MUSICOMP, an experimental computer aid for the composition and production of music
Tanner, P. P.

Publisher's version / Version de l'éditeur:
https://doi.org/10.4224/21276346
Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1972-08

National Research Council Canada
Conseil national de recherches Canada

MUSICOMP, AN EXPERIMENTAL COMPUTER AID FOR THE COMPOSITION AND PRODUCTION OF MUSIC

P. P. TANNER

ERB - 869

AUGUST 1972

RADIO AND ELECTRICAL
ENGINEERING DIVISION

DIVISION DE RADOTECHNIQUE
ET DE GENIE ELECTRIQUE

MUSICOMP, AN EXPERIMENTAL COMPUTER AID FOR THE COMPOSITION AND PRODUCTION OF MUSIC

P. P. Tanner

ERB-869

OTTAWA - AUGUST 1972

RADIO AND ELECTRICAL
ENGINEERING DIVISION

DIVISION DE RADIOTECHNIQUE
ET DE GENIE ELECTRIQUE

ABSTRACT

The programming for the National Research Council's Computer Music Facility is described. Included are programs that allow a composer to "write" and "edit" in core memory, programs that "play" music from core memory, and several support programs.

TABLE OF CONTENTS

Page

I. INTRODUCTION 1.1

II. HARDWARE 2.1

III. FORMAT 3.1

IV. OPERATION OF THE MUSIC SYSTEM 4.1

PROGRAM CHOICES 4.1

SOUND GENERATORS 4.7

UTILITY PROGRAMS 4.10

PITCHGEN 4.11

V. SOUND GENERATORS 5.1

MONOPHONIC 5.1

POLYPHONIC 5.8

VI. WRITERS 6.1

DOLY 6.1

PLAPIC 6.10

DRAMU 6.10

SLIDER 6.10

4 WRITE 6.11

VII. SYSTEMS PROGRAMS 7.1

KLOK 7.1

LOOK 7.1

SPDTA 7.1

DELEZE 7.1

GETTER 7.2

DUMPER 7.2

RHYTHM 7.2

ARRAN 7.2

I$DECR 7.2

DAMEL/TRAN 7.3

FORM 7.3

DYNA 7.5

TABLE OF CONTENTS (cont'd)

VIII. OVERLAYS 8.1

COPY 8.1

KEY SIG 8.4

DUMPO 8.4

AUTOSAVE AND -SAVE IT 8.4

PAR SCID 8.6

SLID PAR 8.6

PITCHGEN 8.6

IX. LIMITATIONS OF THE MUSIC SYSTEM 9.1

X. CONCLUSIONS 10.1

REFERENCES 10.2

APPENDIX A - LIST OF MEMORY CONTENTS.................... A.1

APPENDIX B - LIST OF BUTTON COMMANDS.................... B.1

APPENDIX C - PERCUSSION CONTROL.................... C.1

APPENDIX D - CONTENTS OF BITS 9 - 23 OF THE WORDS REPRESENTING NOTES OR RESTS D.1

APPENDIX E - MAXIMUM LIMITS FOR STORING.................... E.1

APPENDIX F - LIST OF SEL 840A INSTRUCTIONS.................... F.1

SELECTED CRT DISPLAY COMMANDS.................... F.2

APPENDIX G - 2ND PLAY G.1

APPENDIX H - KEYBOARD VARIABLES.................... H.1

APPENDIX I - DOLY VARIABLES I.1

APPENDIX J - 4 WRITE VARIABLES.................... J.1

MUSICOMP, AN EXPERIMENTAL COMPUTER AID FOR THE COMPOSITION AND PRODUCTION OF MUSIC

by

P. P. Tanner

I. INTRODUCTION

A musician with creative concepts or ideas has a limited choice of methods for translating his ideas into music. He may compose on his instrument, writing down the notes as he achieves the effects he desires, or he may write his music first on paper and check it later by playing it. The latter possibility, the playing of the music, may limit him to the instruments available to him at the time, or with which he is thoroughly familiar. Making modifications to the music entails scratching out and rewriting, a cumbersome task if much modification is desired.

A group of engineers working on man computer communications at the National Research Council* have implemented a computer music system (Fig. 1) to aid composers to translate their musical ideas into written melodies and actual sound output. This system makes use of a dedicated medium-sized computer (SEL 840A -Fig. 2.) with a CRT display and several peripheral devices.

A composer using the system can write his music using buttons and a cursor, or play his music using an organ keyboard, into the computer's core memory. When the composer is inputting music, he sees it on the CRT screen, as notes on the standard treble and bass staves ranging in duration from 32nd notes to quadruply dotted whole notes, with pitch ranging from 2 octaves below to $2\frac{1}{2}$ octaves above middle C. At any time the composer may press a button which will cause a list of program choices to appear on the CRT screen, from which he can choose a command that will make the computer play the music written so far. Notes and rests may be inserted into, and deleted from, the music. This makes modification of the written score a very simple matter, and an instant playback of the music is possible after any modification.

To test different sound effects, a composer may add commands to his music to effect changes in the timbre, amplitude, or decay level. He may tell the computer the order in which the segments of his music are to be played. In this way a segment that is repeated often need only be written once. The most important feature, however, is that the composer may listen to his music at any time, and at any stage in the composition.

This report is a comprehensive description of the music system. Chapters 2 and 3 are reference chapters describing the hardware and the internal structure of the data sets. Chapter 4 is very similar to a user's manual in that it explains how the various features of the system may be used. Chapters 5 to 8 contain program descriptions for most of the programs listings.

* Data Systems section, Radio & Electrical Engineering Division.

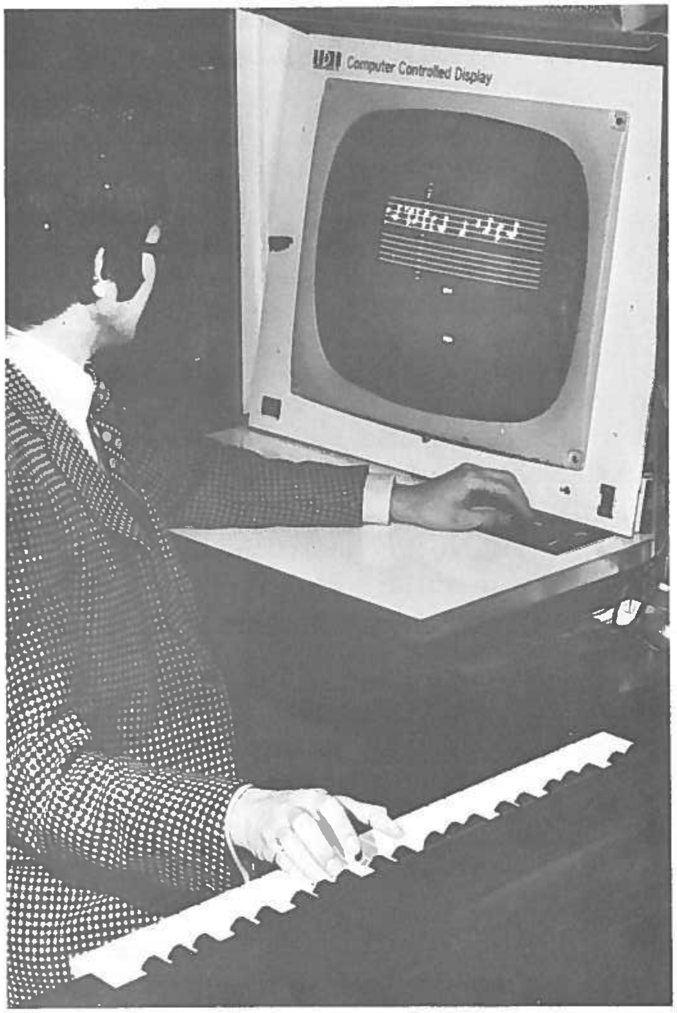


Figure 1. Composer Writing Music into the Computer Using an Organ Keyboard

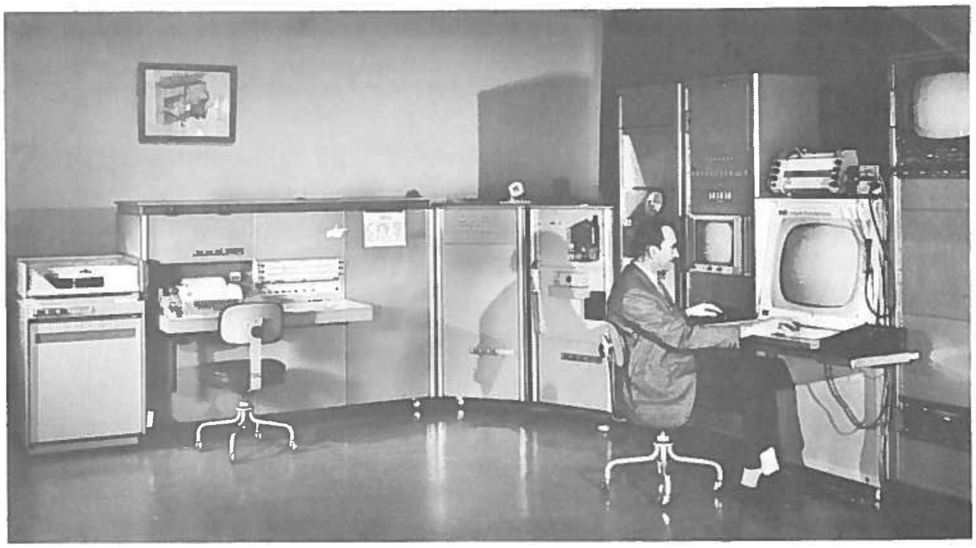


Figure 2. Computer Music Facility at the National Research Council

II. HARDWARE

The NRC music system is implemented on an S. E. L. 840A computer with a 24-bit word, 1. 75-microsecond cycle time, 16K core memory, 4K external memory, 7-track digital tape unit, CRT unit, typewriter, and high-speed paper tape unit.

Extensive use is made of special devices to allow interaction between the musician and the programs. A CRT is used for displaying lists of programs which the user may select, and for writing music and waveform or envelope curves. A pair of horizontally and vertically mounted shaft position encoders allows the musician to select from the lists of programs and to position a cursor on the CRT. Six buttons and two foot pedals control the inputting of notes and commands. An organ keyboard can also be used for inputting notes. A high priority interrupt is caused by a specially placed and coloured button (subsequently referred to as the red button) on the panel. This interrupt stops the current activity and brings back a list of choices to the CRT display for user selection.

The hardware for the output of audible music includes a bank of D/A converters and a bandpass filter. A 4-track tape recorder with a sense tape device allows the individual recording of four tracks, which can then be mixed and rerecorded onto the full-track recorder.

A music synthesizer which is currently still under development will contain voltage controlled oscillators, voltage controlled filters, and voltage controlled amplifiers.

III. FORMAT

At any point in a composition, one melody (broken up into four voices), three waveforms, one envelope, one sound generator, and all the necessary support programs are in the computer memory. All of these except the support programs may be replaced by data of the same type stored on disc; i.e., a melody may be replaced by a melody, a sound generator may be replaced by a sound generator. The only exception permitted is the replacement of waveforms and the envelope by optional support programs. The memory map shows where programs or data are stored in the memory (Appendix A).

MELODY FORMAT

The melody data are stored in the form of a table of amplitude values (from PP to FF) together with a table of decay values (DK1 to DK8), the initial tempo of the piece (120 if none stated), four arrangement tables for the four voices (stating the order in which the bars are to be played), and up to four voices of written music.

This format must be learned to understand the system. The music is a series of 4 voices, each '1000 words long, or 1 voice, '4000 words long.* Each word is one of three things, it can be a note or rest, a bar line, or a command (see Fig. 3).

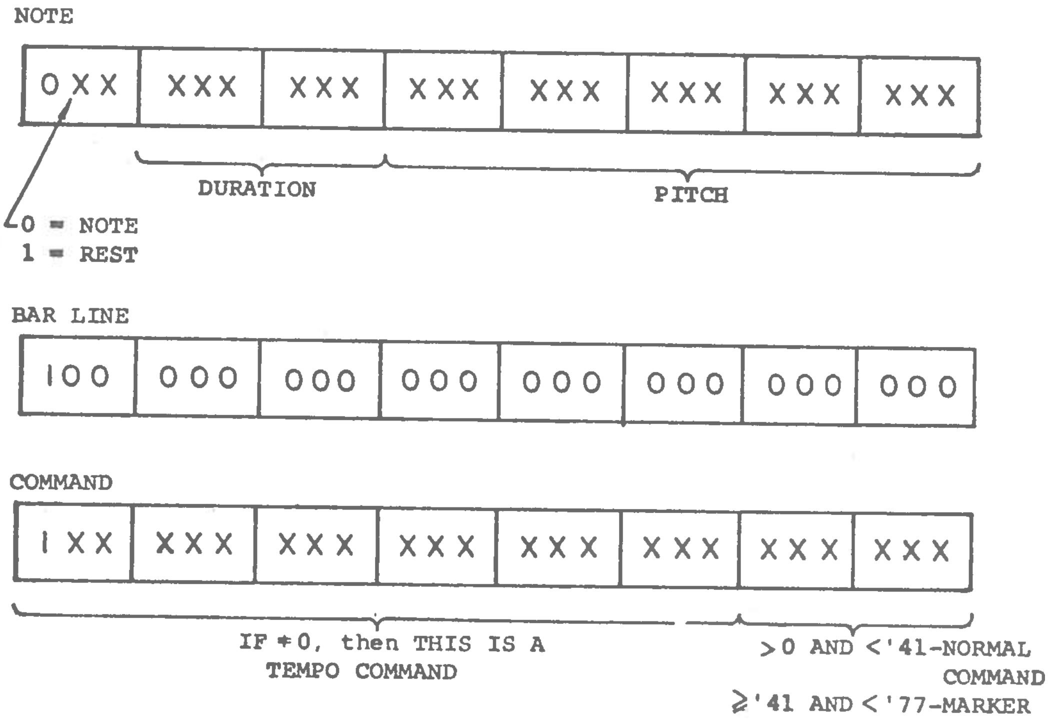


Figure 3. Contents of Words in the Melody

* An apostrophe will be used to denote octal numbers in this paper.

Notes or Rests

Bit 0 = 0
Bit 1 = 0 for a note, = 1 for a rest
Bit 2 unassigned
Bits 3-8 duration in multiples of 32nd notes e.g., 000100 would be an 8th note
001100 would be a dotted quarter.
Bits 9-23 code for the pitch (see Appendix D).

Bar Lines

Bar lines have a 1 in Bit 0, and a 0 in every other bit.

Commands

Commands have a 1 in Bit 0, and the rest of the word identifies the command. If the number in the command (excluding Bit 0) is less than '41 it is a normal command, (see Appendix B); if it is between '41 and '77 inclusive, it represents a letter marker; if it is greater than '77, it represents a tempo. The relationship between tempo T (in quarter notes per minute) and the value N in bits 1-23 is given by the formula $N = 7.5 (10^6)/T$.

