computer has allowed the musicologist to broaden his data base to include

all relevant material, an accomplishment which would have been impossible to do without the time advantage provided by the computer.

1.4 The Computer as an Orchestra

M.V. Matthews of Bell Telephone Laboratories has considered the analogy of the computer to the orchestra in the conductor-orchestra relationship [8,9]. In the Groove system which incorporates his concept, the conductor controls the sound production by the computer by "turning knobs and pressing keys rather than waving a stick...". The computer conductor enjoys not only the influence of the conventional conductor but, for example, may insert an additional voice and allow the computer to provide the rhythm. The computer conductor also possesses full editing capabilities and may halt a performance at any time sustaining the current sound. He may then edit the score at that point and both hear and see the changes as they are being made. The score appears on a cathode ray tube (CRT) display in conjunction with a vertical set of points which move through the score as it is being performed.

Matthews comes to the following conclusion concerning the Groove system: "The Groove System has proven to be so powerful, so rapid and so pleasant to use that the

hybrid computer--a digital device driving an analog synthesizer--will be the central facility in the electronic studios of the future." [8]

1.5 Computer-Aided Music Composition

Whereas the Groove system is primarily concerned with composer control of the computer in sound production, Ken Pulfer of the National Research Council Of Canada has designed and implemented a more complete and functional system for the composer [1]. Although the objective of the National Research Council has been to exploit technology primarily in the input-output interface rather than to facilitate music composition, Pulfer has also attempted to allow and encourage traditional composition.

In Pulfer's system (see Figure 1.1), the composer has two primary input devices. For input of music phrases, a piano-type keyboard allows the composer to play the sequence of notes, which will in turn be processed by the computer, stored, and displayed on a CRT. The composer can then edit his composition via the CRT display. An alphanumeric keyboard terminal is also available for general program control.

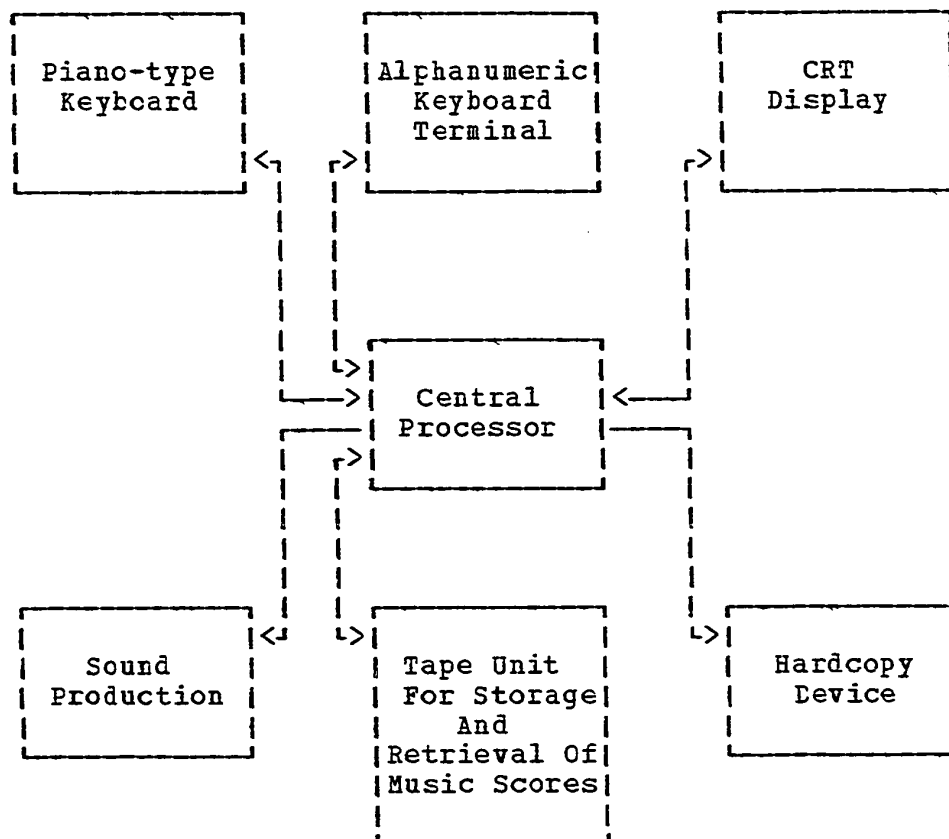


Figure 1.1 Block diagram of the Pulfer system

The computing facility allows for translation of the stored composition into two different forms, under the direction of the user or composer. A sound production allows the composer to appreciate or criticize aurally the results of his labour. A permanent hardcopy of the composition is realized by a plotting facility.

Essentially, a computing facility of the nature described provides the composer with the equivalent of the pen in the input interface, paper in the computer memory, musician and instrument in the output interface and finished score in the plotting facility.

Pulfer has in addition made a step towards implementing a music transformation. By music transformation is meant a mathematical mapping of a sequence of music symbols. His system provides the user with the power to initiate the transposition of a melody into any key. It is evident that one of the primary advantages of a facility such as has just been described is the power available to the user in initiating music transformations such as transposition. This idea will be dealt with in more detail in Chapter II.

1.6 Evaluation

From the previous discussion it is evident that much has been accomplished in computer applications in the music field. Computer composition and research analysis have been exploited with partial success. However, these areas are only a part of what should be included in a computer support system for a music institute or music department. Apart from

the work of Pulfer little has been accomplished in providing a general computer-aided facility which would involve the traditional composer and provide him with powerful tools such as music transformations.

CHAPTER II

COMPONENTS OF A COMPUTER-AIDED MUSIC FACILITY

2.1 Introduction

Across North America, music departments and centers of music research are establishing electronic music studios. The majority of these installations are concerned with electronic synthesis of sound. A small percentage of the installations have also made attempts at including programs for music research or computer composition, or both. But well-balanced systems have not usually been created. The reasons range from monetary problems to conservatism on the part of music faculties. It appears that the evolution of a well-balanced computer-aided music facility will be as time dependent as most evolutionary processes.

2.2 A Well-Balanced System

The following proposal for a well-balanced system will involve an indication of its capabilities, the medium via which each capability is realized, and an indication of the present state of implementation for each capability, if relevant. Numbered paragraphs in the following discussion correspond to block numbers in Figure 2.1.

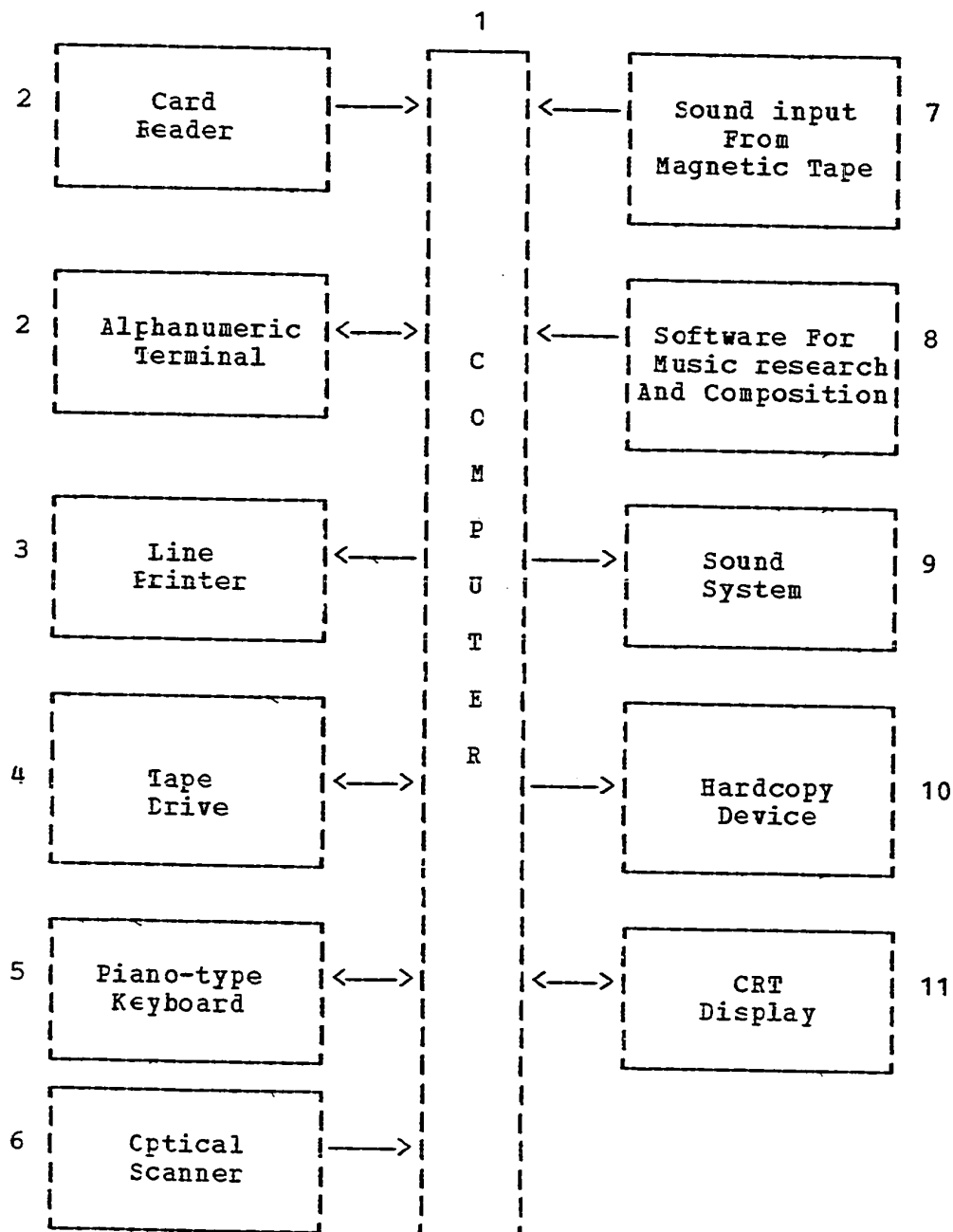


Figure 2.1 Block diagram of the proposed system

(1) Basic to the system is a medium sized computer. Storage peripherals (disks) would be commensurate with the software involved.

(2) An alphanumeric terminal and card reader would be necessary for general program control and the implementation of software packages.

(3) A high speed line printer would be included for output from research programs (music analysis in its various forms).

(4) A tape drive would facilitate short term storage and retrieval of data previously generated by users.

In order for a computer support system for a music department to have potential, it must allow for high speed input of music data. This requirement is dependent upon the user. The composer wants input via a familiar medium at speeds at least as fast as his traditional medium of pen and paper. The researcher is interested only in being able to process large volumes of data in the computer and finds any high speed input medium satisfactory. The requirements of the general music student and educator would be satisfied by

subsets of the input media provided for the composer and researcher.

(5) A piano-type keyboard interfaced with the computer allows the composer to work in a familiar medium. Input via the piano keyboard would be substantially faster than his present pen and paper. However, immense problems must be overcome in implementing such an input device. For example, consider the problems of tempo. The composer, in playing the musical phrase, must adhere to strict rhythmic patterns. Expression in terms of tempo changes would not be practical during input. (Output would allow for any desired expression.)

Music symbols such as slurs, ties, phrase markings, beamed notes, dynamic markings, tempo markings and so on would not occur in the score from keyboard input alone, but would be placed in the score via the CRT display. (The CRT display will be described in Section 2.3.)

The implementation of a keyboard input can best be understood by considering

an organ-type keyboard where each note is essentially represented by an on-off switch. Input from the keyboard during one time-slice could be accomplished by setting a bit corresponding to that note. For example, a sixty-one-note manual would require a sixty-one-bit field for each time slice during the input of a music phrase from the keyboard. It should be noted that this allows for input of complex chord structures. Both Bell Telephone Laboratories in New Jersey [8] and the National Research Council of Canada in Ottawa [2] have implemented keyboard inputs and they are working successfully. Sufficient sophistication has been achieved so as to require only insignificant editing of mistakes, (i.e. mistakes occurring during the input interface). It is evident that a keyboard input unit is a powerful device and a primary component of the system being described.

The desire of the music research analyst for high speed input of music scores has not been completely realized. There are two possibilities: (i) input by an optical scanner; (ii) input of sound by magnetic tape.

(6) The optical scanner is the more promising of the above two possibilities. Considerable work has been done with optical scanners, particularly in the area of recognition of alphanumeric characters. Speeds of 14,000-20,000 characters per second are common; rates approaching 50,000 characters per second are possible [10].

Optical scanners for reading music scores have not been developed to any degree of sophistication because of lack of demand. Only in recent years have music researchers begun to use the computer, making optical scanners for reading music a desirable device.

An optical scanner for reading music symbols seems entirely feasible at present, since it would require less sophistication than an optical scanner for alphanumeric characters. The set of symbols is smaller, difference in symbols is greater, and fewer fonts are used in printed music.

(7) Input of sound recorded on magnetic tape is considerably more difficult. Not only

would the overtone structure of notes cause problems, but each instrument would have to be recorded on a separate track. In its present state this type of input would not enhance the system proposed. Conceivable future technological advances, however, may make magnetic tape a reliable music input medium.

(8) The system would also include software packages for music research and computer composition. The capabilities of these packages have been explained in Chapter I.

(9) A sound production unit would be an integral part of the system. It would resemble the sound production unit in the Groove system described in Chapter I.

(10) A plotting device (like a Calcomp plotter) would provide the user with an "elegant" hard copy of the music score he has created or manipulated. This facility would save the user and music copyist many hours of uncreative labour.



Figure 2.2 A sample music plot

The only prerequisite is that the music score must be resident in the computer. This may be accomplished with the aid of the piano-type keyboard, CRT display and/or optical scanner.

An example of the production possible by a plotter is shown in Figure 2.2. Figure 2.2 does not demonstrate all music symbols and structures which a music score demands. However, additional software would make possible the plotting of any music symbol or structure.

Plotting time is also of interest. At present, plotter pens move at two to four inches per second, comparatively much faster than hand copying of music scores. Figure 2.2 required eight minutes to plot of which four minutes was required to place the music symbols on the staves.

A second method of obtaining a hardcopy of a music score involves photographing the score as it appears on a CRT display. There are two advantages of this method. First and foremost, the cost of

equipment for photography is considerably less than the cost of a plotter and drive for the plotter. Secondly, this method would produce a hardcopy substantially faster than a plotter. The disadvantage of the method is that the CRT does not provide adequate resolution for an elegant hardcopy of music symbols.

(11) A CRT display would be the most versatile and powerful device in the system. Because of its unique and extensive capabilities, it will be treated in detail in Section 2.3.

2.3 Graphical Display Unit (GDU)

The purpose of the display unit in the system is threefold.

(i) It acts as a control unit whereby the user may load the system with the software and data he desires.

(ii) It provides a visual representation of the music score. Software is available

allowing the user to edit and modify the music score.

(iii) It allows the user to initiate music transformations and to specify arrangements of submelodies. The results appear on the screen.

2.3.1 GDU as a Control Unit

As a control unit the GDU is very dependent upon the system implemented (i.e. the facilities included). An obvious duty would be to query the user by presenting him with different menus of information on the screen. Once the control program has acquired the desired information, control would be transferred to the appropriate program for the task specified by the user.

A GDU is ideal as a control unit for several reasons. First, it is a quiet medium. The hammering of a type ball or clatter of a card reader is usually not conducive to creative thinking. Secondly, the light pen permits the user to specify his desires with pen picks, which is much faster and more convenient than typing from a keyboard. Lastly, users generally find a graphic

representation less alienating and impersonal than an alphanumeric terminal.

2.3.2 Visual Representation of a Music Score

The GDU provides the musician with the medium with which he is most familiar. It is a powerful representation of the score because of its dynamic nature. From the GDU any staff in the score may be recalled to the screen in a matter of a few seconds; response time is a factor dependent upon the number of symbols on the staff. The command to recall a staff consists of primarily two parts. First, the user picks the proper keyword in the menu (e.g. the word RECALL). He then specifies the staff desired by referring to its unique number in the score. (The method of referencing a staff can vary with the implementation.)

The GDU also provides the user with the capability to move back and forth throughout the score editing and modifying it. The ability to edit and modify the score is a necessary part of the system for two reasons. First, the composer requires the ability to change the phrases he has previously composed. Secondly, errors and deficiencies will sometimes occur during input from the piano-type keyboard and optical scanner. These will need correction from the user at the GDU.

The command language for editing and modifying is very simple. The user may erase a succession of individual symbols or an entire staff of symbols. This is accomplished by picking a keyword (like ERASE) and then the symbols or staff of symbols to be erased. Again the response time is fast, considerably faster than the traditional pencil and eraser method.

Once symbols have been erased the user may leave those locations blank or replace the erased symbols. He may place a symbol by picking the symbol prototype (from the menu provided) and the desired location(s) for the symbol. Locations for symbols need not be limited to locations from which symbols were erased. The symbols may be positioned at any location on any staff on the screen.

During the above process of editing, modifying and composing, the representation of the score on the screen never becomes cluttered or messy. The score is always neat with consistent symbol shapes and sizes and consistent spacing of symbols. As a result the user is never delayed and hampered by a disorganized visual representation of his score.

2.3.3 Initiation of Transformations

In addition to the short commands surveyed above, the system would provide the user with commands specifying powerful transformations. Generally, the commands would consist of four or more components. In order to preserve the power of the command the user requires a quick and convenient way of specifying the components of the command. The GDU, along with the light pen, provides this facility. Section 2.4 considers the transformation commands in detail.

2.4 Music Transformations and Arrangements

As mentioned previously, the primary advantage of the total system lies not in its ability to edit and modify, achieve a hardcopy and produce a sound production (though these are very necessary and useful), but in its ability to allow initiation of music transformations and arrangements. This is true because of the computer's ability to provide such a significant time saving when manipulating numbers and symbols. Therefore it is important that the essence of the transformations and arrangements be understood. In the discussion of each of the transformations and arrangements, it will be assumed that a suitable code has been developed for an internal representation of a music score. The proposed data structure is discussed in Section 2.6.2.

2.4.1 Arranging Capability

The arranging capability, specified by a suitable command, is primarily an organizational task. Once the user has decided upon an order of submelodies he would like combined, he may use the light pen to indicate the submelodies desired and the new position at which they are to be placed. The order of the submelodies would be indicated by their position in the sequence of pen picks. The principal advantage to the user is the time saved. It is true that the desired sequence could be entered via an input device. However, this would require considerably more time and would inconvenience the user. Also errors could occur during input.

It should be pointed out that the sound production system will allow the user to hear different sequences of submelodies even when they are not sequential in the music score. Therefore, it is likely that the arrange command would be used only after the user has decided upon the desired sequence of submelodies.

2.4.2 Transposition

In an installation, transposition would be the most widely used of the transformations indicated. It would mean a saving of time, and in some cases money, for any individual with a knowledge of music.

Transposition has a different meaning for the computing scientist than for the musician. The computing scientist is not interested in the musical significance of transposition. Rather he is interested in the mathematical and algorithmic definition of the change which constitutes the transposition. A detailed and formal definition of an algorithm for transposition is of little value apart from its contribution to decisions about the internal representation of the music score. However, an algorithm could be easily derived with the aid of the following characteristics of transposition. The user will specify the following information in the message received by the program: the interval to be transposed (say COL); and the beginning and end of the sequence to be transposed. It should be noted that the user need not be aware of how the computer accomplishes the task of transposition (as defined in the following discussion).

First, not all the music symbols on the staff are affected by the transpose command. Only key signatures and

notes (and accidentals where necessary) are changed. Secondly, the change in key signatures can be determined by the following method which utilizes the arrays in Figures 2.3 to 2.5. From the position of the key signature in the data structure (DS), the number of flats or sharps in the key signature will be evident. If the key signature contains N flats, refer to the Nth position in Figure 2.4 to determine the tonic of the old key signature, say X. Determine the row in Figure 2.3 which represents X. Reference the X row and COL column of Figure 2.3 for the tonic of the new key signature, say Y. From Y one can determine whether the new key signature is in flats, sharps or naturals (key of C major or A minor). Then scan Figure 2.4 for the occurrence of Y. The position occupied by Y indicates the number of flats in the new key signature. This information replaces the old key signature representation in the DS. The same procedure is used for key signatures in sharps, except that Figure 2.5 replaces Figure 2.4. The keys of C major and A minor are treated as special cases.

Thirdly, consider the change which occurs to notes. The change in notes can be explained in the following manner. Consider the music staff with an overlay of horizontal and vertical lines creating cartesian coordinates for the symbols. Then the change in any note can be viewed as an increment or decrement of the Y value (pitch) in conjunction with refinements for special situations

(accidentals). If the note to be transposed does not have any accidentals, the transposed note is determined by incrementing or decrementing by the interval specified by the user. More calculation is required for situations where accidentals affect the pitch of the note. Any accidental placed on a note is in effect until a subsequent accidental or bar line nullifies its effect. Transposition of notes in the latter case requires the following method.