ENVELOPE FORMAT

The format of the envelope is quite simple. It is a '5000 word long table, each word containing a number between 0 and '37777777. When the envelope is displayed on the screen, each of the first '2000 words is plotted across the screen with the first 11 bits of the number determining the vertical position on the screen. The '5000 words that represent an envelope can be stored on the disc for later recall. Alternatively, 7 data words which represent the different parameters in an envelope may be stored instead of the whole envelope. This shrinking of the envelope data saves disc space.

WAVEFORM FORMAT

The waveform is '2000 words long, with each word containing a number between - '40000000 and + '37777777. Only the most significant 10 bits are used to determine the vertical position of the waveform plot. The '2000 words that represent a waveform can be stored on disc.

IV. OPERATION OF THE MUSIC SYSTEM

This chapter describes the operation of the music system. To understand how the music system works, and the logic behind the programs, one must understand how a musician uses the system and what he can do with it.

The first step is to load the package of music programs from storage disc (File 16) and choose the program "START" from the "master list of lists". This starts the sound generator in core. As we have just entered the music system, the melody is a series of rests and the output is therefore silent. Then the red button is pressed and this presents on the screen the main list of program choices (Fig. 4), from which all lists can be directly or indirectly accessed.

PLAY + SEE	- listen to the entire melody
PLAYPART	- listen from cursor to the end
QUIT	- exit music system
RECORD	- put melody on magnetic tape
WRITER	- write notes, rests and bar lines on the staff
INSERT	- insert notes, rests and bar lines between others
DELETE	- delete words from the melody
MODIFY	- insert modifying commands into music
GET	- get items from the music library
PUT AWAY	- save items in the music library
ARRANGE	- arrange the order of bars in a melody
TEMPO	- insert a tempo change command
MARKERS	- set marking letters into the music
NEXTPAGE	- more choices on another list
SWITCH?	- play and see or write and check
VOICES?	- monophonic or polyphonic write

Figure 4. Main List of Program Choices

Two writing modes are available to the user - monophonic (one long voice or melody line stored in memory) or polyphonic (up to 4 short voices). In the upper left-hand corner of the screen is a message stating which writing mode the computer is in, and the name of the sound generator in core.

PROGRAM CHOICES

VOICES?

Choosing VOICES? changes the mode from the polyphonic mode to the monophonic mode, or vice versa.

WRITER or WRITE+CH (see SWITCH? below)

Choosing this command causes the computer to display the treble and bass staves and the music as it is written including notes, commands, bar lines, and a controllable cursor. The horizontal wheel (shaft position encoder) moves the cursor line back and forth through the music, and the vertical wheel moves a dot up and down the cursor so that it may be positioned on any line or space on the staves.

To write music, the user positions the cursor at the beginning of the melody, if he is just starting, or after the last note he has written. He positions the dot on the pitch that he wants, and then presses a combination of black buttons, Fig. 5. The duration of the note is determined by the combination of buttons pushed.

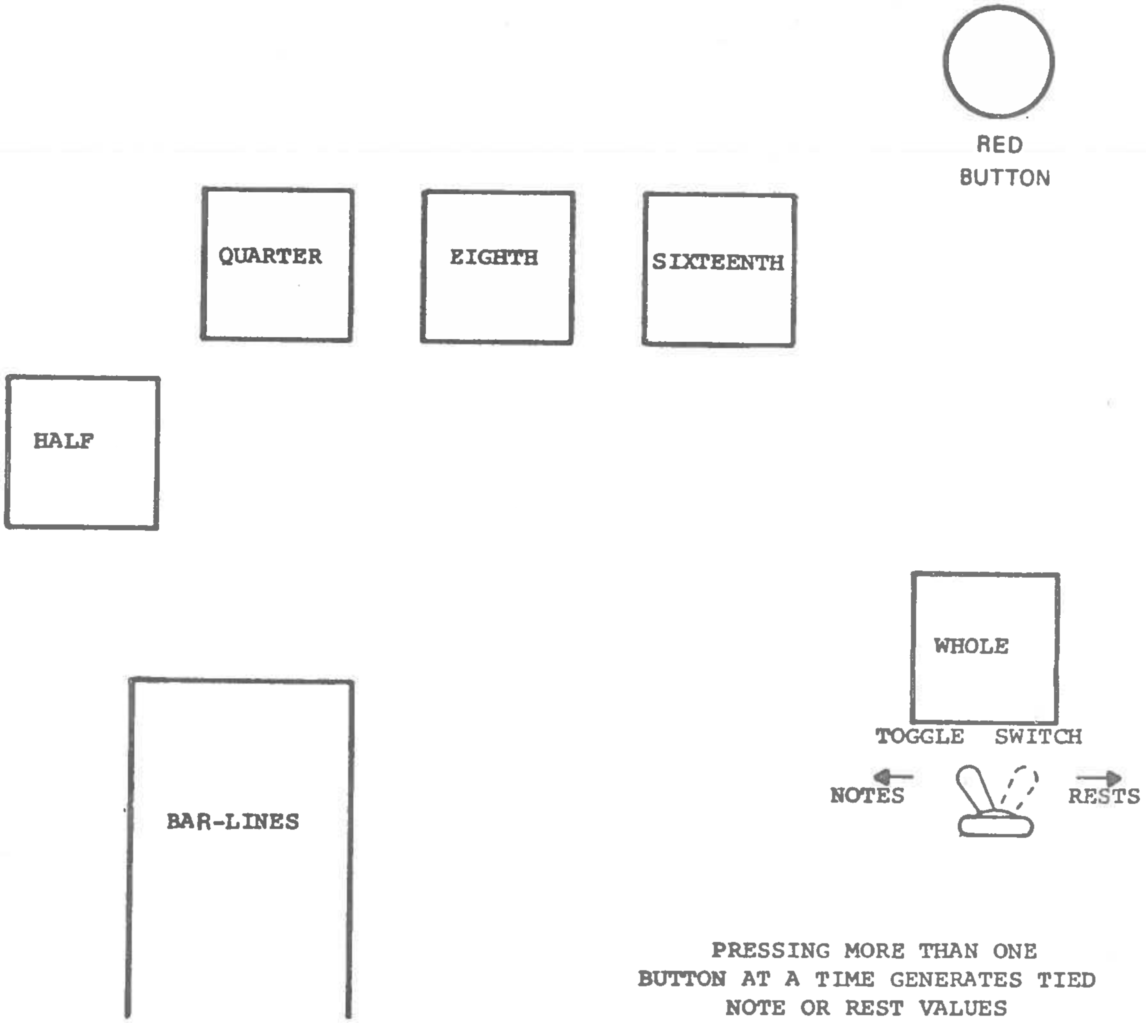

RED BUTTON
QUARTER
EIGHTH
SIXTEENTH
HALF
WHOLE
TOGGLE SWITCH
NOTES
RESTS
BAR-LINES
PRESSING MORE THAN ONE BUTTON AT A TIME GENERATES TIED NOTE OR REST VALUES

Figure 5. Note Values for Black Buttons

The right foot pedal adds a sharp to the note that is being written, and the left foot pedal halves the duration value of the note. In this way any note in the pitch range of 4$\frac{1}{2}$ octaves and any duration from a 32nd note to a quadruply dotted whole note may be chosen.

Bar lines and rests may also be written. A rest may be written in the same manner as a note with the toggle switch in the alternate position. Pressing the palm button together with other buttons has the same effect as switching the toggle switch. Pressing the palm button alone writes a bar line.

Two modes are available for music - polyphonic and monophonic. If in the polyphonic mode, choosing WRITER or WRITE+CH causes the computer to display a list of the four voices. The user chooses one of these voices, say VOICE 2, and the computer displays VOICE 2. As we are now working with VOICE 2, all writing or modifying affects VOICE 2 only, and only VOICE 2 will be played. In the monophonic mode only one voice at a time is available.

SWITCH?

The command SWITCH ? changes WRITER TO WRITE+CH and PLAY+SEE to PLAYER (see below). Choosing SWITCH ? a second time changes both commands back to the original. WRITE+CH displays two numerals in the lower left corner of the screen. These numerals give the duration of the melody up to and including the note under the cursor. The first numeral gives the number of quarter notes and the second is the remainder in 32nd notes, so this second numeral is always less than 8. If the cursor is placed on a bar line, the duration of the entire piece is given.

When switching from one voice to another, the cursor is automatically placed at approximately the same number of 32nd note time intervals from the beginning of the music as it was in the previous voice.

INSERT

This is a different entry to the WRITE program. When a note is written into the music with INSERT, the note under the cursor and all the notes after the cursor are moved towards the end of the music, and the new note is written into the resulting space.

DELETE

This again is an entry to the same program. DELETE moves all the notes after the cursor one word toward the beginning of the music, so the note under the cursor is deleted.

MODIFY

MODIFY is similar to INSERT but commands are inserted instead of notes. These commands can change the decay level, timbre, percussion, glissando, or the envelope control. The command chosen is inserted by a combination of buttons and depends, as well, on the sound generator in use. Different sound generators accept different commands depending on their capabilities. Charts of available commands for the different sound generators are in Appendix B.

TEMPO

TEMPO is used to insert a "change tempo" command in the music. The user must first position the cursor over the point in the melody where he wants the tempo to change. Pushing any button activates the typewriter so that a new tempo may be typed in. If a number, say 85, is typed in, it is interpreted as 85 quarter notes per minute, if two numbers separated by a slash are typed, say 34/21, the new tempo would be 34 quarter notes in 21 seconds. The tempo appears in the written melody as a number, similar in form to the commands.

MARKERS

This command enables the composer to insert letters as markers. After choosing MARKERS he positions the cursor, presses a black button, and types his letter. The letter R caused the playing program to restart the melody. In the polyphonic sound generator an S will restart the whole piece while an R will only restart one voice.

PLAY+SEE or PLAYER (See SWITCH ? above)

When PLAY+SEE is chosen, the computer branches to the sound generator which plays the "in core" melody, and displays the written music at the same time. If PLAYER is chosen, the melody is not displayed. In both cases, the voice that is played is the one that was last written (see WRITER). (However, in a 4-voice polyphonic program, 4 voices may be played.) The monophonic sound generators respond to the last set of buttons pushed as a command, i.e., if the buttons used to select PLAYER were the same as those used to write in DK1 (see the command chart, Appendix B, Table 1), the computer will act as it the command DK1 was the first word in the melody. Each time the program passes a bar line it checks the buttons for a command input. On the polyphonic program, the buttons are used only to determine which of the four voices to display on the screen.

Most sound generators are also capable of controlling an analog percussion generator as the music is playing. If a "percussion on" command is encountered, a monophonic sound generator will use each note and rest to control the percussion. Each note has a specific set of percussion instruments that it activates (see Appendix B). The percussion for a polyphonic sound generator must be written at the end of the 4th line of music and separated from it by a bar line followed by a letter 'D'.

PLAYPART

Choosing PLAYPART causes the computer to play the melody starting with the note under the cursor.

QUIT

This is used to leave the music system.

RECORD

When a piece is to be recorded on audio magnetic tape, the composer can choose RECORD. The computer will wait until it receives an indication that the sense marker on the tape has been detected by the tape recorder, and then it will start playing the music.

GET

GET enables the user to retrieve melodies, waveforms, envelopes (complete tables or parameters), sound generators or utility programs that are stored on the disc. If the composer GETS a data set (a melody, waveform or envelope), the computer transfers the data from the disc to the computer memory and branches to the sound generator program to play the music. If the composer GETS a utility or sound generator program, it is loaded and executed.

PUT AWAY

The opposite of GET, PUT AWAY presents the user with a list from which he can choose what he is going to store on the disc. He may then store an envelope, waveform, or a melody of any number of voices.

Both GET and PUT AWAY access the filing system programs for disc handling. In this system, there is one list stored in core memory at all times. Choosing from this list either loads a data set or another list to replace the present one. Choosing NAME in the PUT AWAY mode activates the typewriter, and after a name is typed, the program stores the appropriate data set on disc and puts the name in the current list. Every list can be accessed directly or indirectly from the "master list of lists", and a choice enabling entry to the master list is at the bottom of every other list (MAST****), so it can always be accessed. (See Data Systems Program Library No. 1085B, D$STOR).

ARRANGE

Bar lines in the computer music system have a different purpose than do bar lines in normally written music. A user may use bar lines to separate phrases if there are some phrases that are repeated. He may then tell the computer in which order he wants his "bars" or phrases played. For example, if he chooses ARRANGE, and then types in 1 2 1 5 3 0 the computer will play the music in the first "bar", then the second "bar", the first "bar" again, the fifth "bar", and finally the third "bar". The computer will then return to the list of choices as the zero at the end indicates the end of the piece.

If the arrangement is longer than one line of the typewriter, a hyphen at the end of the line lets the composer continue on to the next line.

NEXTPAGE

Owing to space limitations, not all the choices can fit on one "page" of the CRT display. Choosing NEXTPAGE presents the user with a second list of choices (Fig. 6).

SEE ENV

If desired, the amplitudes of individual notes may be placed under an envelope. This is useful if one wishes a note to become louder as it is played.

The standard envelope contains 3 exponential curves, one increasing "attack" curve and two decreasing "decay" curves. SEE ENV displays the first '2000 words of the envelope, and enables the user to change the rate of change of any of the curves, or to change their beginning points. Using the "Envelope Generator Commands" in Appendix B, the user picks the parameter he is going to change, then, depressing the left foot pedal, he turns the horizontal wheel. This changes the desired parameter, and generates a new envelope.

REV ENV

This command reverses the first '2000 words of the envelope.

GEN ENV

Seven parameters which define a simple envelope are always stored in core memory. GEN ENV regenerates the envelope from these parameters. The same thing can be done by choosing SEE ENV and pressing the left foot pedal.

TRANSFRM

TRANSFRM computes word by word the average of WAVE 1 and the first '2000 words of the envelope table and stores the result in the envelope table.

RETROGRD

This reverses the last melody that the user was writing.

SET TEMP

This enables the user to reset the initial default tempo that is stored with the melody by typing the new tempo.

COPY and BRANCH

These commands start the execution of utility programs.

SOUND GENERATORS

After choosing GET on the main list of choices, it is possible to replace the present sound generator with another one. The monophonic sound generators are stored on the list *PLAYERS and the sublist *MORE. The polyphonic generators are stored on *POLYFON. The name of the sound generator that is currently in use is always printed in the top left corner of the screen when the main list of choices is being displayed.

Monophonic Sound Generators

Each standard monophonic sound generator uses fixed length waveform tables. Figure 7 gives the names of these sound generators and the number of steps in the waveforms that each produces.

NAME OF STANDARD SOUND GENERATORS	NO. OF STEPS
2ND PLAY, 12THPLAY, 22NDPLAY	8
3RD PLAY, 13THPLAY	12
4TH PLAY, 14THPLAY, 24THPLAY	16
5TH PLAY	5
H1 PLAY	2

Figure 7. Standard Sound Generators

22NDPLAY is similar to 2ND PLAY but provides a rapid decay during the last 32nd interval of a note. This accentuates the articulation (see Appendix B, Table 5).

The following special sound generators only have a choice of three timbres, all based on the contents of the three waveform tables (WAVE 1,WAVE 2, WAVE 3) As the waveform table contains '2000 words, only the first few are used.