Let X be the note to be transposed. Reference the X row and COL column of Figure 2.3 for the new note Y . Now three possibilities arise. First consider the case where Y has no sharps or flats. Check the key signature and current accidentals in the measure to determine whether a natural has to be added to nullify the effect of the key signature or previous accidentals in the measure. Place the new note Y in the DS along with the natural sign (if required). Secondly consider the case where Y has one sharp or one flat. Then note the combined effect of the key signature and accidentals in the measure to determine whether a flat or sharp must be placed on the note. For the third case of Y having two flats or two sharps, the key signature need not be considered. Both flats are placed on the note in the case where two flats are not currently in effect on the note in that measure. A similar result holds for two sharps.

	↑	d2	M2	M2	A2	d3	M3	M3	A3	d4	P4	A4	d5	P5	A5	d6	M6	M6	A6	d7	M7	M7	A7	d8	P8	A8		
A ^{bb}	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} C ^{bb} C ^b	C	C ^b	D ^{bb} D ^b	D ^b	E ^{bb} E ^b	E ^{bb} F ^{bb} F ^b	F	F ^b	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A	A ^b	A ^{bb} A ^b	A	A ^b	A ^{bb} A ^b	A	A ^b	A ^{bb} A ^b	A		
A ^b	A ^b B ^{bb} B ^b	B	B [#]	C ^{bb} C ^b	C	C [#]	D ^{bb} D ^b	D	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	G [#]	A ^{bb} A ^b	A	A [#]	A ^{bb} A ^b	A	A [#]	A ^{bb} A ^b	A	A [#]	A ^{bb} A ^b	A	
A	B ^{bb} B ^b	B	B [#]	C ^b	C	C [#]	C ^x	D ^b	D	D [#]	E ^b	E	E [#]	F ^b	F	F [#]	F ^x	G ^b	G	G [#]	G ^x	A ^b	A	A [#]	A ^x	A [#]	A ^x	A [#]
A [#]	B ^b	B	B [#]	C	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F ^x	G [#]	G [#]	G [#]	G ^x	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
A ^x	B ^b	B [#]	B ^x	C ^x	C [#]	C ^x	D [#]	D [#]	D [#]	D [#]	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F ^x	G [#]	G [#]	G [#]	G ^x	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
A ^{bb}	B ^{bb} C ^{bb} C ^b	C	C ^b	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F ^b	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A	A ^b	B ^{bb} B ^b	B	B ^b	B ^{bb} B ^b	B	B ^b	B ^{bb} B ^b	B	B ^b	B ^{bb} B ^b	
B ^b	C ^{bb} C ^b	C	C [#]	D ^{bb} D ^b	D	D [#]	E ^{bb} E ^b	E	F ^b	F [#]	G ^{bb} G ^b	G	G [#]	A ^{bb} A ^b	A	A [#]	B ^{bb} B ^b	B	B [#]	B ^{bb} B ^b	B	B [#]	B ^{bb} B ^b	B	B [#]	B ^{bb} B ^b	B	
B	C ^b	C	C [#]	C ^x	D ^b	D	D [#]	D [#]	D [#]	D [#]	E ^b	E	E [#]	F ^b	F [#]	F [#]	F ^x	G ^b	G	G [#]	G ^x	A ^b	A	A [#]	A ^x	A [#]	A ^x	A [#]
B [#]	C	C [#]	C [#]	D [#]	D	D [#]	D [#]	D [#]	D [#]	D [#]	E [#]	E	E [#]	E [#]	F [#]	F [#]	F [#]	F ^x	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
B ^x	C [#]	C ^x	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F ^x	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
C ^{bb}	C ^{bb} C ^b	D ^{bb} D ^b	D ^{bb} D ^b	E ^{bb} E ^b	E ^{bb} F ^{bb} F ^b	F	F ^b	G ^{bb} G ^b	G ^{bb} G ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
C ^b	C ^b	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A	A ^b	B ^{bb} B ^b	B	B ^b	C ^{bb} C ^b	C	C ^b	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	
C	D ^{bb} D ^b	D	D [#]	E ^{bb} E ^b	E	E [#]	F ^b	F [#]	G ^b	G [#]	A ^b	A [#]	A [#]	B ^b	B [#]	B [#]	C ^b	C [#]	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	G ^{bb} G ^b	G	
C [#]	D ^b	D [#]	D [#]	D [#]	E ^b	E [#]	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
C ^x	D	D [#]	D [#]	D [#]	E [#]	E [#]	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	
D ^{bb}	D ^{bb} D ^b	E ^{bb} E ^b	E ^{bb} F ^{bb} F ^b	F	F ^b	G ^{bb} G ^b	G ^{bb} G ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
D ^b	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	A ^{bb} A ^b	A	A ^b	B ^{bb} B ^b	B	B ^b	C ^{bb} C ^b	C	C ^b	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	
D	E ^{bb} E ^b	E	E [#]	F ^b	F [#]	F [#]	G ^b	G [#]	A ^b	A [#]	A [#]	B ^b	B [#]	B [#]	C ^b	C [#]	C [#]	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	
D [#]	E ^b	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	
D ^x	E	E [#]	E [#]	E [#]	F [#]	F [#]	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	B [#]	
E ^{bb}	E ^{bb} F ^{bb} F ^b	F	F ^b	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
E ^b	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	G [#]	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
E	F ^b	F [#]	F [#]	G ^b	G [#]	G [#]	A ^b	A [#]	A [#]	A [#]	B ^b	B [#]	B [#]	C ^b	C [#]	C [#]	C [#]	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	
E [#]	F ^b	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	C [#]	C [#]	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	
E ^x	F [#]	F [#]	F [#]	G [#]	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	C [#]	C [#]	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	
F ^{bb}	F ^{bb} F ^b	G ^{bb} G ^b	G ^{bb} G ^b	A ^{bb} A ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
F ^b	F ^b	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
F	G ^{bb} G ^b	G	G [#]	A ^{bb} A ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
F [#]	G ^b	G [#]	G [#]	A ^b	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B ^b	B [#]	B [#]	C ^b	C [#]	C [#]	C [#]	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	
F ^x	G	G [#]	G [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	C [#]	C [#]	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	
G ^{bb}	G ^{bb} G ^b	A ^{bb} A ^b	A ^{bb} A ^b	A ^{bb} A ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	B ^{bb} B ^b	
G ^b	G ^b	A ^{bb} A ^b	A	A ^b	B ^{bb} B ^b	B	B ^b	C ^{bb} C ^b	C	D ^{bb} D ^b	D	D ^b	E ^{bb} E ^b	E	F ^{bb} F ^b	F	F [#]	G ^{bb} G ^b	G	G ^b	A ^{bb} A ^b	A	A ^b	A ^{bb} A ^b	A	A ^b	A ^{bb} A ^b	A
G	A ^{bb} A ^b	A	A [#]	B ^{bb} B ^b	B	B [#]	C ^b	C	C [#]	D ^b	D	D [#]	E ^{bb} E ^b	E	E [#]	F ^b	F [#]	F [#]	F ^x	G ^b	G	G [#]	G ^x	A ^b	A	A [#]	A ^x	A [#]
G [#]	A ^b	A [#]	A [#]	A [#]	B ^b	B [#]	B [#]	B [#]	C [#]	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	
G ^x	A	A [#]	A [#]	A [#]	B [#]	B [#]	B [#]	B [#]	C [#]	C [#]	C [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D [#]	D ^{#</}				

↓ A7 M7 M7 D7 A6 M6 M6 D6 A5 P5 D5 A4 P4 D4 A3 M3 M3 D3 A2 M2 M2 D2 A8 P8 D8

Figure 2.3 An array of intervals above and below any pitch

1	2	3	4	5	6	7
F	B	E	A	D	G	C

Figure 2.4 Order of the flat key signatures

1	2	3	4	5	6	7
G	D	A	E	B	F	C

Figure 2.5 Order of the sharp key signatures

2.4.3 Copying

The copy command provides the program with the same information as the transpose command. The copy command reproduces a sequence of notes at a different pitch, retaining the melodic intervallic relationship between the notes. The copy command varies from the transpose command in that key signatures are not changed.

The copy command would be used primarily by the composer. Frequently throughout a composition, phrases will occur which are duplicates of other phrases at the same pitch or at different pitches. The copy command is an attempt to reduce the composing time by eliminating uncreative labour.

2.4.4 Inversion

Composers use several types of imitation in composing. One type of imitation of a theme is inversion. This means that the intervals in the theme are answered in contrary motion. For example, an interval of a minor third up is answered by an interval of a minor third down.

The invert command must have two parts. First, the invert command must reproduce the desired theme at any pitch. Then the reproduced theme must be inverted. To fulfil these requirements, the invert command will first use the copy command to reproduce the theme at the desired pitch, then the task of inverting the theme begins.

Let the first two notes in the theme be X and Y. These notes can be determined from the DS. Then determine the interval between X and Y. If X and Y are the same pitch, no change is required and the next pair of notes may be

considered (i.e. Y and the next note in the sequence). If X and Y are not the same pitch then the interval may be determined using Figure 2.3. Scan the X row of Figure 2.3 for the occurrence of Y. The number of columns to be scanned is never more than four because the pitches of X and Y provide information as to whether the interval is a type of second, third, fourth, etc. The pitches of X and Y will also indicate whether the interval is up or down. This will indicate whether the column names at the top or bottom of Figure 2.3 should be considered. (All ascending intervals are noted at the top of the figure and descending intervals are noted at the bottom of the figure). Once Y has been found, the column in which Y occurs, say COL, indicates the type of interval between X and Y. To determine the note which occurs by an interval in the opposite direction, consult Figure 2.3. Extract the new note, say Z. The problem of determining the accidentals which are to be placed on the note is similar to the problem discussed in transposition. Once the new information is inserted into the DS, the above process may be used on note Y and the next note in the sequence.

It is apparent that the invert command is somewhat slower than the transposition or copy command. This is due to the increased number of references to Figure 2.3. But the increase in response time will be negligible for short phrases or themes.

2.4.5 Retrograde

Retrograde is a type of thematic imitation used by composers. It refers to the process of reversing a theme or passage, (i.e. repeating the passage backwards). Similar to inversion, the composer may desire the retrograde of a passage at any pitch. Therefore the retrograde command involves two parts. First the required passage must be copied to the desired pitch. Then the program must place the new sequence in reverse.