1ST PLAY has a feature that enables the user to turn a phase slide off and on. When the phase slide is on, the segment of the waveform table used for the sound generation advances through the waveform table instead of remaining stationary. Thus, the timbre of the note is changed as it is played.

7TH PLAY is similar to 1ST PLAY but provides a sort of portamento that can be turned on and off (Appendix B, Table 4).

6TH PLAY, 8TH PLAY and 10THPLAY phase slide is replaced by a capability of sampling the waveform instead of using a section of it. This sampling can be turned off and on. 8TH PLAY and 10THPLAY produce timbres having the quality of random noise combined with a definite pitch section. The commands to control the noise are the same as those used to control the glissando. In 8TH PLAY, the noise is on when the glissando is off and vice versa. In 10THPLAY both the glissando and noise are on at the same time (Appendix B, Table 3).

Polyphonic Sound Generators

The programs found on the upper half of the POLYFON list will play up to 4 lines of music at once. An extra line may be attached to the end of the fourth voice to control the percussion as noted above. It must be separated from the actual music by a bar line and a letter 'D'.

To change the selection of voices played by the two voice players (PLAIN 2 and TONE 2) it is necessary to depress a foot pedal and the button corresponding to a new voice. (See Appendix B, Table 6). The voices that will be played are the two most recent choices. Some of these programs allow two-channel output, and all allow "up one octave" or "down one octave" commands. (See Appendix B, Table 1).

The properties of the four programs are as follows:

DKAY 4

DKAY 4 plays 1-4 lines of music with exponential decays available on each note.

PLAIN 4

PLAIN 4 plays 1-4 lines of music with no decays. Stereo output is available.

TONE 2

TONE 2 plays 1-2 lines of music with decays, and 13 8-step timbres as in 2ND PLAY. Percussion is not available.

PLAIN 2

This program plays 1-2 lines of music with no decays but allows an $8\frac{1}{2}$ octave range with repeated "up octave" or "down octave" commands. Stereo output is available.

The button command control chart in Appendix B gives all the available commands for each sound generator.

KEYBOARD

This program is a special polyphonic sound generator which allows music to be played from the keyboard and stored in the computer. The sounds produced by this generator are similar to those of PLAIN 4.

Choosing PLAY on the main list of choices (PLAY replaces PLAY+SEE or PLAYER) causes a secondary list of choices to be displayed. This list includes the following choices:

PLAY+SEE, PLAYER, PLAYPART

These commands have their usual effect.

PLAY KEY

This command enables one to play from the keyboard at the same time the computer is playing from memory. The thumb and palm buttons control the number of voices to be played from the keyboard and how many are played from the written music. (See Appendix B, Table 7).

STOR KEY

STOR KEY is the same as PLAY KEY except that all the notes played on the keyboard are written into the melody in the computer memory.

CONTINUE

If this command is chosen, the computer starts playing all 4 lines of music from memory until the user starts playing the keyboard. Then the computer will act as if STOR KEY had been chosen and will start storing the music you are playing into the computer memory. When the red button is pushed after CONTINUE has been chosen, the cursor is moved to the last note that was sounded.

On this program, as in all the polyphonic sound generators, the line of music displayed depends on the button pushed. (See Appendix B, Table 6).

UTILITY PROGRAMS

Utility programs are usually not loaded with the music system and have to be loaded by selecting them from the *PROGRMS list. They are stored in the computer memory in space normally reserved for waveforms or envelopes. (See memory map in Appendix A).

4 WRITE

As the name suggests, this program displays all four lines of music on the screen at once. It has its own list of choices which allow writing, inserting, deleting, modifying, and the adding of tempos and markers. The music is shown in the arranged order; i. e. , if Bar 1 is repeated three times in the arrangement, it will be displayed 3 times in the written music. 4 WRITE has its own list of choices, many of which are the same as those in the main list of choices. The four commands at the top of the list (VOICE 1, VOICE 2, VOICE 3, and VOICE 4) display all 4 voices but display the chosen voice at a brighter intensity, with larger notes and in a different colour. Commands and markers are only displayed for the chosen voice. These commands allow the overwriting of notes as in the WRITER command, but again, only the chosen voice will be written into. INSERT, DELETE, MODIFY, MARKERS, and TEMPO all display four voices, but only change the last chosen voice. LOOK displays all 4 voices or any combination of voices depending on what buttons are pushed (see Appendix A, Table 6). LOOK also allows an automatic advance or back up through the music. If the thumb button is pushed, the cursor automatically advances through the music, if the palm button is pushed, the automatic backward motion is turned on, and if both buttons are pushed all movement stops. If RESTART is chosen the beginning of the music is displayed.

VOICE 1	- see all voices, write in VOICE 1
VOICE 2	- see all voices, write in VOICE 2
VOICE 3	- see all voices, write in VOICE 3
VOICE 4	- see all voices, write in VOICE 4
INSERT	- insert notes in the chosen voice
DELETE	- delete notes from the chosen voice
MODIFY	- insert commands in the chosen voice
MARKERS	- insert letters in the chosen voice
TEMPO	- modify the tempo
LOOK	- see all or any combination of voices
QUIT	- all done (CALL EXIT)
GET	- go to library to get something
RESTART	- display melody at beginning
MONO	- display only one voice at a time (same as WRITER in main List)

Figure 8. List of Choice in 4 WRITE

TUNER

This allows a new set of pitches to be stored for the present sound generator. The user can stipulate the pitch of the lowest note, the number of intervals per octave, and whether he wants equitempered scale, or a tempered scale.

COPY

COPY performs the copying of bars to the end of the same voice or to different voices, the copying of whole or portions of melodies to other voices and the copying of waveforms to envelopes and vice versa. It also contains commands that cause the augmentation or diminution of all the notes in the bar in which the cursor was left, and a command that adds sharps to conform to a specific key signature.

COPY ALL		- copy whole voice
COPY BAR		- copy bar that cursor is in
C TO END		- copy from cursor to double bar line
WAV1>ENV		- copy WAVE 1 into envelope
WAV2>ENV		- copy WAVE 2 into envelope
ENV>WAV2		- copy ENVELOPE into WAVE 2
QUIT	)	- see main list of choices
GET	)	
PUT AWAY	)	
WRITE	)	
AUGM BAR		- double the note durations
DIMN BAR		- halve the note duration
REV BAR		- reverse the notes in the bar
KEY SIG		- put in accidentals to conform to key signature
SAVE IT		- save items in the music library

Figure 9. COPY List of Choices

DRAMU

The program DRAMU enables the computer to print out the music on the digital plotter.

PITCHGEN

PITCHGEN is a combined sound generator and graphical input program. It uses the SLIDER writing program which allows the placing of notes anywhere on the staves, not just on the semitones. (The user can return to the old writing program by choosing CLASSICL). Melodies written with the standard writer routine must be translated before they can be played on PITCHGEN and vice versa. This can be done by choosing PAR>SLID or SLID>PAR on the *PARAMUS list. PITCHGEN provides for graphical input to the melody. The user may "draw a melody" into WAVE 1, WAVE 2, or WAVE 3, or use the smooth curves available in the envelope. Random melodies are also available (see chart in Appendix B, Table 9).

As in KEYBOARD, there is a sublist that includes PLAY+SEE (or PLAYER) and PLAYPART. This list also includes the following choices.

GENERATE

Choosing GENERATE will cause the melody to be created according to the button control commands that are used, for instance, pressing the middle finger and little finger buttons will cause a melody to be written based on WAVE 1, (see Appendix B, Table 9).

GEN PART

Similar to PLAYPART, GEN PART starts the melody generator at the current position of the cursor. Sheets of paper with the staff printed on them are available as a guide to a composer "drawing" a melody into a waveform.

V. SOUND GENERATORS

The following chapters describe in more detail the individual programs with the use of flow charts and listings of the programs. The line numbers referred to are the line numbers in the listings (available from the Data System Section, REED, NRC). The programs are written in the assembler language of the S. E. L. 840A computer, and the list of the mnemonic instructions and their meanings is in Appendix F.

The first programs to be discussed are the sound generators. There are over 20 software sound generators in current use, but they all follow one of two basic patterns, the monophonic or the polyphonic pattern. Within each pattern there are, of course, many variations. 2ND PLAY is used as the basis for the monophonic sound generators.

MONOPHONIC SOUND GENERATORS

2ND PLAY

The purpose of most monophonic sound generators is to output a digital step waveform which has a frequency, amplitude and decay rate controlled by the computer. For example, if we want to output the A above middle C (440 Hz) using an approximate eight-step sine wave, we would have to output a "0" for 1/8 of a cycle (1/8 x 1/440 of a second), then output a 700 for the same length of time, then 1000 then 700 again, then 0, -700, -1000, and back to 0 to start the next cycle. If signals representing these numbers are sent through a D/A convertor the output from the convertor could be amplified, and an approximate sine wave would be heard.

The amplitude is controlled by a number called CAMP (current amplitude). If a decay is desired, a decay amplitude (DAMP) is used. DAMP is a number a little smaller than one. Every cycle of the pitch, CAMP is multiplied by DAMP, and the resultant number is stored in CAMP. This causes CAMP to decay, or become smaller exponentially. The envelope also affects the amplitude if it has been enabled. The final amplitude output then is CAMP, multiplied by the current value of the envelope, and multiplied by the amplitude of the current step in the waveform.

If the user loads an index register with a negative number, the computer has an instruction where the index register will be incremented continuously until the register reaches zero and then the next instruction will be executed. Each increment of the register takes one computer cycle or 1. 75 microseconds. This is the method used for calculating the time intervals between successive steps in the waveforms. There is a table of numbers that is used for loading the first index register. The number loaded depends on the pitch of the note being played.

There are two clock interrupts that are used in the monophonic programs. One of these is controlled by the computer and controls the tempo of the music. It interrupts the program at time intervals equal to a 32nd note and increments a counter ($IFLG) which had been set to the negative of the number of 32nd time intervals in the note. For example, if a dotted half note is being played, $IFLG would be set to -24 as a dotted half is equal in time to 24 32nd notes. When this counter reaches zero it is time to get a new note.

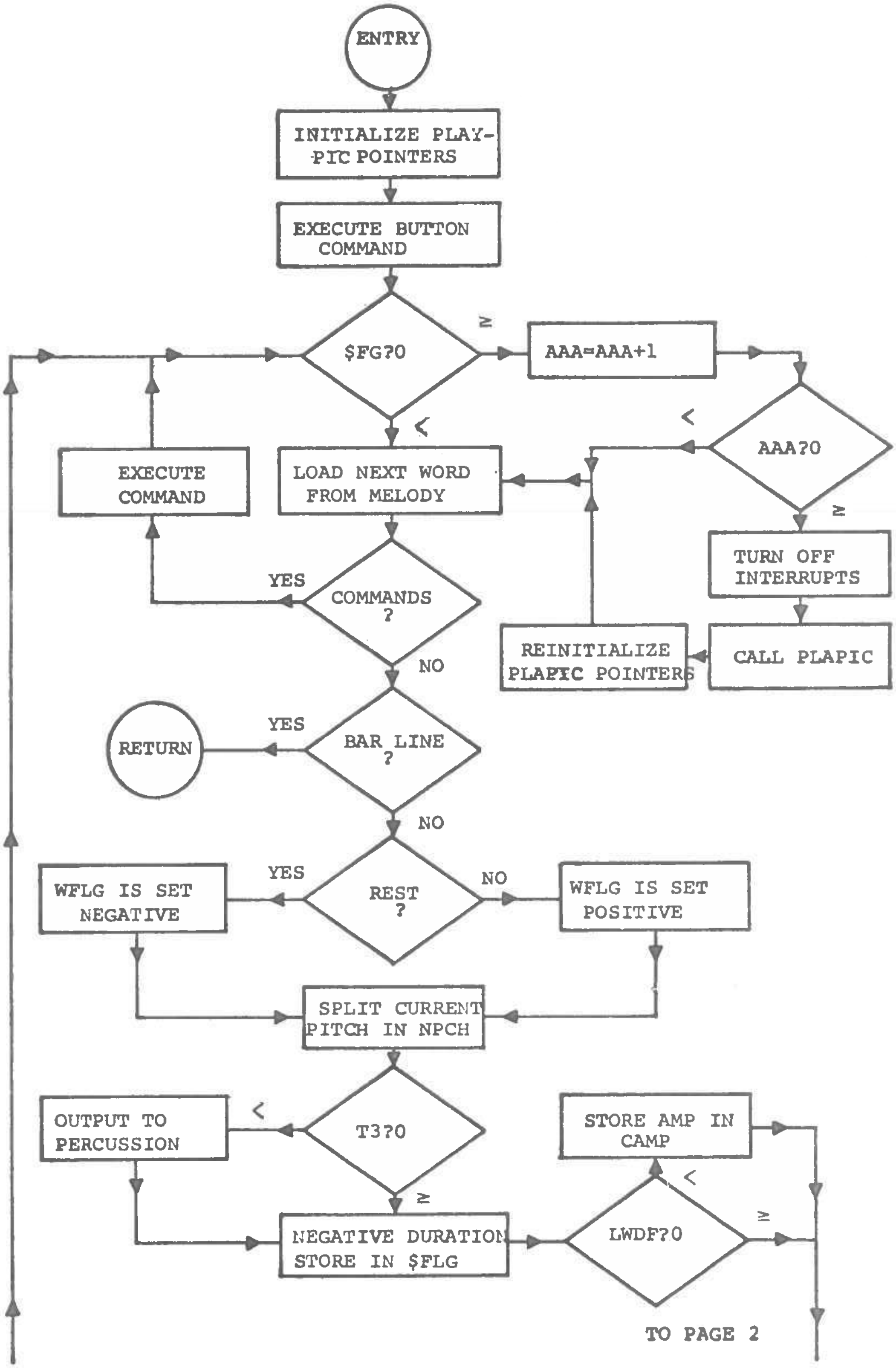


Figure 10. (a) Monophonic Play Subroutine

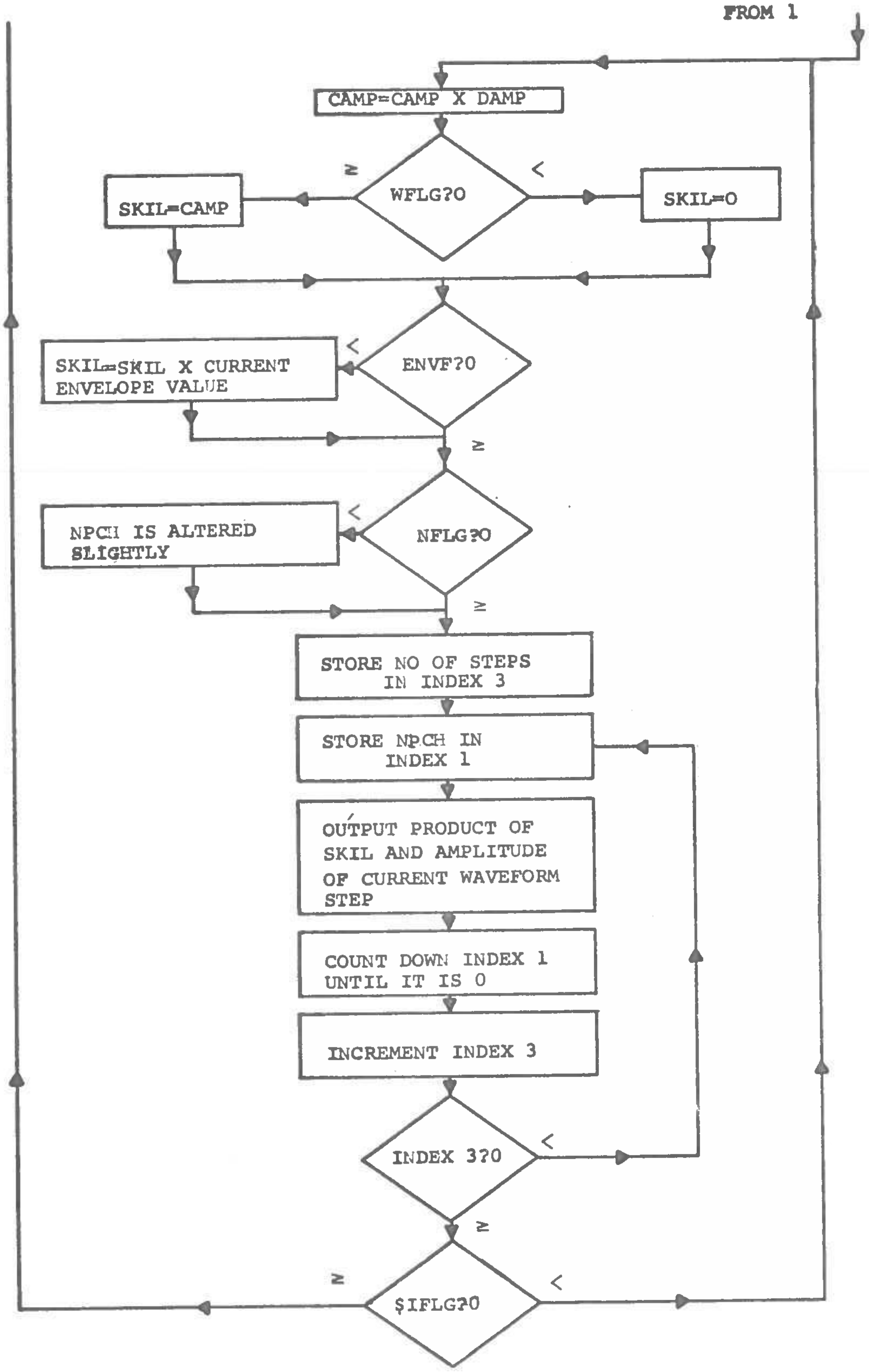

FROM 1
CAMP=CAMP X DAMP
≥
WFLG?0
<
SKIL=CAMP
SKIL=0
<
ENVF?0
SKIL=SKIL X CURRENT ENVELOPE VALUE
≥
<
NFLG?0
NPCH IS ALTERED SLIGHTLY
≥
STORE NO OF STEPS IN INDEX 3
STORE NPCH IN INDEX 1
OUTPUT PRODUCT OF SKIL AND AMPLITUDE OF CURRENT WAVEFORM STEP
COUNT DOWN INDEX 1 UNTIL IT IS 0
INCREMENT INDEX 3
INDEX 3?0
<
≥
≥
$IFLG?0
<

Figure 10. (b) Monophonic Play Subroutine

The frequency of the other hardware clock cannot be controlled by the computer but can be controlled by two knobs to the left of the display. It is used to advance a pointer (index register 2)through the waveform (for glissando) and through the envelope. In the case of the envelope, (if the envelope has been turned on), we load the number that is stored in the memory (if the envelope has been turned on), we load the number that is stored in the memory location, the address of which is calculated by adding the contents of index register 2 and the starting address of the envelope ($ENVT). This number that is loaded is then multiplied by the current amplitude and is then used in place of it. Similarly, if the glissando has been turned on, we load a number that is stored in WAVE 2 whose address is $FORMT1 (beginning address of WAVE 2) plus the contents of index 2. This number is then used to vary the pitch according to how far the waveform value deviates from the midpoint. Therefore every time the clock interrupts the program, we advance one word through the envelope table and the waveform table.

With this background, it is now possible to explain, in detail, the subroutine of the monophonic sound generator that actually plays the music. This subroutine uses different flags to determine which commands are on or off, for example, a minus one is stored in the memory location labelled ENVF when the envelope is turned on, and a 0 is stored there when the envelope is turned off. A complete list of the memory location labels and their uses is in Appendix G.

When the PLAY subroutine is called, the computer first sets up the external subroutine PLAPIC which displays the picture while the music is playing. This setting up is achieved by storing the address of the beginning of the bar as $PLCPIC, and a '20000037 as $PLBPIC. This number in $PLCPIC is increased as we get further from the beginning of the bar. (See PLAPIC, Chapter 6). Next the computer loads AAA with a -1. AAA is incremented before each note is played, and when it reaches zero PLAPIC is called, so the music is rewritten on the screen.

The buttons are then checked and the command they contain is executed. If PLAY+SEE as opposed to PLAYER was chosen, AAA is incremented and if it is now 0, (as it will be the first time through), a new picture is written into external memory to be displayed on the screen. On returning from PLAPIC, the number of notes written on the screen is added to $PLBPIC, and also negated and stored as AAA. This is the number of notes, rests, or commands that can now be played before another call is made to PLAPIC.

The next word in the melody is then loaded. If it is a bar line, we return from the PLAY subroutine. If it is a command, it is executed and the program branches back to incrementing AAA (last paragraph). If the word is a note or a rest, WFLG is set positive in the case of a note and negative in the case of a rest, the pitch is stored in NPCH, a percussion output is made if necessary, the length of the note is stored in $IFLG, and the amplitude and the second index register are reintialized if the slur command is off.

Now we are ready to play the note. The amplitude (CAMP) is multiplied by DAMP, set to zero if we are playing a rest, and stored in SKIL. If necessary, the contents of the present position in the envelope are multiplied by SKIL, and then stored in SKIL. If the glissando is on, the pitch is changed slightly before it is loaded into

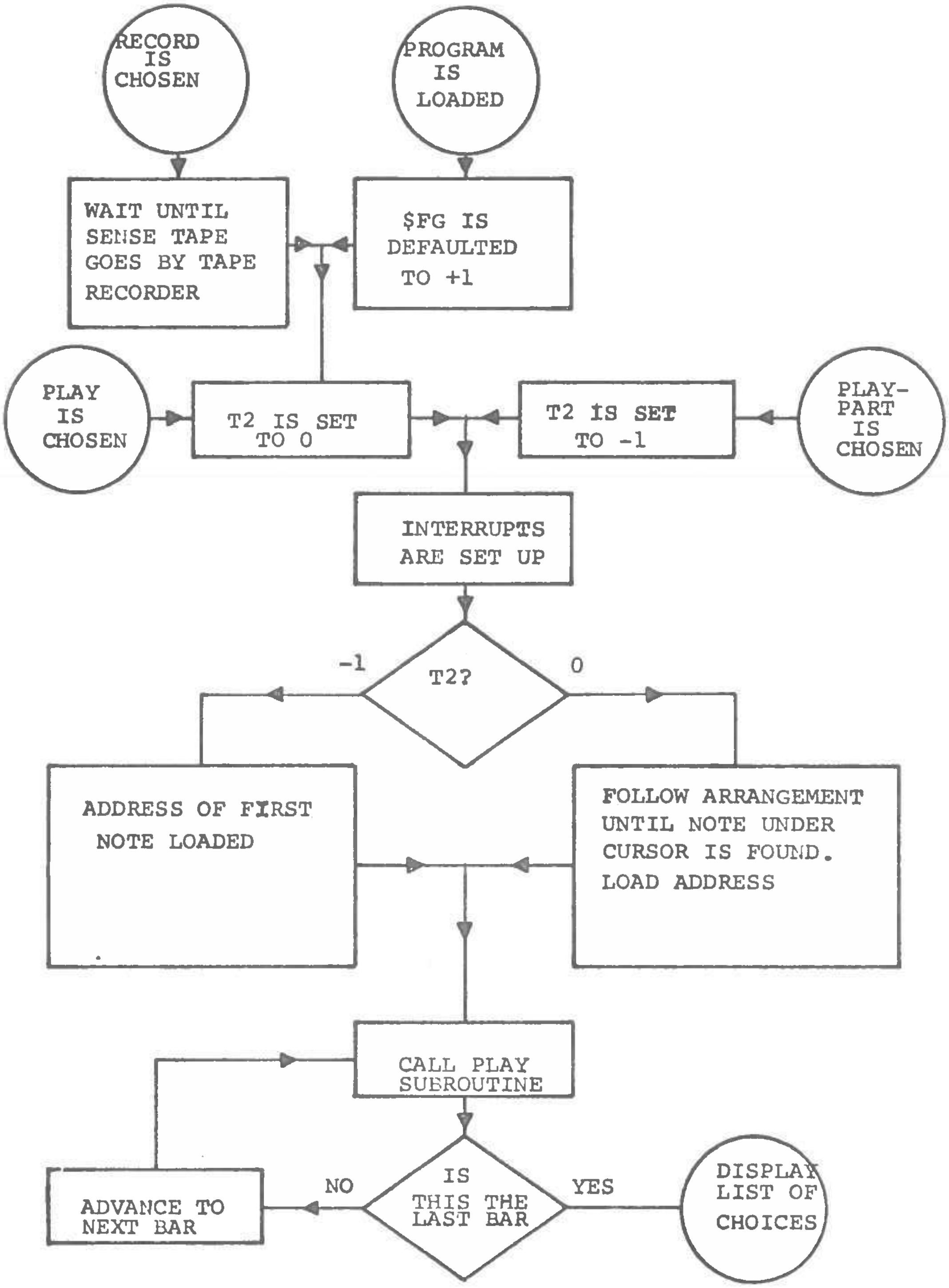


Figure 11. Monophonic Sound Generator

the first index register. Index register 3 is set to the number of steps in the waveform (negated). The computer then outputs the product of SKIL and the first step of the waveform. The first index register is then counted down, and when it reaches zero, the product of SKIL and the next step of the waveform is output. When the end of the cycle is reached, $IFLG is checked to see if the note is over, and if it is, the next word from the melody is loaded, but if the computer is to continue with the present note, the amplitude and pitch are reloaded, and another cycle is sounded.

Remember that as all this is going on, both clock interrupts are repeatedly interrupting and incrementing the second index register (advancing the envelope and waveform) and $IFLG.

The monophonic sound generator is much more than just the previously described PLAY subroutine. As can be seen from the listings, it contains the table of pitch values that are used to load index 1, the control for the main list of choices, and the program which calls the PLAY subroutine.

The sound generator has 3 entries, PLAY (i. e. PLAY+SEE or PLAYER), RECORD, and PLAYPART. RECORD is the same as PLAY, only the computer waits for a sense tape to be detected by the tape recorder before it starts playing. The flag T2 is used to determine whether or not PLAYPART was chosen. The program, after initializing T2, sets up the interrupts, and then stores the address of the first note of the first bar of the arranged melody in PEEC, or if PLAYPART was chosen, it scans through the melody until the note under the cursor is found, and the arrangement table pointer (POIN) points to the proper bar. The address of this note is then stored in PEEC. A call is then made to the PLAY subroutine. When control is returned from this subroutine, it means a bar line has been encountered. In this case we advance to the next bar in the arrangement table and recall the PLAY subroutine, or, if the last bar played was the last bar in the melody, a branch is made to the part of the program that displays the main list of choices.

1ST PLAY

Most of the monophonic sound generators are the same as 2ND PLAY, differing only in the number of steps in their waveform, and the waveforms used. A few, however, make use of the waveforms in the long ('2000 word) tablet controlled waveform tables. These are 1ST PLAY, 6TH PLAY, 7TH PLAY, 8TH PLAY and 10THPLAY. As 1ST PLAY is the basic generator of this type, most of the explanation will be centered around it.

1ST PLAY differs only slightly from 2ND PLAY. In 2ND PLAY, a negative number was loaded into index register 1, an output of the amplitude of the current waveform step was made, and then the index register was incremented until it reached zero. Thus, the timing of the waveform step is controlled by the number loaded into index register 1. In 1ST PLAY, however, a negative number is loaded into the first index register, an output is made based on the first point of the waveform table (the product of SKIL and that first point), the index register is incremented, then an output is made based on the second point, the index register is incremented again, and so on, incrementing the index register and outputting successive points of the waveform table. When the register reaches 0, the cycle is repeated, the negative number is reloaded into the register, and the first point of the waveform table is output again. The index register is used to time a whole cycle instead of a fraction or a step of the cycle. The instruction which loads the present position of the waveform is named ZO or ZO8. ZO contains the instruction "load from $FORMTB (start of waveform) minus NPCH (negative number that is loaded into index and used for pitch control) plus index 1". At the beginning of the cycle NPCH equals index 1 so that the number loaded by ZO is the first number in the waveform, and as index 1 is incremented, it causes the advance through the waveform.

A feature of 1ST PLAY is its ability to slide along the waveform. This phase slide uses a memory location called ZX. ZX is set equal to ZO at the beginning of each note (unless the slur is on), then every time that the envelope and glissando interrupt is activated, ZX is incremented by one. If the phase slide is on, ZX replaces ZO at the end of every cycle. The effect of this is that the segment of the waveform used to produce the note slowly drifts through the waveform changing the sound of the music as it drifts.

Another feature of 1ST PLAY is a down octave command. In this case the amplitude in every other cycle is set equal to zero, which effectively lowers the pitch one octave. The sound produced is slightly different, but the difference is hardly noticeable.

In 1ST PLAY, index 1 is used to time the pitch, and index 3 is used to advance through the glissando and envelope (index 2 is used for this in 2ND PLAY). As an index register is not used for advancing through the waveform as in 2ND PLAY, it may be used instead of $IFLG for counting down the duration of the note. Therefore, the second index register is loaded with the number of 32nd notes (negated), and the clock which interrupts the program every 32nd note increments index 2. When index 2 reaches zero, a zero is stored in $IFLG to indicate the end of the note. As incrementing an index is quicker than incrementing a memory location, the interrupt is slightly quicker, and a faster tempo is possible as less time is spent in the interrupt.

As mentioned before, there are other players that use the 1ST PLAY technique for sound production. 7TH PLAY is the same as 1ST PLAY except for a primitive sort of portamento command that is available. When the portamento is on, the number that is loaded into NPCH (to determine the length of the cycle) is one eighth the difference (truncated) between the written pitch and the pitch used in the last cycle, plus that last pitch. This causes the pitch to slide to the present pitch from the previous pitch.

6TH PLAY, 8TH PLAY and 10THPLAY do not have a phase slide but they do have a "waveform modify" facility. When this facility is on, the waveform output is not a segment of the waveform table, but rather a sampling of it. Before the beginning of each cycle, '2000 (the number of points in the waveform) is divided by the number from the pitch table (the one that is negated and loaded into index 1 and NPCH). The resulting number gives the number of steps to advance through the waveform each time index register 1 is incremented by one, so that the end of the waveform is reached when the index register reaches zero. In this way, the whole waveform is advanced through once per cycle no matter what the pitch is.