The first requirement is fulfilled by using the copy routine. The second requirement involves working from both ends of the sequence, exchanging the information concerning the two symbols under consideration. (Some care must be taken when considering the termination criteria.) This process is easily accomplished in a linked structure, where an exchange of pointers is all that is required.

The retrograde command is very fast to execute in comparison to the invert command because it requires only one-third the number of references to Figure 2.3.

2.4.6 Augmentation and Diminution

Imitation of a thematic passage at the same or different pitch may occur by augmentation and diminution.

Augmentation refers to increasing the time value for all of the notes and rests in the passage by some suitable factor such as doubling. Correspondingly, diminution indicates a decrease in the time value of all the notes and rests in the passage by some suitable factor.

The augmentation or diminution command requires two processes, as did the invert command. The passage under consideration must be copied to the proper pitches, and then the note values must be changed accordingly. Note that special checks will be required to insure that the augmentation or diminution may be carried out using the time factor specified by the user. The change in note value can be easily calculated. The only other requirement is that a running total of the number of beats will be required to determine positions of bar lines.

Often composers use the transformations such as inversion, augmentation, diminution, and retrograde in combination. The user may accomplish this task by first applying one transformation to a passage and then applying a second or third transformation to achieve the desired result. It is conceivable that for convenience an implementation would allow the user to specify any combination of transformations in one command.

2.5 Shortcomings of the GDU

Foremost among the disadvantages of the GDU is the limited portion of the score visible at any one time on the screen. Generally, six to ten staves can be displayed on the screen. The number displayed is dependent upon the CRT involved and the size of each individual staff. This requires the composer to recall the desired staves more often. Because the whole score cannot be displayed at one time, a referencing scheme must be operative. This places extra onus on the composer to remember the staff on which the desired information occurs. Given current hardware facilities, the problem cannot be eliminated. However, certain software and hardware considerations could reduce the degree of the problem. Using additional software, miniature staves could be displayed on a portion of the screen. These would be used only for observation. Because of the reduced size of the staves, larger portions of the score would be made visible to the user. From a hardware standpoint, a large storage tube type CRT could be used for display of the score. It would not be used for editing and modifying because it would lack a light pen. The addition of a storage tube would naturally increase the cost of the system.

Another disadvantage of the GDU is a hardware limitation. As mentioned previously, the CRT required for

the proposed system must have a light pen. Therefore, the CRT cannot be a storage tube, and thus any picture on the screen must be continuously regenerated to produce a flicker free image on the tube phosphor. The regeneration rate is generally thirty to sixty times per second.

The requirement of a flicker free image places an upper bound on the number of symbols which can be displayed on the screen at any one time. The upper bound is dependent upon the model of CRT being used. As a result it is difficult to indicate how many music symbols a CRT will support on the screen without flicker. The problem of flicker can be reduced in at least two ways. In many of the new CRT's, the cycle time of the computer used for refreshing the screen has decreased by a factor of five, making the refresh problem far less severe. Also the addition of a storage tube in the proposed system would eliminate the need of displaying large portions of the score on the refresh type CRT.

2.6 Data Structure

2.6.1 Introduction

From a software point of view, the one element most important to the proposed system is the data structure. A clumsy representation of the data would make the system

useless as an aid to the composer. Consider the demands that the system places on the data structure.

First, the response time is crucial to the user (especially the composer). Editing, modifying, and use of the transformation commands must be sufficiently fast to cause him no delay or frustration. It is common knowledge that the computer completes most tasks many times faster than is humanly possible. However, if the composer performs the desired task, he is occupied; if the computer performs the desired task, the composer is impatiently waiting. The data structure must therefore be designed to facilitate quick and easy access.

Secondly, the data structure must be compact. When the system is being used for music analysis, lengthy music scores will be required in core. Considering that as many as twenty fields are required to define some music symbols, the storage problem becomes significant. Therefore, the data structure must allow for compact storage of music data.

Lastly, the data structure must be flexible. Two types of flexibility are required. As mentioned previously, the user has the capability of erasing and modifying the music score. This will mean that the data structure must allow for insertions and deletions of music symbols and still retain efficient use of storage. The data structure

must also be sufficiently flexible to store different types of music scores and music symbols. The second type of flexibility concerns the general structure of music scores. Several staves may be tied together by a vertical bar on the left hand side. Each staff in the block of staves will contain music symbols for a particular instrument. The staves in the block are to be considered as being parallel rather than sequential. The blocks of staves are to be considered as sequential. This means that corresponding staves in the blocks are sequential. The data structure must provide for this consideration. Flexibility in reference to music symbols is more easily acquired. Different numbers of fields may be required to define the many different types of music symbols which occur. Again the data structure must provide efficient use of storage.

2.6.2 Proposed Data Structure

The proposed data structure is a combination of a sequential vector (PTSS - pointer to symbols) and linked lists. It can be broken down into two levels.

2.6.2.1 Level 1

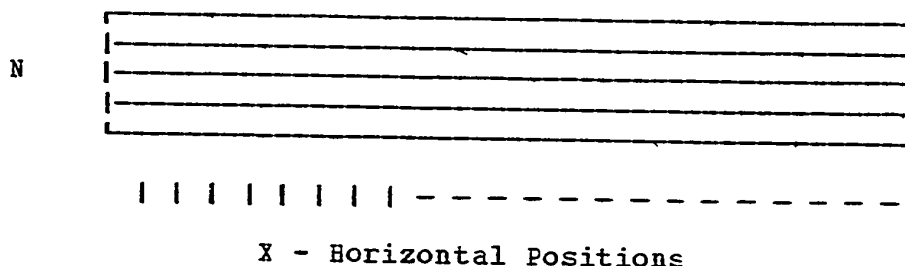


Figure 2.6 A sample music staff

At each of the X horizontal positions on staff N in Figure 2.6, a number of symbols may be placed. Sequential vector PTSS is in direct one-to-one correspondence with the horizontal positions on the staff. Each of the fields of PTSS points to a linked list which contains all symbols occurring at that position on the staff. Each of the staves occupies a section of sequential fields in PTSS. Access to the section of fields pertaining to a particular staff is made possible by a number assigned to each staff (N).

The structure of vector PTSS is complicated by the situation where several staves are tied together by a bar on the left hand side creating a block of staves. As mentioned in Section 2.6.1 the staves in the block of staves are to be considered as being parallel rather than sequential. This situation is accommodated in the following manner.

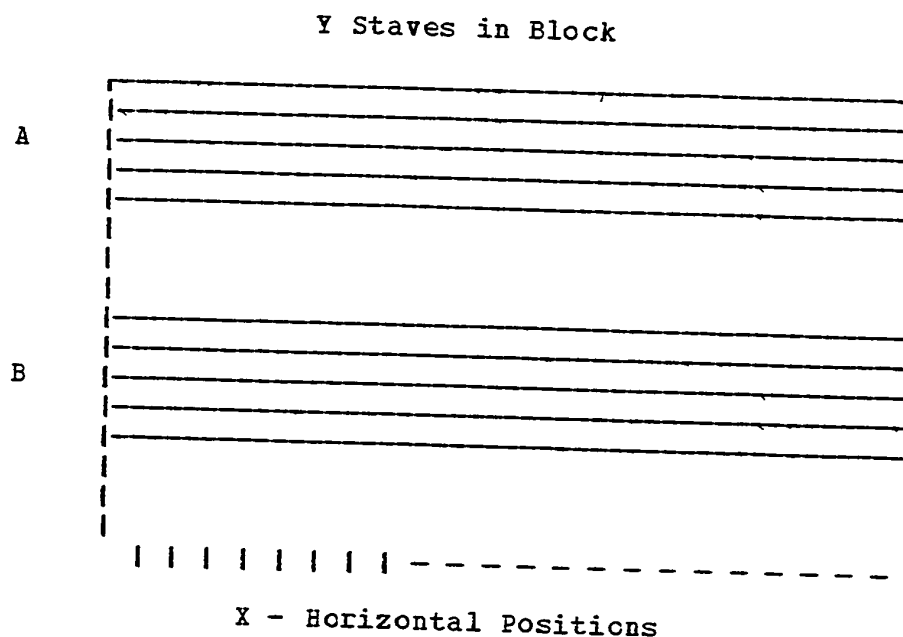


Figure 2.7 Example of a block of staves

Let the number of staves in a block be Y . Then each node in PTSS will have Y fields. Each field points to symbols occurring on a particular staff (in the block) for a particular horizontal position. For example, in Figure 2.7 all nodes in PTSS will have Y fields, of which the i th field will point to symbols on staff A, field $i + 1$ will point to symbols on staff B, and so on (i.e. i goes from 1 to y). Diagrammatically, the structure can be represented by Figure 2.8.

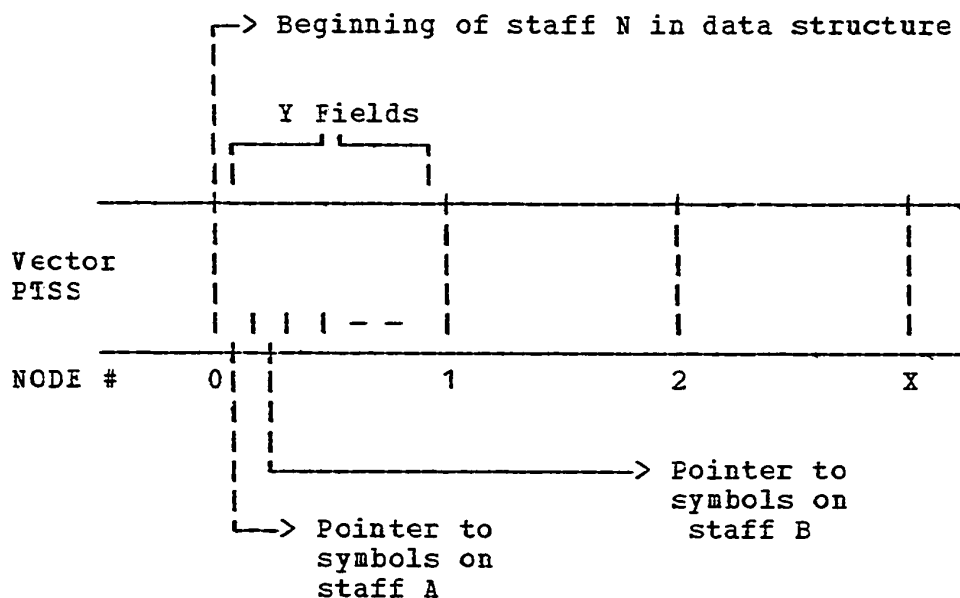


Figure 2.8 Format of vector PTSS

Due to the sequential character of PTSS, any position on any staff is easily referenced. Generally four operations (two multiplications and two additions) are sufficient to reach a desired position. From the representation of Figure 2.8, consider referencing the horizontal position X of staff A in the block of staves numbered N. Each block of staves requires a predefined number of fields in PTSS, say M. Therefore, $M \times N$ will index the starting position in PTSS for the block of staves numbered N. The X-coordinate of the light pen pick (LPP) will indicate the horizontal position in the block of staves, say X. Then $M \times N + X \times Y$ will index the starting

position of all vertical elements occurring at the horizontal position. The Y-coordinate of the LPP will indicate which staff in the block is being referenced (i.e. which of the Y fields is being referenced, say Z.) Therefore, any position on any staff can be referenced by using the formula $MxN + XxY + Z$.