8TH PLAY and 10THPLAY are slight extensions of 6TH PLAY. They both have the capability of producing a noisy sound output. This noise can be turned on and off using the glissando on and off commands. In 8TH PLAY, the noise is on only when the glissando is off, and in 10THPLAY both noise and glissando are on, or they are both off. This is the only difference between the two sound generators.

The noise is created by adding a pseudorandom number to CNTR, (the counter that points to the present position in the waveform). The pseudorandom number is generated by multiplying a K1 (any number) by a K2 (say '1234567) and storing the second word of the result back into K1. K1 would then be changed rather randomly by each multiplication.

POLYPHONIC SOUND GENERATOR

Chapter 4 gives a description of how the polyphonic sound generators are used, and a description of the logic behind the programming can be found in ERB-862, Reference (18). The KEYBOARD listing in that report is slightly out of date, and a more modern listing can be found in Appendix H of this report.

VI. WRITERS

The following chapter describes in detail the most complicated programs in the computer music system. These are the programs that are responsible for writing the music on the screen. This includes the drawing of the staff and then writing in of all the notes, rests, commands, bar lines, and markers; and the user controlled overwriting, inserting or deleting of new notes, rests, etc. There are five different WRITE programs. DOLY writes one line of music on the screen, and is the program loaded with the music system. PLAPIC writes the music into the external memory so that it can be displayed while the music is playing, and is also loaded with the music system. DRAMU, loaded on top of the envelope, writes the music on paper using the digital plotter. SLIDER is a new version of DOLY which is overwritten in the same memory area as DOLY when PITCHGEN is loaded. It allows the writing of music anywhere on the staff, not just on the semitones. 4 WRITE displays all four lines of music at once and allows overwriting, inserting and deleting of notes in any chosen voice. This program is loaded on top of the three waveform tables.

In this chapter, the names of in-core counters or storage places will often be referred to without any explanation of their meaning, especially after they have been mentioned and explained once. In the appendicies there are tables containing the names of all these memory locations, their meaning, and sometimes a mnenomic instruction or an index that the memory location contains. If the name contains a comma followed by a number, this name refers to a small data table and a specific element of the table is located by adding the contents of the index register to the address of the name. The following line is an example from Appendix I.

```
K5,1     E     1,STA            address of end of arranged bar line tables
```

As this is from DOLY, index 1 is used to refer to the four voices, and ranges between 0 and 3, the above statement means that there is a table starting at address K5 +0 (or K5) and ending at K5+3. Each of the four elements of the table contain the address of the end of the arranged bar line table for the voice that it refers to. (K5 refers to voice one, K5+1 refers to voice two). These memory locations also contain the mnemonic command STA, which is indexed with index 1.

Another point to note is that the flow diagrams sometimes include the name of the entry to the section of the program that is being explained. This is useful only if the listings are being consulted simultaneously. These flow diagrams refer to the A and B accumulators as *A and *B. Index 1, 2 and 3 are referred to as I1, I2, and I3.

DOLY

DOLY is the longest program that is loaded with the system, and it will be explained in sections. The first section is the initialization procedure which sets flags to indicate which of the many choices was last chosen. DOLY also must change default values when a new voice has been selected. The display loop is responsible for drawing the staff and the notes on the screen. Each time it comes to the note which has the cursor, a check is made to see if any buttons have been pushed, or shaft encoder wheels turned. The input processing routine is responsible for this and making the necessary changes. In addition to these main sections of the program there is a little

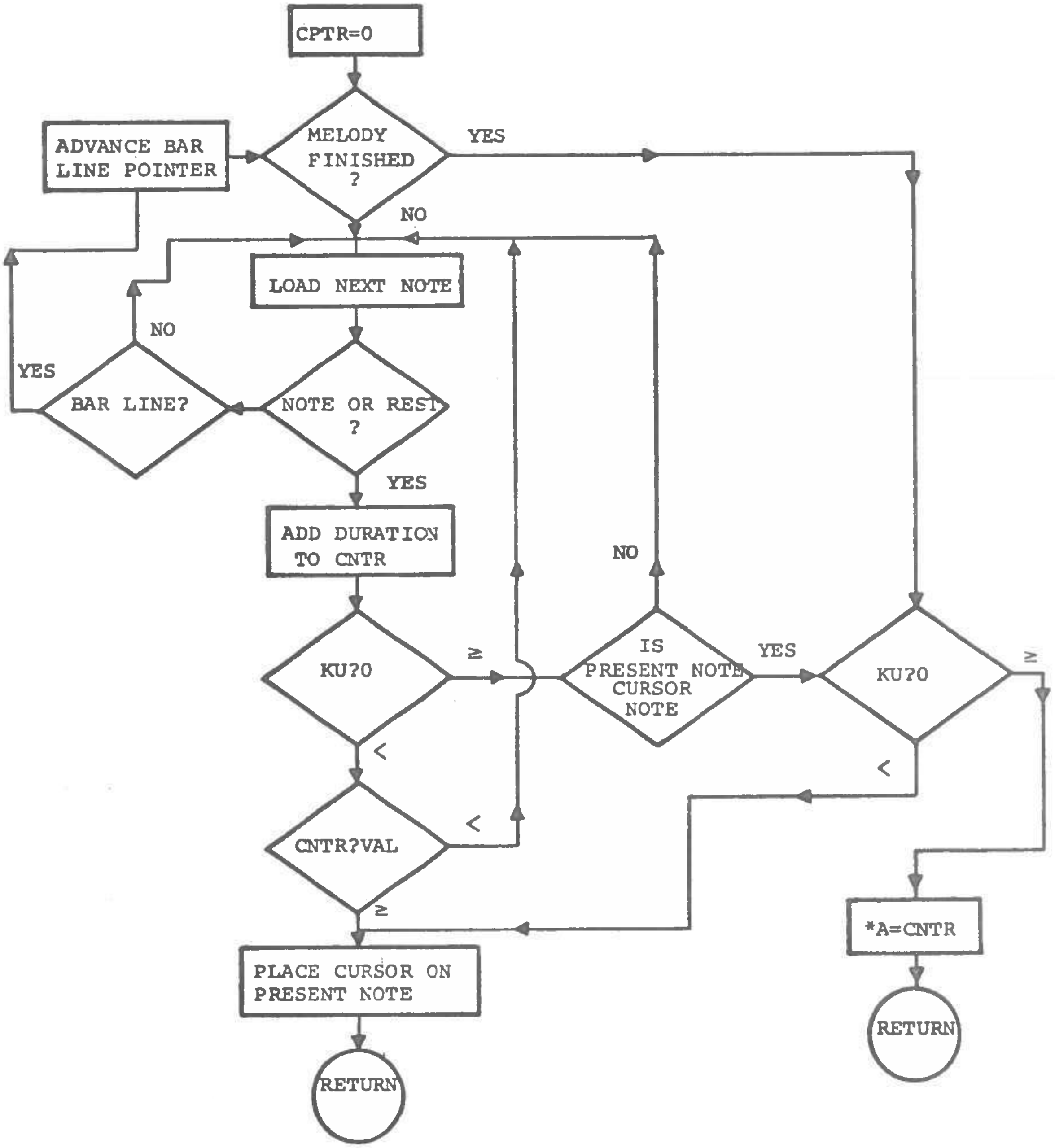


Figure 12. COUN Subroutine

subroutine that is called every time buttons are pushed. This clears KFLG which indicates that the buttons have been pushed, and stores the input word in KEYB.

INITIALIZE

The choice WRITER or WRITE+CH causes a branch to the main set of initialize instructions. These are the instructions that are responsible for changing the mode of the computer from voice to voice.

The first thing to be initialized is the position of the cursor. If the cursor is past the second note, ADDX (cursor position) will be greater than 2. In this case ADDX, INDX and IADD are all decremented and UPTR is incremented until ADDX reaches 2.

The next step is to find the duration of the melody in 32nd notes.

There is a routine called COUN which scans through the melody following the arranged bar line table, and adding to CNTR, a counter, the duration of each note. If K4 is set positive COUN will stop at the address of the note where the cursor was last left, and return the duration of the piece up to this point in the A accumulator. If K4 is negative, the count is continued until it equals or exceeds the value in VAL. It then sets UPTR (address of the first word on the screen) to equal the last note checked minus the number of words the cursor is in from the beginning of the display. The duration now, between the beginning and the new note under the cursor, will be about equal to VAL.

This subroutine is used by the initialization routine twice. The first time, K4 is positive, and the duration up to the cursor is found. This is stored in VAL. A list of the four voices is then presented, and when one is picked, a subroutine NENT is called. This initializes the external pointers TUTAB, TUMEL, THUD, TULUP2 and TULUP4 (see Appendix A) and the internal pointers MEEP, BEEP, JPTR and FIRS (see Appendix I). Calls are then made to DAMEL and TRAN to set up the arranged bar line table. The second call is then made to COUN after K4 has been set negative. With the duration between the beginning of the old voice and the old position of the cursor in VAL, subroutine COUN moves the cursor to a spot in the new chosen voice where the duration between it and the beginning of the voice is the same as it was in the last voice. This completes the special WRITER or WRITE+CH initialization.

The next step in the initialization is the setting up of flags so the program will know what command was chosen. The final condition of these flags after this step is shown in a chart in Appendix I.

Finally, there is the final initialization of KPTR, RPTR and UPTR to the values shown in Appendix I.

DISPLAY LOOP

The DISPLAY LOOP is really two loops, one inside the other. The outer loop starts by drawing the staff on the screen, and resetting index 2 and CRFL. The inner loop then takes over to plot the actual notes. Each cycle through the inner loop causes one word of the melody to be plotted. At the end of the loop, Index 2 is incremented and a check is made to see if the words being plotted have reached the end of the staff. If so, a branch is made back to the outer loop, if not the inner loop is repeated.

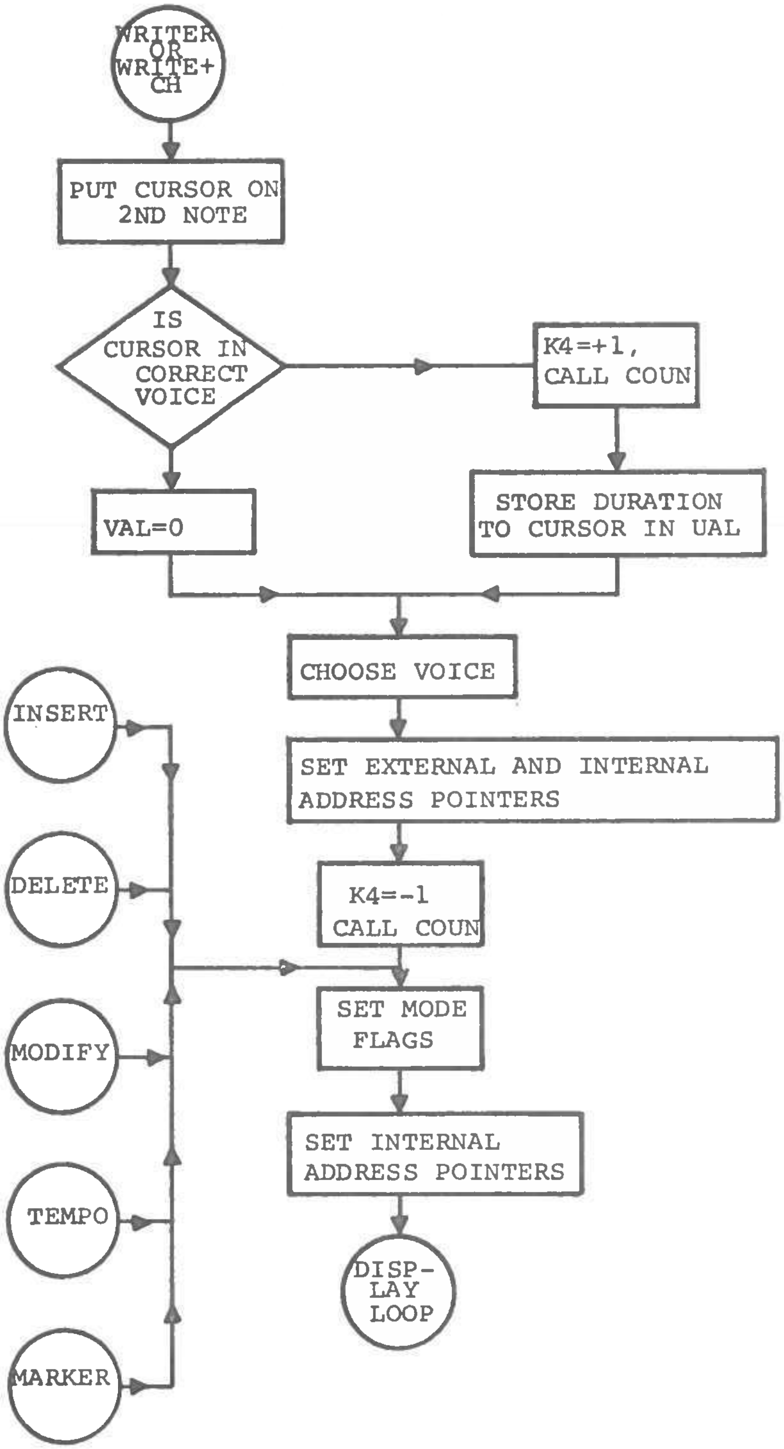


Figure 13. Initialize Routine

The word UPTR contains the address of the first word to be displayed on the screen plus '40. It also contains an Index 2 command, so that if a load is made using UPTR, the address of the word loaded is the sum of the address of the first displayed word, '40, and Index 2. After the staff is drawn, Index 2 is set to -'40. This means that UPTR will point to the address of the first displayed word. Every time the inner loop is cycled through, Index 2 is incremented which will cause UPTR to point to the next note.

At the beginning of the inner loop, the word that UPTR points to is checked. If UPTR points to a note as opposed to a rest, command, or bar line, it is checked to see if it has a sharp sign, and if it does, a sharp sign is plotted with an X position equal to POS, and a Y position depending on the pitch which is found in the least significant 15 bits of the note. Each bit of the duration bits (3-9) are checked and when one is set, the corresponding note is drawn; for example, if bit 3 is set, a whole note is drawn, or if bit 5 is set, a quarter note is drawn. The bit after the one that was set is then checked and if it is on, a dot is placed after the note. If this bit is not on, we go back to look at the next bit that is on and if one is found, a corresponding note tied to the first one is drawn. For example, if bits 3 to 8 of the note word are set to 100 110, a whole note is drawn as bit 3 is set, bit 4 is not set so no dot is drawn, the next bit that is set is bit 6 which causes an eighth note to be drawn close to the whole note, and bit 7 is on so a dot is drawn after the eighth note.

After the complete note is drawn, the X position is incremented depending on the size of the note just written and is then checked to see if the end of the staff has been reached. If it has not, a return is made to the beginning of the inner loop to check the next note.

If UPTR (the address of the note that is loaded) points to a rest, it is dealt with in the same way as the notes were but with rest symbols being drawn instead of note symbols. Again a check is made to see if the end of the staff has been reached and if not, a branch is made to the beginning of the inner loop.