The sequential structure of PTSS provides several advantages over a linked structure. As already mentioned, referencing a desired position is both fast and simple. Secondly, a sequential structure is much simpler to manipulate than a linked structure. Thirdly, insertions into a linked structure would require significantly more time. Although, the time required for referencing the desired position for placement of a symbol would be approximately the same, a linked structure would in addition require setting pointers. Fourthly, the pointers in a linked structure require valuable space. The required storage would approximately double (depending upon the extent of the linking).

The primary advantage of a linked structure would be in relation to vacancies in the staff. In the proposed structure a position on the staff which has no symbols will still have a place for a pointer in the data structure. In a linked structure no position in PTSS would be left blank. However, the overhead in additional information (i.e.

information as to the X position represented by each field in the linked list) and pointers would make a linked structure inefficient in terms of storage and retrieval time. Lastly, in dealing with the transformation commands, it is necessary to deal with one particular staff which is sequential in the blocks of staves. Therefore, it must be possible to move quickly from one element to the next element in the staff under consideration. Note that in the structure of Level 1, a constant (Y in the previous discussion) is all that is required to index the next symbol on the staff.

2.6.2.2 Level 2

Level 2 of the data structure consists of a series of sublists. Each sublist is tied to PTSS by the pointers in PTSS.

As mentioned previously, there are many different types of symbols, and most of these are defined by a different number of fields. The symbols which occupy a horizontal position on the staff are the following: notes, rests, barlines, time signatures, key signatures and clef signs. Other music symbols such as dots, accidentals, beams, slurs, ties, phrase markings and tempo and dynamic markings which are not alphanumeric in character, will be viewed as

attributes of the previous list of symbols. Alphanumeric dynamic and tempo markings occupy a special place in the data structure and will be discussed in a later section. The representation of each of the symbols will now be discussed in detail.

Clef signs, key signatures, bar lines, and time signatures are symbols which will not have attributes. As a result, these symbols require no storage other than that provided in PTSS. The positions (for pointers) which are provided for these symbols in PTSS can be used to store a code representing the symbol itself rather than a pointer to the symbol. Different techniques are available to distinguish data from pointers in PTSS. One technique is to store all data as negative numbers and all pointers as positive numbers.

The requirements of chord structures and rests provide the primary data of Level 2. The information regarding one note or rest is stored in a node. Each node has four fields as shown in Figure 2.9. The type field indicates whether the symbol is a whole, half, quarter, eighth, sixteenth, thirty-second, (etc.) note or rest. The pitch field indicates the Y-coordinate of the symbol relative to the base of the staff on which it occurs. The attributes field of a note node contains information concerning accidentals, staccato or legato marks and dots.

These symbols are noted in the attribute field by special binary codes. Information concerning another class of symbols is also included in the attribute field. Beams, phrases, slurs and tempo and dynamic markings (which are not alphanumerically defined) are coded in the attribute field. These symbols are similarly indicated in the attribute field by special binary codes. For example, a beam joining several notes may be indicated by turning a predefined bit on for the first and last note in the beamed sequence of notes.

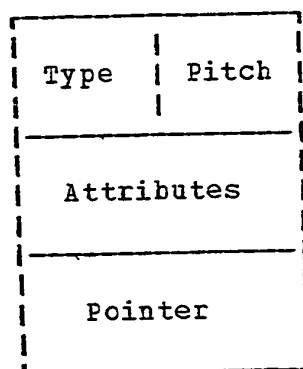


Figure 2.9 Note or rest node

Storage of additional information (i.e. information not indicated in the discussion, for example, sound production information) can be placed in the attribute field.

The attribute field of a node representing a rest is less packed. The only information it contains is the number of dots affecting the time duration of the rest.

The pointer field provides a link to the node defining the next symbol occurring on the staff at the same horizontal position. For example, the representation of a M-note chord on a particular staff has a data structure representation similar to Figure 2.10.

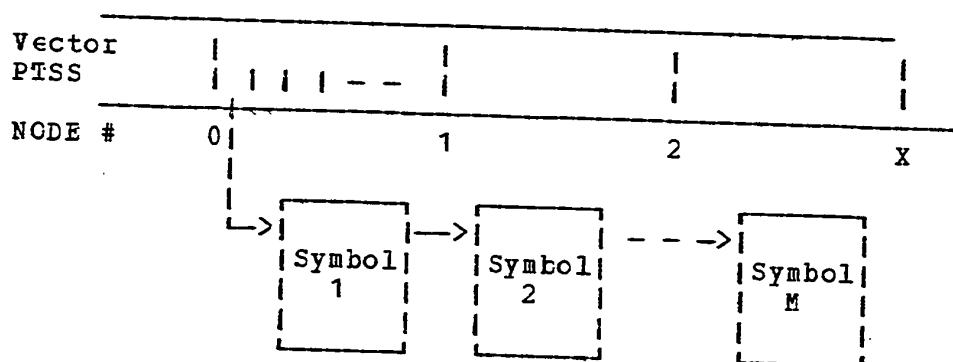


Figure 2.10 Structure of an M-note chord

Alphanumeric tempo and dynamic markings are stored in a second linked structure on Level 2. Each field in Level 1 will be subdivided into two subfields to allow for two pointers. The use of the first subfield has already been indicated. The second subfield will contain a pointer to a

linked list of text,tempo or dynamic markings (MKS), if markings exist on the score starting at that horizontal position. Because markings will not occur at each horizontal position on a staff , the majority of the second subfields will be unused. This inefficient use of storage can be eliminated if the user will tolerate a minor restriction. It is evident from the discussion of Level 2 that the POINTER field of the last node in a linked list of note and rest nodes will not be used. It is conceivable that this PCINTER field could be used to point to markings (MKS) occurring at that horizontal position. If this type of structure is used, the second subfield of each subfield in Level 1 can be eliminated. The user would be restricted to beginning alphanumeric markings on a staff only at horizontal positions where notes or rests occur.

MKS will have a simple four field format (see Figure 2.11). The pointer field points to another node of tempo and dynamic markings starting at the same horizontal position, if another node exists. The length field indicates the number of characters in the alphanumeric string. The pitch field indicates the vertical positioning of the markings relative to the staff. The string field contains the alphanumeric tempo or dynamic markings. Figure 2.12 indicates the relation of MKS to the rest of the data structure.

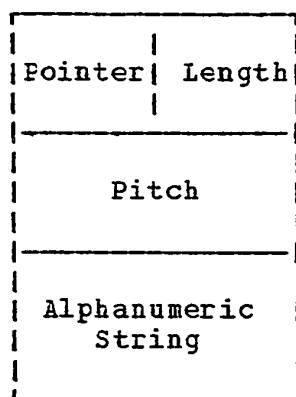


Figure 2.11 Node in MKS

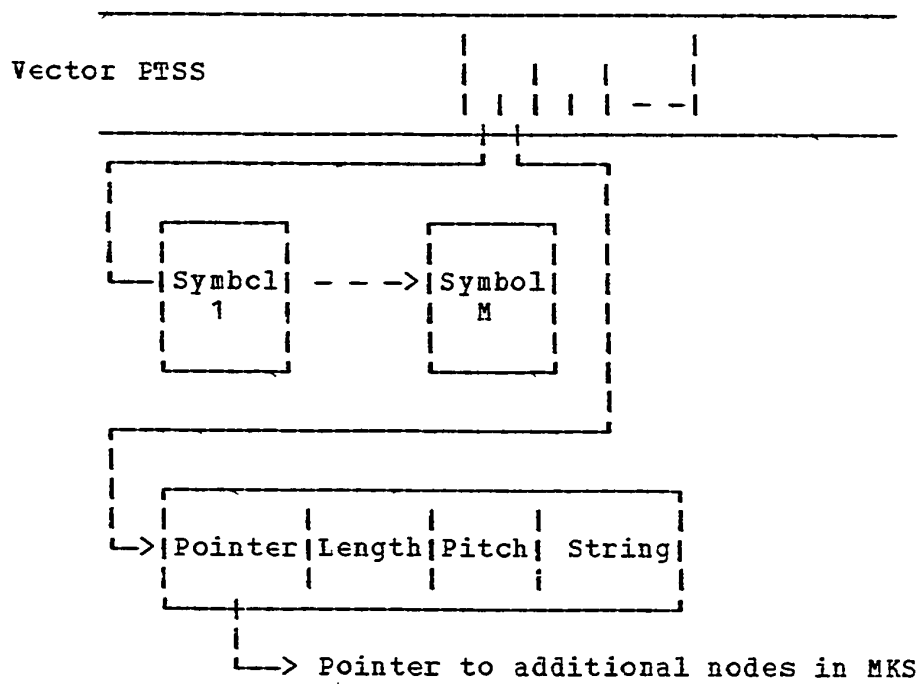


Figure 2.12 MKS in the data structure

The structure of Level 2 of the data structure is very important. Essentially, Level 2 dictates the type of sophistication of symbols which can be represented. The linked list structure is necessary at this level in order to make efficient use of storage and provide the generality required. In addition the linked structure is very easily processed. Each node can be processed by the same code. A sequential list is not practical at Level 2 due to the varying number of symbols which can occur at a horizontal position on a staff.

CHAPTER III

IMPLEMENTATION

3.1 Introduction

The components of a computer-support system for a music department have been dealt with in detail in Chapter II. Most of the components have been developed sufficiently at various installations to operate effectively in the system. However, the capabilities of the GDU have not been fully exploited. Most implementations [2,8] have required the composer to learn either a partially or completely different representation of music data in order to interact via the GDU. As a result, the majority of composers in contact with such systems have been apprehensive. This need not be the case. It is possible to provide the composer with the basics with which he is most familiar, the traditional staff and established music symbols. The implementation at the University of Alberta is intended to show the feasibility of this approach.

The implementation is operational on the GRID-360 display system at the University of Alberta [5]. The implementation uses the Fortran language in conjunction with a package of subroutines used to communicate with the GRID

[5]. At present, the user must load the application program at the beginning of each session. The user interacts with the system by using a light pen (LP) and various buttons on the keyboard (Figure 3.1). The LP is a light sensing device. On the LP is a switch via which the user indicates his desires. When the LP is held close to light on the screen and the switch on the LP is depressed (i.e. a light-pen-pick - LPP), a signal or interrupt is sent to the supervisor in GRID core. From this interrupt, the supervisor is able to determine the coordinates of the LPP and thus determine the desired object. The GRID supervisor acknowledges its find by placing a flashing arrow on the object.

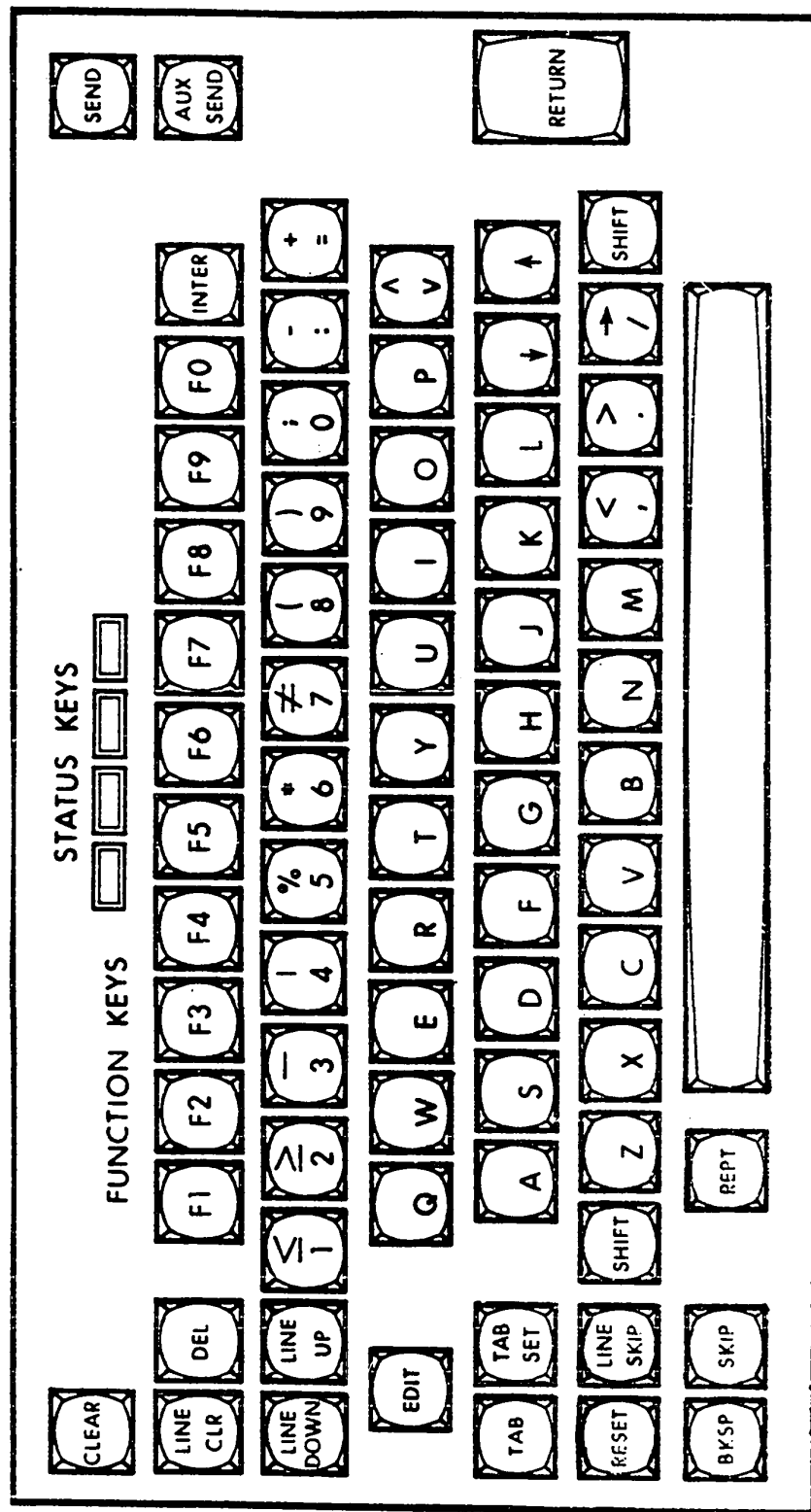
Once the composer has indicated the desired score with a IPP, this one component command can be sent to the application program in the 360 by depressing the SEND key on the alphanumeric keyboard.

Initially, the program queries the composer as to the desired type of score by displaying a menu of possibilities on the screen similar to Figure 3.2. There are various restrictions on each of the types of music scores. The single staves type of score(type I) initializes the program such that each block of staves contains only one staff. Therefore during a terminal session, a user request for a new staff will result in one new staff being displayed. On a staff the user may place any symbols

provided in the menu. Note that only one voice can be displayed on any one staff (i.e. no chord structures are allowed). Figure 3.3 illustrates a type I score.

The piano staves type score (type II) initializes the program such that each block of staves contains two staves. Therefore a request by the user for a new staff will display two staves joined on the left hand side by a vertical bar (i.e. a typical piano staff). This type of score is more general in that it provides for a chord structure of one or two voices. Any symbols from the menu may be placed on a piano staff. Figure 3.4 illustrates a type II staff.

The third type of score (solo staff with accompaniment staff) provides the user with a combination of a type I score and a type II score. A request for a new staff by the user will display a block of three staves joined by a vertical bar on the left hand side. The bottom two staves in a block are joined by an additional vertical bar signifying correspondence to a type II score. Figure 3.5 illustrates a type III score.



ALPHANUMERIC/FUNCTION KEYBOARD

Figure 3.1

STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()

BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF

MARKINGS/TEXT
BEAM NOTES
BAR LINE
DOT

REQUEST A TYPE OF MUSIC SCORE
WITH A LIGHT PEN PICK
SINGLE STAVES
PIANO STAVES
SOLO STAFF WITH ACCOMPANIMENT STAFF

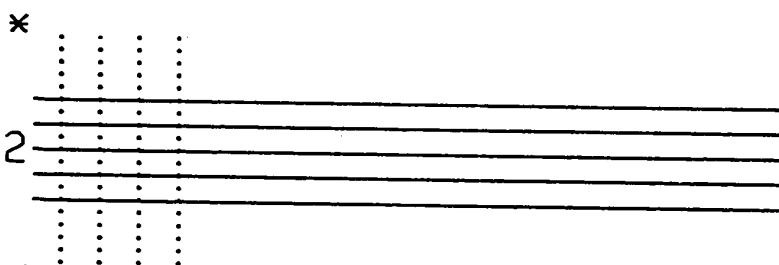
RETROGRADE
INVERT
TRANSPOSE
REPRODUCE
COPY
FINISHED
RESTART
HARDCOPY

Figure 3.2 Initial display on screen

STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()



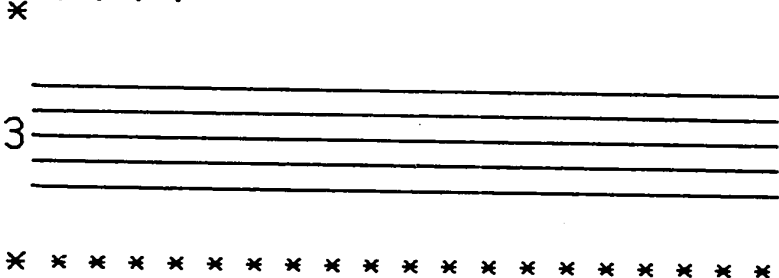
BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF



MARKINGS/TEXT
BEAM NOTES
BAR LINE

DOT

b
4 7
7



RETROGRADE
INVERT
TRANSPOSE
REPRODUCE
COPY
FINISHED
RESTART
HARDCOPY

*

*

Figure 3.3 A type I score

STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()

BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF

MARKINGS/TEXT

BEAM NOTES

BAR LINE

DOT		
#		
b		
4	7	
	7	

RETROGRADE

INVERT

TRANPOSE

REPRODUCE

COPY

FINISHED

RESTART

HARDCOPY









The figure displays a musical score interface with three staves. Staff 1 is populated with musical notation, including notes, rests, and a bar line. Staff 2 and 3 are currently empty. To the left of the staves is a vertical menu of commands: STAFF, POINTS, DELETE, CLEAR, ERASE, SAVE STAFF, RECALL STAFF (), BASS CLEF, TENOR CLEF, ALTO CLEF, TREBLE CLEF, MARKINGS/TEXT, BEAM NOTES, BAR LINE, and a series of commands (DOT, #, b, 4, 7, RETROGRADE, INVERT, TRANPOSE, REPRODUCE, COPY, FINISHED, RESTART, HARDCOPY) each accompanied by a small icon. To the right of the staves, there is a grid of dots and asterisks, which appears to be a visual representation of the musical data or a control interface for the score.