If a bar line is found, it is simply drawn, the-end-of staff check is made, and followed by a return to the beginning of the inner loop.

Tempos, messages, and markers if found, are written on the staves. A small computation and a call to SPDTA and DELEZE have to precede the writing of a TEMPO to change the value to the number of quarter notes per minute and then to change it to 4 ASCII characters representing a decimal number.

When enough notes have been displayed so that the staff has been filled up, the program leaves the inner loop, to re-enter the outer loop. A check is then made to see if the WRITE mode is set to WRITE+CH as opposed to WRITER. If it is, SHOO is incremented. If SHOO is now equal to 0, it is reset to -11, and K4 set to a positive number, COUN is called to count up the duration to the cursor and store it in CNTR. This number (in 32nd notes) is divided by 8 with the result displayed on the lower left side of the screen after a call to SPDTA and DELEZE. The remainder is then printed as a second number. The duration up to the cursor is now given by these numbers, the first number being the number of quarter note line durations, and the second being the remainder in 32nd note time durations.

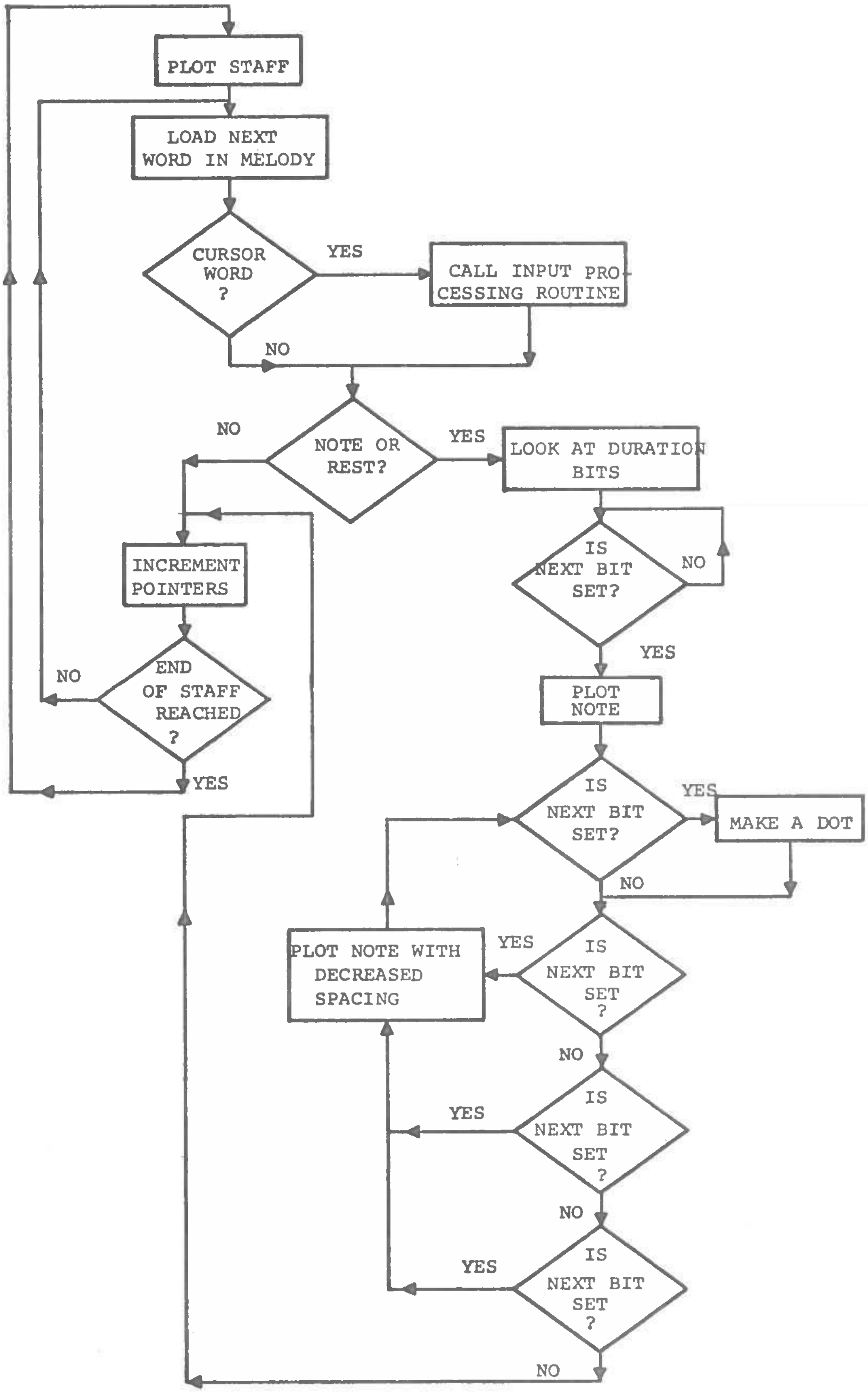


Figure 14. Display Loop

As the duration is only checked each time SHOO reaches 0, and SHOO is then set to -11, the duration is checked every eleven times through the outer loop. If SHOO was incremented but did not reach 0, the present duration value is still displayed on the lower left side of the screen before the return to the beginning of the outer display loop.

INPUT PROCESSING

Something that was not mentioned before was that just before a note is loaded to be displayed, a check is made to determine whether or not it is on the cursor line. To do this, index register 2 (which was loaded with a -'40 before the first word was drawn, and is incremented after each word is finished with) is stored in CPTR. A '40 is subtracted from ADDX which represents the number of notes that are drawn before the cursor is drawn and this is compared to CPTR. If they are the same then the next note to be drawn is the cursor note.

When the cursor is found in this way, a word is input from the shaft position encoders. The horizontal position from this word is stored in ENC and the vertical position is MWRD. The cursor is then drawn, with a dot placed at the vertical position MWRD.

The cursor motion depends on three variables, ENC (present horizontal wheel position) ENCO (old horizontal wheel position) and IADD (counter). ENCO is compared to ENC. If it is less than ENC, then the wheel position has been increased by one, and IADD is incremented; if ENCO is the same as ENC then no change is made, and if ENCO is greater than ENC, IADD is decremented by one. IADD is then checked to see if it is less than 2 or greater than 7. In the former case IADD is decremented and UPTR is incremented, while in the latter case IADD is incremented and UPTR is decremented. A check is then made to see if the boundaries of the voice have been reached, and if so a correction to UPTR is made. IADD is then stored in ADDX, and as IADD is kept between 2 and 7, the note which the cursor is on is kept between the third and eighth notes (inclusive). It is only when one of these boundaries have been reached that the actual music starts to move along the screen.

The foot pedals are then checked, and if the right pedal is down, FFLG is set to a negative number and if the left pedal is down, LFFL is set negative.

KFLG is now checked to see if the button interrupt has been set since the last check. If it has not, a return is made to the display loop.

If the buttons have been pushed, the action necessary will depend on the flags MAFL, TFLG, MFLG, DFLG, and NFLG. All these flags are set to a -1 or a 0 depending on the entry that was chosen to the WRITE program. (see chart in Appendix I). In the case that WRITER has been chosen, the buttons give the duration of the new note, and MWRD the pitch, and the combination of these make the note word which is stored at the address given by UPTR (presently the note under the cursor). INSERT calls a subroutine SPRD which moves all the words from the cursor to the end of the voice forward one word in memory and then stores the input note under the cursor (UPTR).

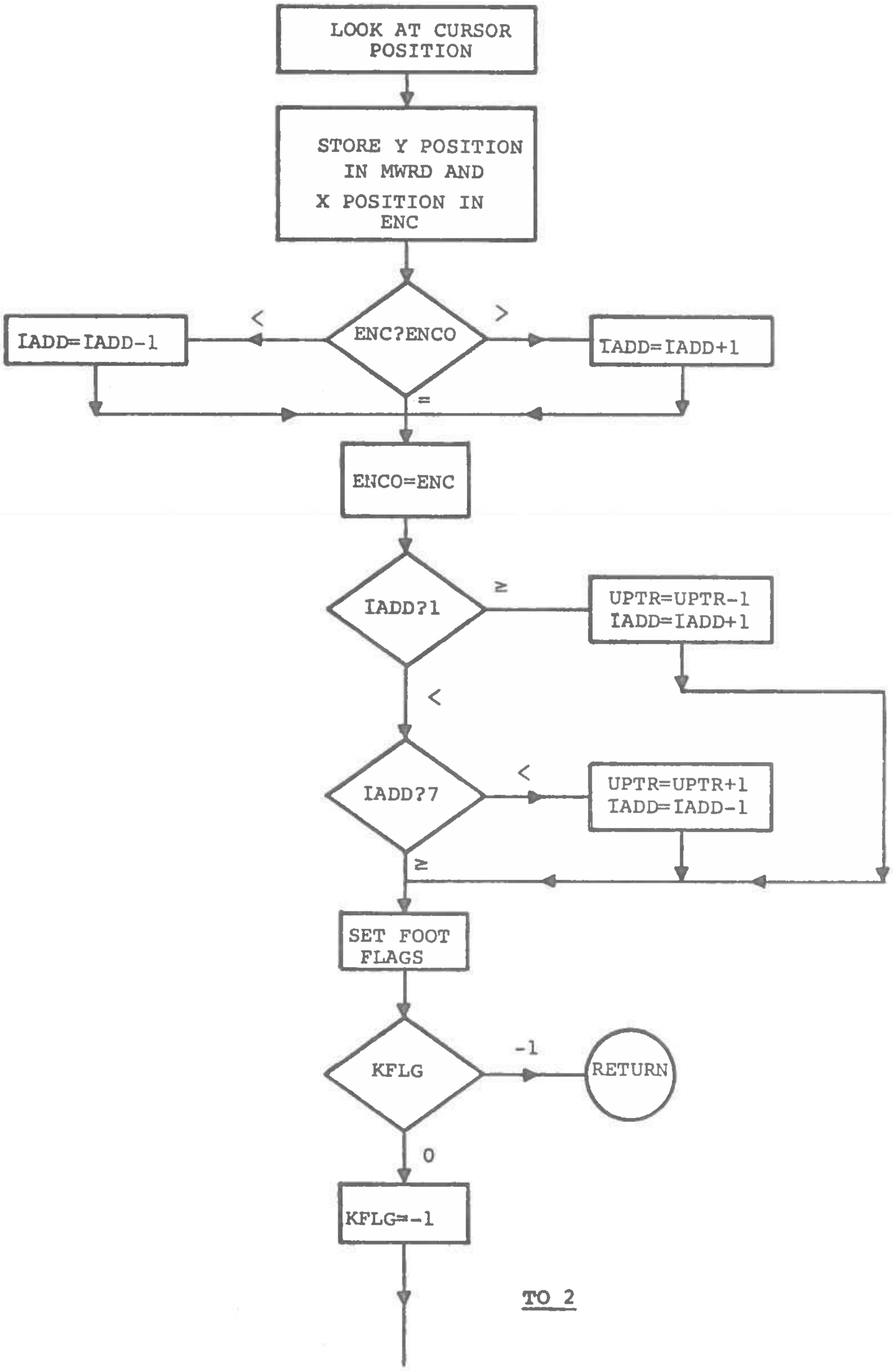


Figure 15. (a) Input Processing Routine

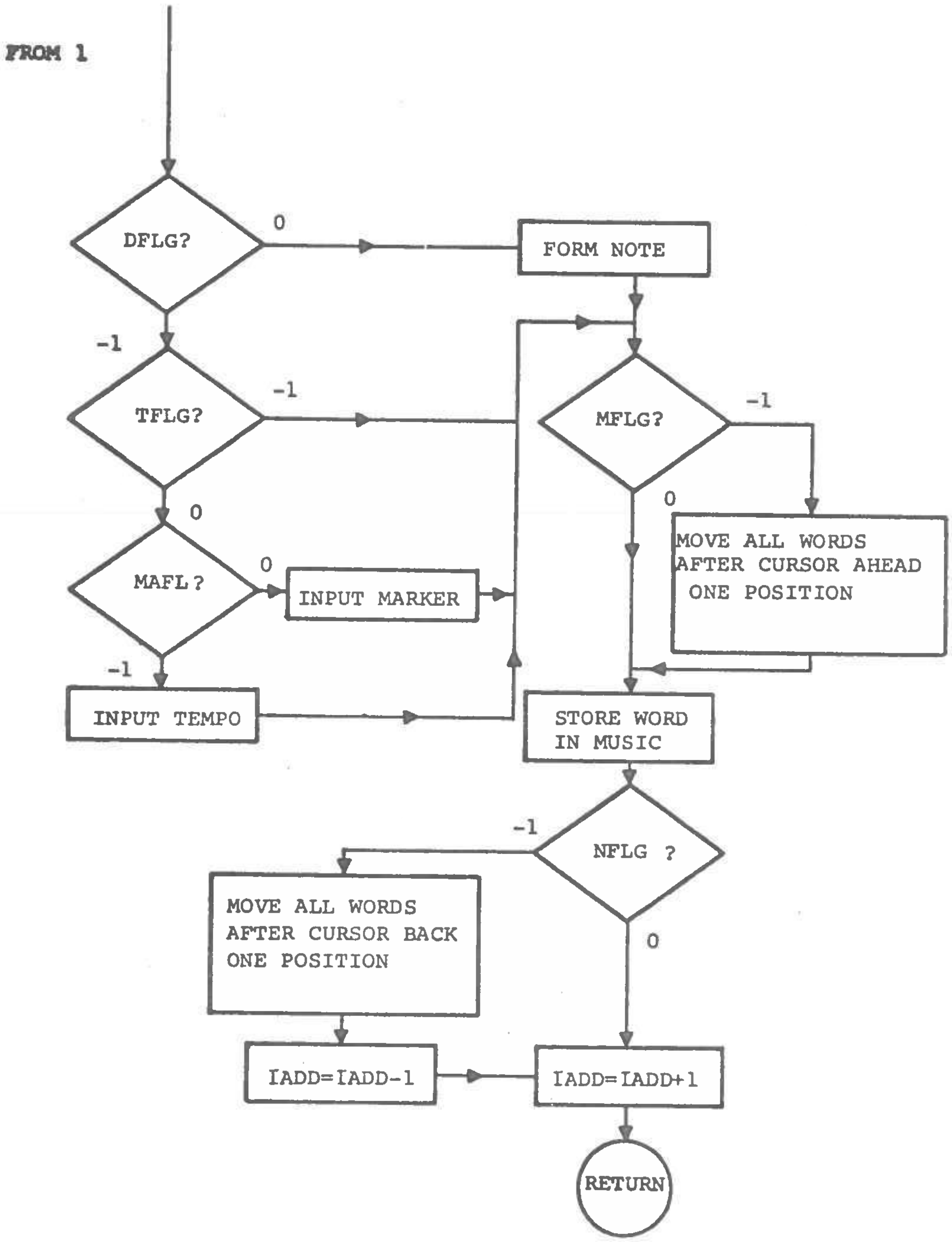


Figure 15. (b) Input Processing Routine

MODIFY, MARKERS, and TEMPO all calculate what they have to store, call SPRD, and then store their input word under the cursor. Delete also calls SPRD, but as it has NFLG set, SPRD moves every word after the cursor back one. IADD is also decremented.