Figure 3.4 A type II score

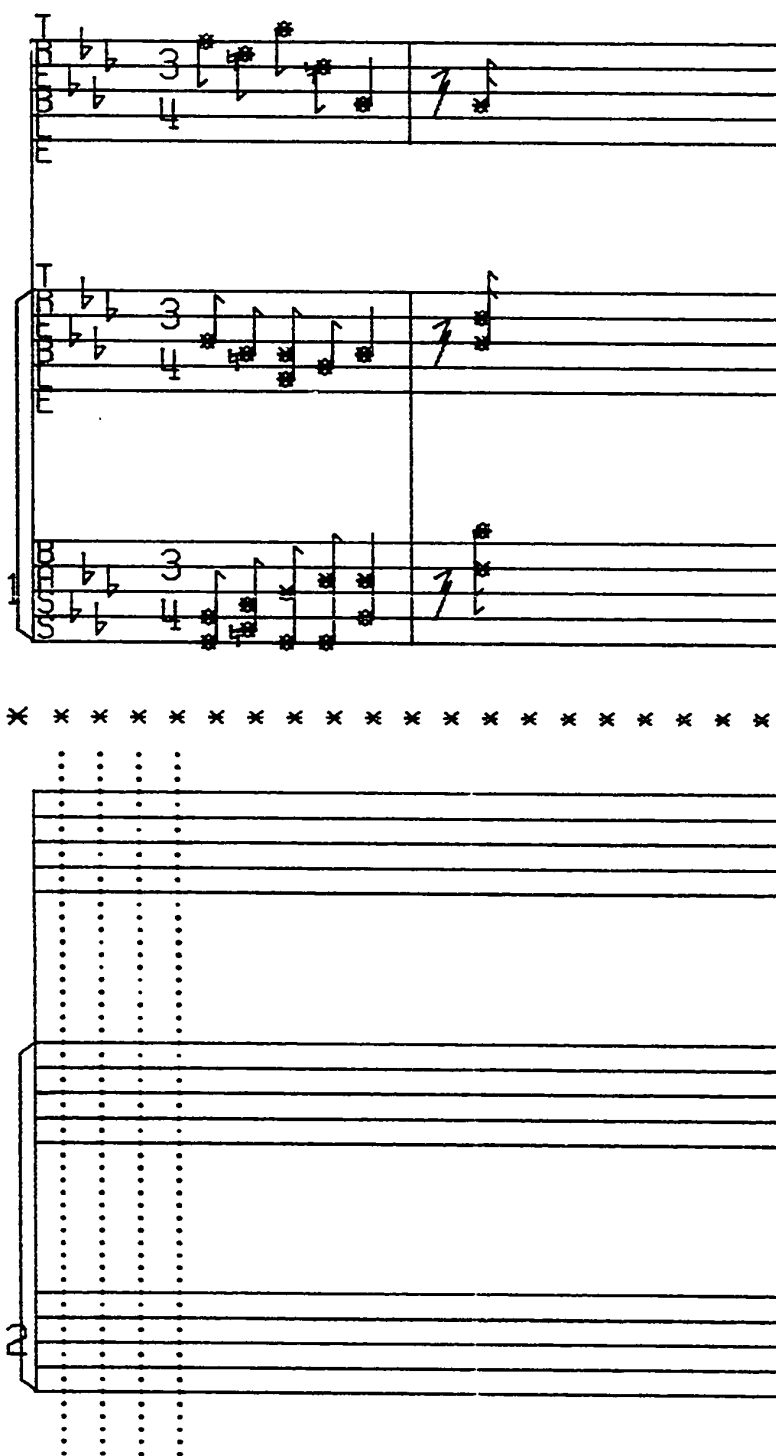
STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()

BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF

MARKINGS/TEXT
BEAM NOTES
BAR LINE

DOT  
 
b  
h 7 
7 

RETROGRADE
INVERT
TRANSPOSE
REPRODUCE
COPY
FINISHED
RESTART
HARDCOPY



The diagram illustrates a musical score layout. It features three staves with musical notation. The first staff is labeled '1' and contains a treble clef, a key signature of one flat, and a 3/4 time signature. The second staff is labeled '2' and contains a bass clef. The third staff is labeled '3' and contains a treble clef. The notation includes notes, rests, and a bar line. Below the staves, there are two rows of dots and a row of asterisks. The diagram is labeled with '1' and '2' at the bottom left.

Figure 3.5 A type III score

Following the initialization according to the staff type, the composer may begin his composing in earnest. The program provides the user with a variety of commands, which permit editing and modifying, composing and specification of transformations.

In subsequent discussions, the word staff will refer to a block of one or more staves, unless otherwise specified.

3.2 Commands

Commands consist of one or more components. The components are LFP's or depressions of buttons on the keyboard. Once a message of one or more commands has been constructed, it can be sent to the application program in the 360 by depressing the SEND key (Figure 3.1).

Two basic types of commands are possible. First a number of commands can be constructed by picking command words or keywords which occur on the left hand side of the screen or by depressing buttons on the keyboard which have special meaning. The second type of command involves prototypes of symbols. The prototypes are also found on the left hand side of the screen. Symbols resembling the prototypes can be placed on staves on the screen. However,

in order to pick the desired point for a symbol (say a note), the LP must be able to sense light at the desired position. This is made possible by displaying a block of dots or points on a staff as shown in Figure 3.5. These dots can be moved around the screen by various commands to be described later in the section.

STAFF - The composer may obtain a staff on the screen by picking the keyword **STAFF**. He may further indicate the position of the staff on the screen by picking one of the asterisks which occur in a vertical column to the right of the keywords. If he does not indicate a position for the staff, the program will find the first vacant position on the screen, proceeding down from the top of the screen. Each staff is displayed with a number (SN) on its left which is unique to that staff.

POINTS - The block of dots referred to earlier may be displayed on a staff by picking the keyword **POINTS**, picking the number of the staff on which the points are to be displayed, and picking one of the asterisks which occur horizontally on the middle of the screen. The last LPP indicates the horizontal position of the beginning of the block of dots.

Once the block of dots is on the screen, they may be moved horizontally on the staff by picking one of the

horizontal asterisks on the middle of the screen. In addition, for convenience, a depression of the SEND key will move the block of dots ahead one horizontal position (provided no commands are currently in the message).

Music Symbols - Music symbols such as notes, rests, and accidentals can be placed on a staff by picking one of the prototypes occurring midway up the screen on the left hand side. Positions are then selected for the symbol by picking point(s) in the block of points displayed on the screen. In the case of accidentals or dots, the second component of the command is a LPP of a note or rest already displayed on a staff.

DELETE - A LPP of the keyword DELETE and of a SN will erase the staff and the symbols on the staff from the screen and from the data structure (DS) in computer memory.

CLEAR - The clear command differs from the delete command in that the staff remains on the screen. The symbols on the staff are erased from the screen and the DS.

SAVE STAFF - The save command differs from the delete command in that the symbols on the staff are not erased from the DS when the staff disappears from the screen.

RECALL STAFF - The RECALL command permits the user to recall any staff in the score and to display the staff at any vacant position on the screen. The command consists of picking the keyword RECALL and indicating the number of the staff to be recalled by using the keys F1 - F9 (Figure 3.1). The position for the staff may be indicated by a LPP of one of the vertical asterisks on the screen. If no position is specified, the program places the recalled staff at the first vacant position proceeding up from the bottom of the screen.

Clef Signs - A clef sign may be placed on a staff by picking one of the keywords TREBLE CLEF, BASS CLEF, ALTO CLEF or TENOR CLEF, followed by a pick of the staff (in the block of staves) on which the clef sign is to be placed. This is done by picking one of the points displayed in the block of points.

Time Signatures - Time signatures can be placed on a staff by the numbered F-keys (Figure 3.1) for the two numbers in the time signature. The two numbers are separated by a virgule symbol from the keyboard. The time signature is positioned on a staff by picking one of the points in the blocks of points.

Key Signatures - Key signatures can be placed on a staff by (1) depressing an F-key to indicate the number of flats or sharps , (2) picking the prototype of a flat or sharp on the left hand side of the screen, and (3) picking one of the points in the block of points to indicate both the staff and horizontal position for the key signature.

DOT - A dot may be placed beside any note or rest on a staff by a LPP of the keyword DOT and a LPP of the note or rest beside which the dot is to be placed.

BAR LINE - A bar line can be placed on a staff by picking the keyword BAR LINE and a point in the block of points on the screen.

ERASE - The erase command is the primary editing command. It consists of two or more components. The first component consists of a LPP of the keyword ERASE. The other components are LPP's of the symbols to be erased.

BEAM NOTES - The keywords BEAM NOTES provide the user with the power to beam several notes together. The command consists of a LPP of the keywords and two additional LPP's indicating the beginning and end of the sequence of notes to

be beamed. A double beam will result if the keywords are picked twice.

MARKINGS/TEXT - The key words MARKINGS/TEXT provide the user with the power to place alphanumeric strings anywhere on a staff, provided a note occurs at the same horizontal position as the starting position of the string. The command consists of a LPP of the keywords, an alphanumeric string from the keyboard (Figure 3.1), and a LPP indicating the vertical positioning of the string. The last LPP need not be one of the points in the block of points.

TRANSPOSE - The transpose command provides the user with the power to transpose any sequence of symbols from one staff (in a block of staves) to the same staff or another staff (in a block of staves) and into any key. If the two staves are incompatible (i.e. the first staff has chord structures and the second staff does not), the transpose command will copy the lower pitches of the chord structures occurring on the first staff. Note that the sequence of symbols must begin with a note. The command has four components. The first component is a LPP of the keyword TRANSPOSE. The second and third components are LPP's indicating the beginning and end of the sequence to be transposed. The fourth component is a LPP of one of the points in the block of points which can occur on any staff.

COPY - The copy command differs from the transpose command in that the sequence is copied rather than transposed to the new position. The restrictions mentioned for the transpose command also hold for the copy command. The components of the copy command differ from the transpose command only in that the keyword **COPY** is picked rather than the keyword **TRANSPCSE**.

REPRODUCE - The reproduce command differs from the copy command in that the pitch of any note is not changed in the copying of the sequence of symbols. The restrictions mentioned for the transpose command also hold for the reproduce command. The components of the reproduce command differ from the transpose command only in that the keyword **REPRODUCE** is picked rather than the keyword **TRANSPCSE**.

RETROGRADE - The retrograde command differs from the copy command in that the sequence under consideration is placed in retrograde at the new position, rather than copied to the new position. The retrograde command has four components. The first component is a LPP of the keyword **RETROGRADE**. The second and third components are LPP's indicating the beginning and end of the sequence to be placed in retrograde. The fourth component is a LPP indicating the pitch and starting position for the sequence of symbols in retrograde .

INVERT - The invert command allows the user to invert any sequence of music symbols occurring on a staff (in a block of staves) and place the inverted sequence on any staff (in a block of staves) at any specified pitch. If the sequence to be inverted contains chord structures, only the lower pitches of the chord structures are considered. Note that the sequence of symbols must begin with a note. The components of the invert command differ from the components of the copy command only in that the keyword picked is INVERT rather than COPY. Figures 3.6 and 3.7 show the results of using the transformation commands. Consider staff number one as the original music phrase. The succeeding staves show the results of using the specified transformation command on staff number one.











RESTART - During a terminal session, a user may become exasperated with his accomplishments thus far in the terminal session. He may again start from the beginning by picking the keyword RESTART. The program reinitializes as required and provides the user with the equivalent of a new terminal session.

FINISHED - The finish command will terminate the terminal session. A LPP of the keyword FINISHED will accomplish the task.

STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()

BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF

MARKINGS/TEXT
BEAM NOTES
BAR LINE

DOT		
#		
b		
4		
		

RETROGRADE
INVERT
TRANSPOSE
REPRODUCE
COPY
FINISHED
RESTART
HARDCOPY



STAFF
POINTS
DELETE
CLEAR
ERASE
SAVE STAFF
RECALL STAFF ()

BASS CLEF
TENOR CLEF
ALTO CLEF
TREBLE CLEF

MARKINGS/TEXT
BEAM NOTES
BAR LINE

DOT		
#		
b		
4		
7		

RETROGRADE
INVERT
TRANPOSE
REPRODUCE
COPY
FINISHED
RESTART
HARDCOPY



HARDCOPY - The hardcopy command generates a file of Calcomp commands. The file of commands can be used to produce a hardcopy of the image on the screen. Figures such as 3.6 and 3.7 are the result of using the hardcopy command.

To add even further to the versatility of the commands, any message sent to the 360 may contain one or more commands. The upper bound on the number of commands allowed in one message to the 360 is dependent upon the total number of components. The message may contain up to twenty components. If the user has misconstrued a command, subsequent commands in the same message will be ignored. A minimum amount of error checking has been done in detecting bad syntax and semantics in the command language. There are two reasons for not implementing a full set of diagnostics. First, limited time for programming did not allow for detailed error checking. Secondly, the block file provided by the Multibank Supervisor accommodated too few blocks to allow for the definition of appropriate diagnostic messages.

3.3 Data Structure

3.3.1 Level 1

The first level of the data structure in the present implementation consists of a vector (SYMTYP - SYMBOL TYPE). SYMTYP is similar to the sequential vector PTSS described in Section 2.6.2. Recall that in one of the proposed structures for PTSS, each of the Y fields in a node was subdivided into two subfields. Each subfield consisted of either two pointers or one pointer and data. SYMTYP has the same field and node structure as PTSS. It differs from PTSS in the definition of the subfields. The first subfield indicates the type of symbol occurring on a staff at a particular horizontal position. If the symbol type is a note, the second subfield contains a pointer to level 2 where additional information is stored. If the symbol type is not a note, the second subfield contains any additional information concerning the symbol type indicated in the first subfield.

3.3.2 Level 2

The second subfield of note fields on level 1 points to nodes on level 2 which contain additional information concerning the notes. The nodes are not linked on the second level.

There are two types of nodes on level 2 . The first type of node contains information concerning one note. This type of node is used in conjunction with staves which allow only one note at each horizontal position on a staff. Figure 3.9 indicates the fields in the node.

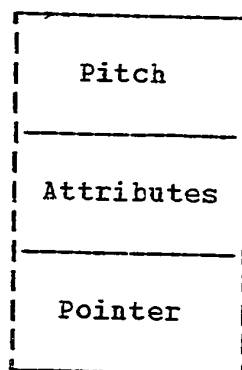


Figure 3.9 A one-note node

Bit Numbers	Information
1,2	unused
3,4	naturals
5,6	flats
7,8	sharps
9-12	unused
13,14	dots
15,16	beams
17-32	unused

Figure 3.10 Format of information in the attribute field

The pitch field is self-explanatory . The attribute field is a packed binary coded field. Figure 3.10 indicates the information represented by each bit. The pointer field points to level 3 . Level 3 contains linked lists of either one or two nodes. These nodes contain alphanumeric tempo and dynamic markings or text.

The second type of node provides information concerning one or two notes . This type of node is used in conjunction with staves which allow a maximum of two notes in a chord. Both notes must be of the same value. Figure 3.11 is an example of a type two node.

Pitch of Note 1	Pitch of Note 2
Attributes of Note 1	Attributes of Note 2
Pointer	

Figure 3.11 A two-note node

3.3.3 Level 3

Level 3 consists of one or two node lists pointed to by the pointer field on level 2. The nodes contain information pertinent to alphanumeric strings (tempo or dynamic markings or text). The storage required for a node is dependent upon the length of the string field. Figure 3.12 is an example of a node on level 3.

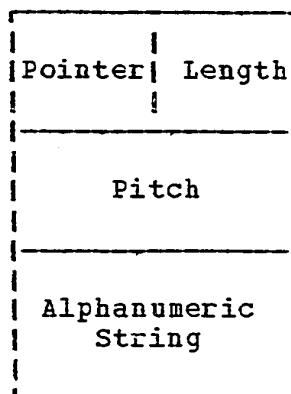


Figure 3.12 Node in MKS

The pointer field points to a second node with fields similar to the node being described. The length field indicates the number of alphanumeric characters in the string field. The pitch field contains the vertical positioning of the string on the staff. The string field contains the alphanumeric characters. This data structure

allows alphanumeric strings to begin only where a note occurs at the same horizontal position (see discussion of this point in Section 2.6.2.2).

CHAPTER IV

Conclusions

4.1 Evaluation of the Proposed System

The proposed system of Chapter II is very comprehensive. It includes both the features desired by music users and the advances made in computer application to the music field. It differs from present computer-music facilities in primarily one way. Whereas present facilities tend to eliminate the traditional composer and his methods, the proposed system places emphasis upon meeting the needs of the traditional composer. In terms of the composer user, the purpose of the proposed computing facility is twofold. First, the facility seeks to eliminate uncreative labour on the part of the composer. This is provided by the music transformations and the storage and retrieval capabilities of the computer. Secondly, the facility acts as an external stimulus to the composer.

The capability of the system to give a sound simulation with the desired orchestration provides the composer with immediate feedback on what he has composed. In addition, the facility has the power to "compose" short

themes according to parameters specified by the user and certain predefined rules. The composer can then hear the composed phrase played through the sound system.

This emphasis on the traditional composer contrasts with facilities available in current computer-aided systems. For example, the Groove system (implemented at Bell Telephone Laboratories [9]) requires the user to learn a different representation of a music score. This may be legitimate when dealing with music analysis due to the mathematical nature of the analysis. When composing, however, the visual characteristics of a music score are essential. It is undesirable to require a user to change his "language" in order to communicate with the system.

The reason for different representations of music data are directly related to software. The Groove system represents music data in terms of time functions. A minimal amount of software is required to manipulate time functions on the GDU. NRC at Ottawa uses a subset of traditional music symbols, a subset too limited to allow for the display of most music scores. In displaying scores in traditional music symbols, the primary decision to be made concerns the set of symbols to be made available to the user. An adequate set of symbols can be derived through minor additions to the subset of symbols used in the implementation described in Chapter III.

4.2 Extent of Implementation

The implementation at the University of Alberta was only part of the proposed system defined in Chapter II. Apart from the interests of the writer, there is a variety of reasons for the extent to which the proposal was implemented. First and foremost, the implementation was a feasibility study intended to investigate the possibilities of manipulating a traditional music score on a GDU. Present computer-aided music installations have not exploited the capabilities of the GDU. Secondly, most of the elements of the proposed system are not directly related to the GDU. Therefore, they would be of little value in a feasibility study of the GDU. Thirdly, it would be impossible to include a piano-type keyboard, sound production unit, or optical scanner because of the cost involved. Lastly, inclusion of all elements of the proposed system would be impossible in the limited time available. The fact that only part of the system has been implemented has not hindered the study of the GDU.

4.3 Direction of Extensions

The order in which extensions could be made to the implementation at the University of Alberta is somewhat dependent upon the interests of the researcher, although some priorities may be suggested.

As viewed by the composer user, the present implementation of the GDU has sufficient capabilities to allow a study of its interaction with other components of the proposed system. For convenience, however, the program should be modified in the following manner. At present the DS representation of a score is lost at the termination of a terminal session. It would be a natural extension to provide storage of music data from one terminal session to the next. This could include a simple cataloging scheme. Also, the program does not include extensive error checking. Additional error checking would avoid the inconvenience of abnormal termination of terminal sessions and/or incorrect results, either of which may now be the consequence of a misconstructured command.

The components of the proposed system which are not available in the University of Alberta implementation can be added as directed by interest and funds. The most natural

extension would be toward a fast input-device, possibly a piano-type keyboard. The piano-type keyboard provides relatively fast input and also acts as an aid to the composer user. The piano-type keyboard may also function as a sound production unit. If the sound production is not satisfactory, a sound production system similar to the Bell Telephone Laboratories implementation (Groove) would be a natural extension.

The software extensions provide some exciting possibilities. Present hardware facilities are sufficient to warrant the writing of software packages for computer composition. Software packages for limited music analysis could also be written. These would be partially limited by slow input of music data, a task which would be simplified by the addition of a piano-type keyboard.

In summary, the present study has (1) given a brief survey of the present state of computer applications in music, (2) proposed in detail a well-balanced computer-aided music facility, (3) implemented a part of the proposed system involving the GDU, and (4) shown the feasibility of using the computer and GDU in aiding the traditional music composer.

REFERENCES

1. Coleman, E., Newsweek, January 4, 1960, p. 53.
2. "Computer Music Facility", Science Dimension, National Research Council of Canada, Vol. 2, No. 3, June 1970, pp. 8-13.
3. Hiller, L.A. Jr., Music Composed With Computer--An Historical Survey, Technical Report No. 18, University of Illinois, School of Music, Experimental Music Studio, 1968.
4. Hiller, L.A. Jr. and L.M. Isaacson, Illiac Suite for String Quartet, New Music Edition, Vol. 30, No. 3, Theodore Presser Company, Bryn Mawr, Pa., 1957.
5. Jackson, W.C., Computer Graphics for the Applications Programmer, Computing Services Reference Manual, University of Alberta, Edmonton, Alberta, 1972.

6. Lefkoff, G., "Computers and the Study of Musical Style", in G. Lefkoff (ed.), Computer Applications in Music, McClain Printing Company, Parsons, West Virginia, 1967, pp. 43-62.
7. Lincoln, H.B. (ed.), The Computer and Music, Cornell University Press, Ithaca, New York, 1970.
8. Matthews, M.V., "The Computer as a Musical Instrument", Computer Decisions, February 1972, pp. 22-25.
9. Matthews, M.V. and F.R. Moore, "Groove - A Program to Compose, Store and Edit Functions of Time", Communications of the ACM, Vol. 13, No. 12, December 1970, pp. 715-721.
10. Rabinow, J., "The Present State of the Art of Reading Machines", in L.N. Kanal (ed.), Pattern Recognition, Thompson Book Company, Washington, D.C., 1968, p. 11.