After this has all been done, IADD is incremented so the cursor will end up on the next word. In the case of DELETE, IADD has already been decremented so the cursor ends up on the same word.

OTHER ENTRIES

There are two entries to DOLY which do not display any music but use routines in the program. SET TEMP uses the same routine for translating the typed TEMPO command as MOD TEMP, but stores the result in SPED instead of in the actual melody. SPED is the initial default tempo.

PATCH2 is a subroutine which uses the WRITER initialization routine to call DAMEL and TRAN to set up arranged bar line tables for all four voices when it is loaded by the system. GETTER calls PATCH2 whenever a melody is loaded.

PLAPIC

PLAPIC is a shortened version of DOLY leaving out most of the initialization and all of the input processing. All of the output is loaded into the external memory to control the display while the computer is playing music.

When PLAPIC is called, both index 1 and 3 are saved in LIX1 and LIX3 respectively. The initial address is loaded from PLCPIC (pointer to the current bar of the arranged bar line table) and added to PLBPIC (distance of the first note to be displayed from the beginning of the bar).

The display loop is then used to write the notes into the external memory. When the end of the staff is reached, the external memory is reset and activated, and the number of notes that have been displayed minus 37 is returned in index 2. Index 1 and 3 are reloaded with their original contents from LIX1 and LIX3.

DRAMU

Another simplification of DOLY, DRAMU outputs the music to a digital plotter. When called, it moves the plotter pen to the top of the paper, draws the double staff and proceeds as in the DOLY display loop. When the end of the staff is reached, a second double staff is drawn below the first one. After the third double staff has been completed, the pen is sent to the top of the paper, past the first staff, to start a new page.

A major difference between DRAMU and DOLY is the absence of character plotting in the plotter. To compensate for this,DRAMU has to call the program CALCAR Reference (3) each time a character is plotted.

SLIDER

SLIDER is very close to DOLY. By changing the tables of the cursor lines, eliminating the sharp signs, enabling the dot to be placed anywhere on the screen and storing the actual position of the dot in each written note you have the SLIDER program. Everything else in the program is the same as in DOLY.

4 WRITE

This is the most complicated program in the music system, and one that will be of extreme use when fully debugged and streamlined. An extension of DOLY, this program displays all four voices on the screen at once. WRITING, INSERTING, etc, is allowed in any previously chosen, single voice. This chosen voice is displayed in a different colour; with brighter and larger notes, and with all bar lines and commands. The other voices do not have their bar lines or commands displayed. If LOOK is chosen the voices that are displayed can be chosen by using the buttons. (Appendix B, Table 6).

To keep all four voices matched up, and representative of a true picture of the music, it was necessary to have 4 WRITE write the melodies in their arranged order. If, for example, the arrangement for a particular voice is 1 2 1 2 0, the first two bars will be displayed twice followed by a double bar line.

This program is divided into even more segments for the purpose of documentation than DOLY was. The additional segments are MESSAGES, the part of the display loop that is responsible for displaying messages; GOING FORWARD, for advancing through the music, and GOING BACKWARD, for retreating through the music.

Most data tables in this program are referenced by index 3, which contains a number between -4 and -1, with -4 referring to VOICE 1, -3 referring to VOICE 2, -2 referring to VOICE 3, and -1 referring to VOICE 4. For example, COLR, 3. contains the colour information of the four voices. This means that the four memory locations immediately preceding COLR contain this colour information. The actual memory location COLR is different and not connected to the contents of COLR, 3. (In this case, COLR contains the information that is put into COLR, 3 in the case of red notes).

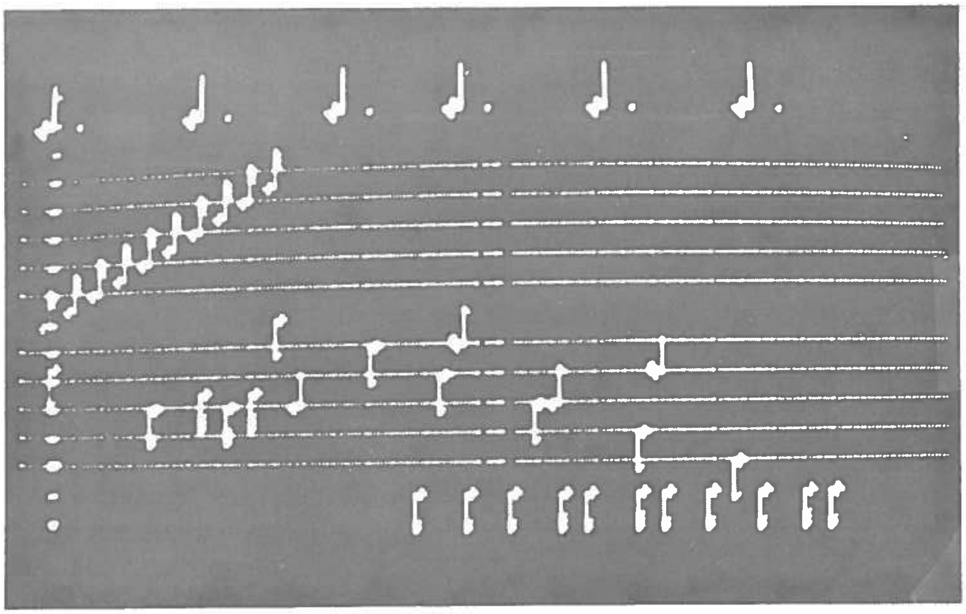


Figure 16. Music Written Using the 4 WRITE Program

INITIALIZATION

When 4 WRITE is loaded, the present red button command is stored in INTZ and a new command causing a branch to the 4 WRITE list of choices is stored as the new red button command. When the red button is pressed, the 4 WRITE list of choices will appear, and the old red button command from INTZ will be reactivated, so that if the red button is pressed again, the main list of choices will appear. When the program is loaded, it branches to the LOOK initialization.

As in DOLY when WRITER is chosen, when VOICE 1, VOICE 2, VOICE 3, or VOICE 4 is chosen, the external addresses TUTAB, TUMEL, THUD, TULUP2 and TULUP4 have to be reset for the new default voice, and calls are made to DAMEL and TRAN. The number of the default voice, minus five, is stored in index 2. The resulting number is a number between -4 and -1. If LOOK is chosen, index 2 is set to 0.

For all the entries into the program, T is set to a number representing the entry that was used. Next the colours, size, intensity and spacing of the notes is set to blue, small, dark, and small respectively, for all the voices. Then the chosen voice is reset to red notes, large notes, bright intensity, and large spacing. An example of what is set if VOICE 1 is chosen is shown below.

	COLR,3 Colour	BRIT,3 Scale & Intensity of Notes	BRI,3 Scale & Intensity of Sharps	XNCR,3 Sharp & Tied Note Spacing
VOICE 1	COLR(red)	BRIT	BRI	XNCR
VOICE 2	BLUE	DARK	DAR	XNCS
VOICE 3	BLUE	DARK	DAR	XNCS
VOICE 4	BLUE	DARK	DAR	XNCS

If LOOK had been chosen, all the voices would be reset for large bright notes with large spacing.

DISPLAY LOOP

The most complicated difference between DOLY and 4 WRITE is the display loop. The notes in different voices have to be matched up vertically depending upon the total duration between these notes and the beginning of the piece. For example, in one section of the piece, a note in each voice may start on the 513th 32nd note into the piece. These four notes will have to be on the same vertical line even though one might be the 20th note into the piece, another, the 35th note in, etc. To this end, running totals of "elapsed" 32nd time intervals are stored.

The display loop begins with the drawing of the staff and the branching to the Interrupt Processing routine (described later in this chapter). On return from this

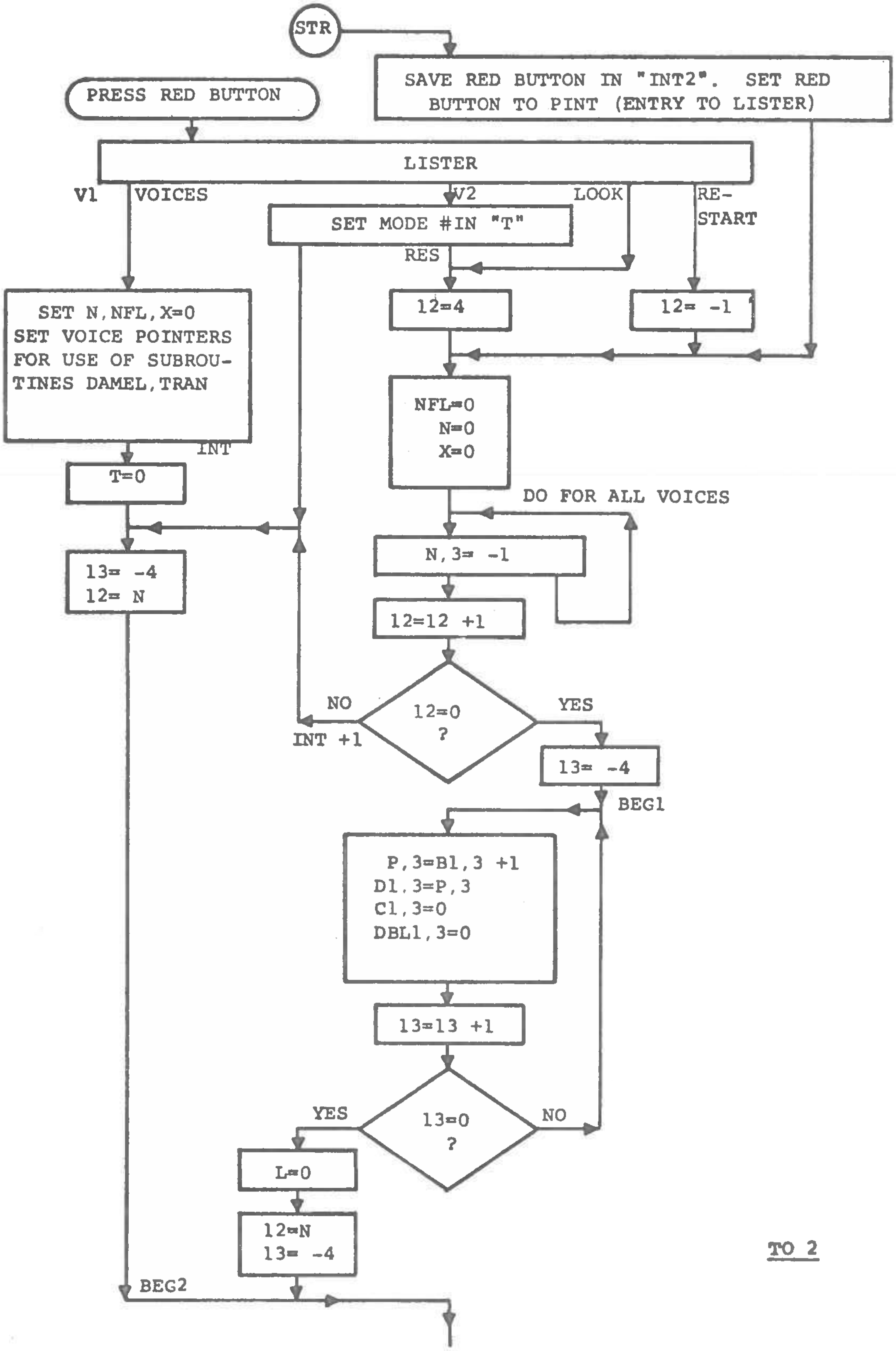


Figure 17. (a) Initialization

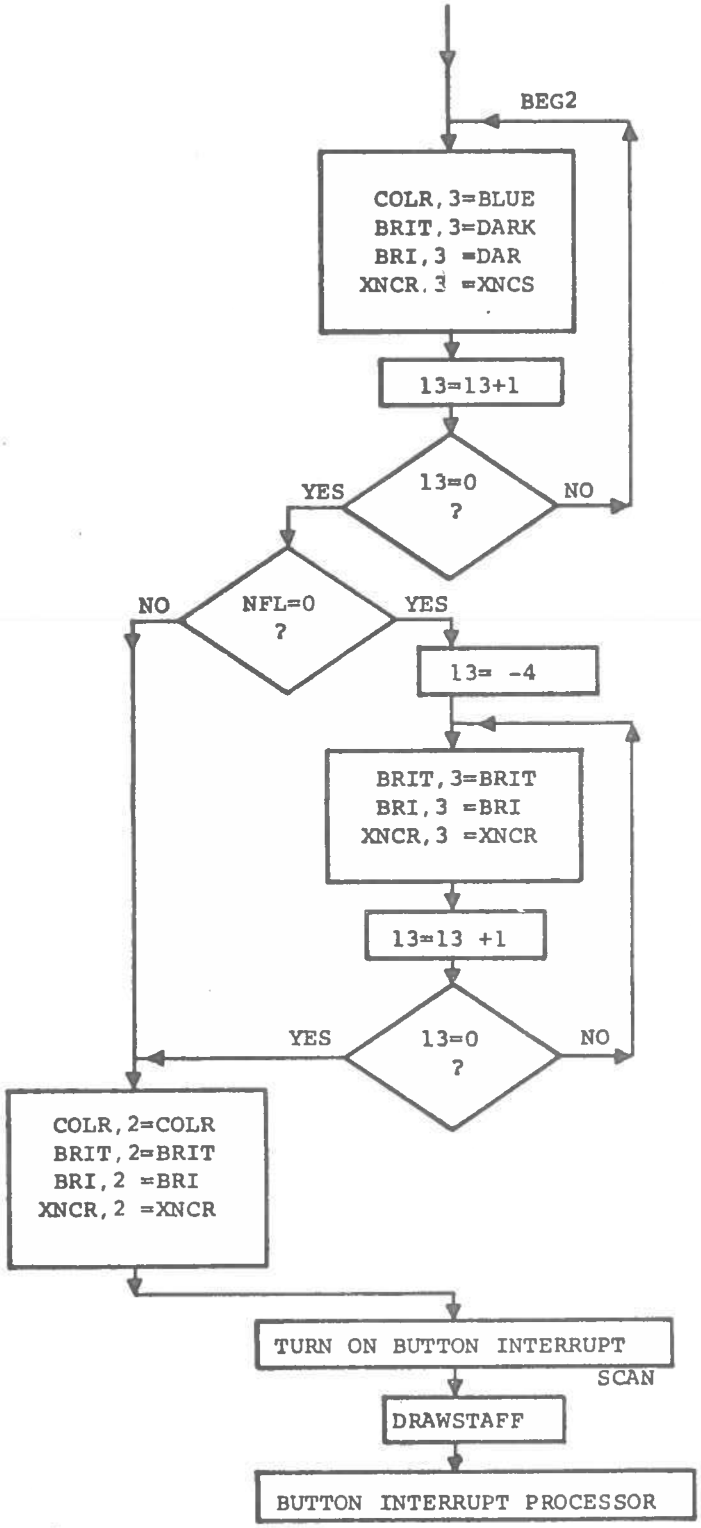


Figure 17. (b) Initialization

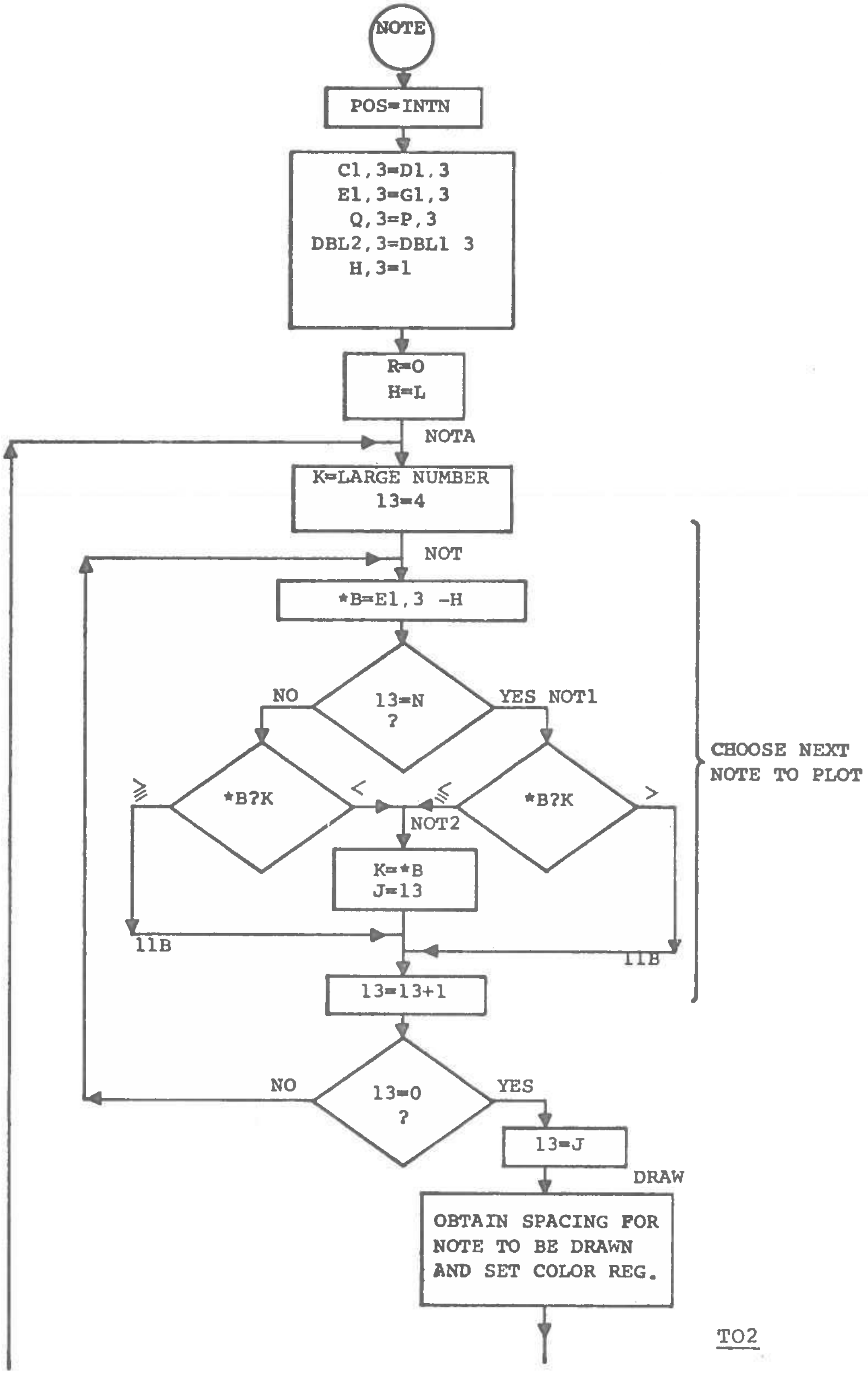


Figure 18. (a) Display Loop

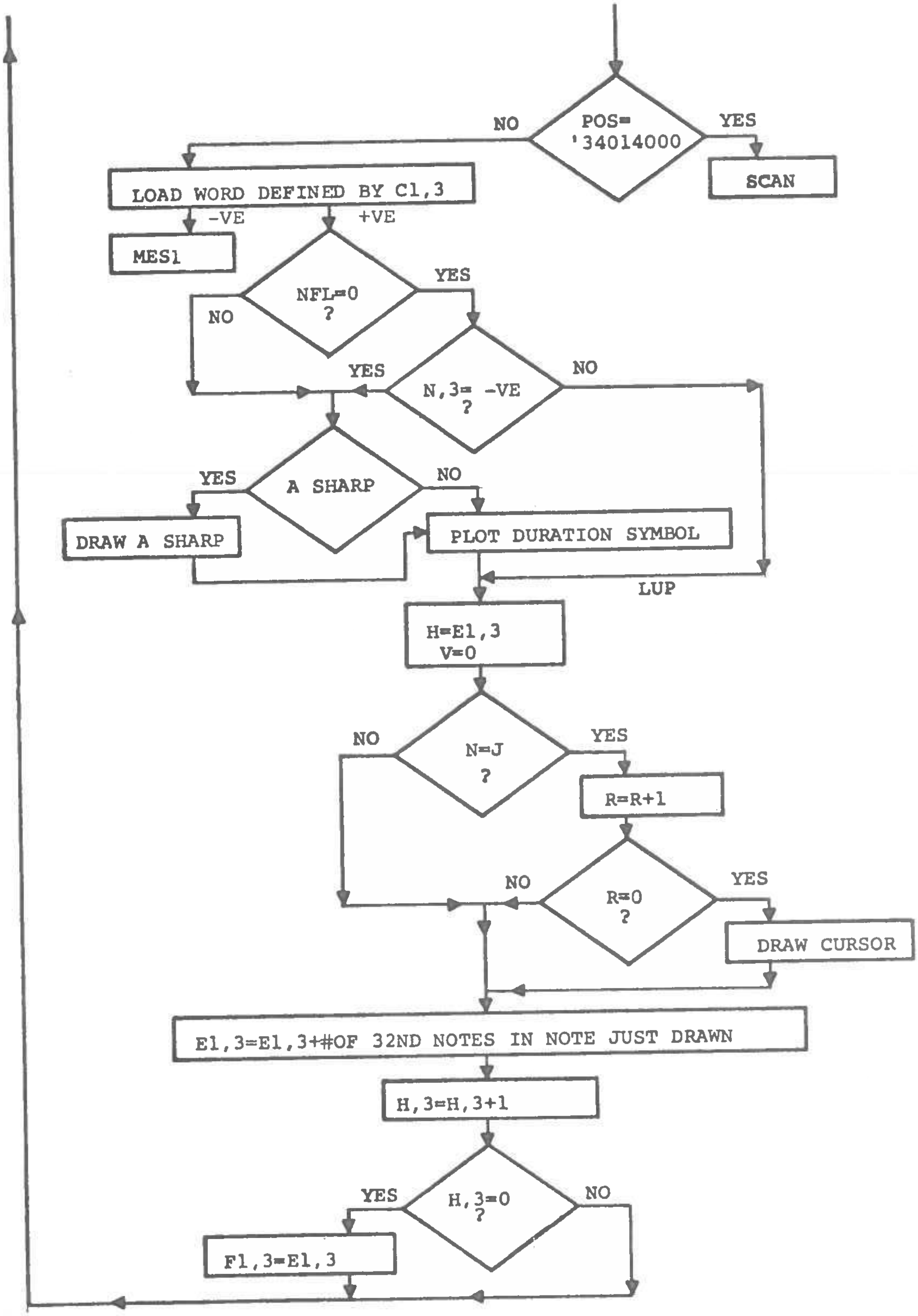


Figure 18. (b) Display Loop

routine, the X position is initialized and one of the above mentioned counters, three pointers, and a flag (C1,3, E1,3, Q,3, DBL2,3, H) are set to their initial parameters (D1,3, G1,3, P,3, DBL1,3, L). The cursor indicator R is then reset to U, and all H,3's are set to -1. Each time through the display loop, we will have a pointer to the next note to be displayed in each voice (C1,3), a pointer to the current position in the arranged bar line (Q,3), the number of elapsed 32nd time intervals up to but not including the last note that was displayed (H).

The abovementioned data are enough to find the next word to be displayed. This next word will be the one with the smallest E1,3. If there are two or more voices that have the same E1,3 and one is the chosen voice, then the note in the chosen voice is drawn first.

The next thing to be done before the actual note is drawn, is to update the X position (POS). The difference between H and E1,3 which is the number of 32nd notes between the last note and the one about to be displayed is found, and a number proportional to it is loaded and added to POS. In the case of notes with the same E1,3 as the last one, E1,3 will equal H, and a 0 will be added to POS.

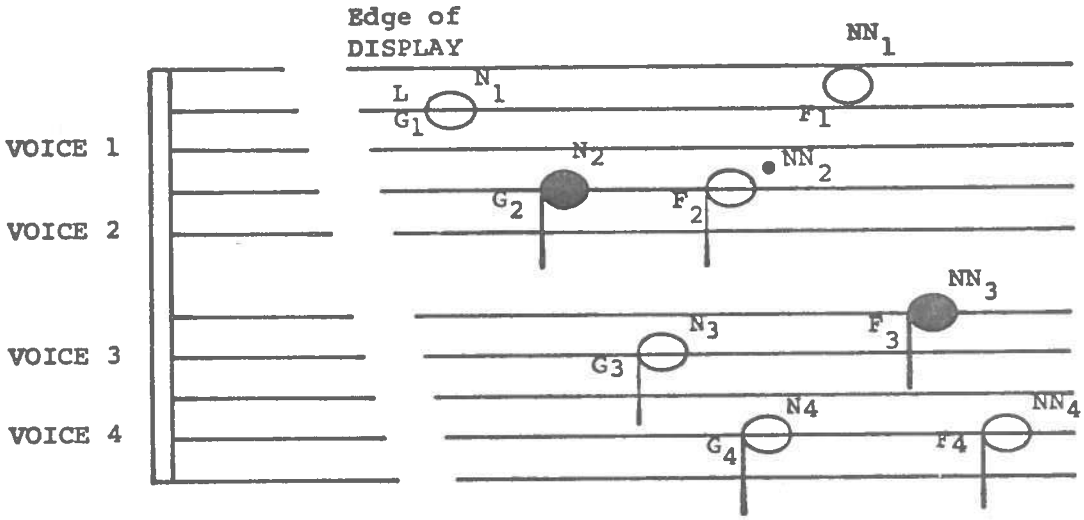
A check is made to see if the end of the staff has been reached; if it has, the duration up to the cursor is written in the lower left corner, if LOOK was not chosen, and a return is made to repeat the scan.

If the end of the staff has not been reached, the program is finally ready to draw the word. When the smallest E1,3 was found, the voice that contained it was stored in J, and after J is loaded into index 3, C1,3 will point to the word that is to be drawn. If the word is not a note or a rest, a branch is made to the Message Processing routine described in the next section. If the word is a note or rest, it is plotted as in DOLY.

After the note has been drawn, the E1,3 of the note is stored in H. R is incremented if a note was drawn in the chosen voice, and if R reaches 0, a branch is made to draw the cursor and to store a -1 in Z. This indicates that the cursor has just been drawn.

C1,3 is incremented and E1,3 must then be updated so that it includes the number of 32nd notes up to and including the note just drawn. This is done by adding the number of 32nd notes in the note just drawn to the old E1,3. During the initialization of the Display Loop a -1 was stored in the H,3's. H,3 is incremented for the voice that the note which was drawn is in. If it reaches 0, this means that the note just drawn was the first one drawn in that voice, and the new E1,3 is stored in F1,3. This F1,3 is used in the GOING FORWARD routine. Z is incremented and if it reaches 0, the cursor was the last note drawn, and the new E1,3 is stored in Y. This represents the number of 32nd notes up to and including the one under the cursor. With everything updated, a branch is made to the beginning of the Display Loop to search for the next word to be displayed.

Fig.19 shows the plotting of the first eight notes in a melody. E1,3(1) represents E1,3 for the first voice.



N - first displayed note of voice

NN - second displayed note of voice

G1,3 - Nö of 32nd time intervals up to but not including 1st note on display

L - smallest G1,3

F1,3 - no of 32 time intervals up to and including 1st note on display

Figure 19. Plot of 1st 8 Notes of a Melody

STEPS

1	Plot N1	H=E1,3 (1)	F1,3 (1)=	E1,3 (1)=E1,3 (1)+duration of N1
2	Plot N2	H=E1,3 (2)	F1,3 (2)=	E1,3 (2)=E1,3 (2)+duration of N2
3	Plot N3	H=E1,3 (3)	F1,3 (3)=	E1,3 (3)=E1,3 (3)+duration of N3
4	Plot NN2	H=E1,3 (2)		E1,3 (2)=E1,3 (2)+duration of NN2
5	Plot N4	H=E1,3 (4)	F1,3 (4)=	E1,3 (4)=E1,3 (4)+duration of N4
6	Plot NN1	H=E1,3 (1)		E1,3 (1)=E1,3 (1)+duration of NN1

MESSAGES

The message routine deals with bar lines, double bar lines, commands, tempo and markers. To avoid confusion, these are drawn on the screen only if they are in the chosen voice. In the case of a message which is not a bar line and not in the chosen voice, C1,3 is incremented and a branch is made back to the display loop to load the next word in the same voice. This return point in the display loop is the only return point that the message routine uses.

The processing for a bar line is relatively the same whether or not it is in the chosen voice. If it is in the chosen voice, it is drawn on the screen, and POS is raised slightly. A check is then made, whether or not the bar line is in the chosen voice, to see if the double bar line flag DBL2,3 is set. If it is, C1,3 is incremented and a return is made to the display loop. If DBL2,3 is not set, the arranged bar line pointer, Q,3 is incremented. Q,3 will now either point to a spot where the address of the first note of the next bar is contained, and in this case, this address is loaded into C1,3 and a return is made, or Q,3 will indicate that a double bar line or the end of the melody has been found. In this case C1,3 is simply incremented by one before the return is made.

To reduce the gaps in the music caused by several messages in a row, the message routine deals with the actual commands very differently than DOLY. The 2nd to 6th TEMPO and MODIFY commands in a row are written directly beneath the first one so it is possible to have a stack of 6 commands written in a vertical row.

When a MODIFY command in a chosen voice is encountered, the program obtains the necessary alphanumeric data from the external message table, MEST. Similarly, if a TEMPO command is found, the alphanumeric value is computed using SPBTA and DELEZE. In both cases a check is made on V (the stacking up pointer). If V is 0, the MODIFY or TEMPO command is plotted right under the staff with no change in X position. After the command is plotted, the X position is incremented by the value in NCR, V is set to -1, C1,3 is incremented by one, and a return is made to the Display Loop. If V was not 0, the command will have to be plotted under the last command. The Y position is decremented by YNME once for every command already in the stack. For example, if V = -3, 3 commands are already in the stack, and the Y position must be decremented by YNME 3 times, and the X position is decremented by NCR, before the message is plotted. If V has reached -6, it is reset to 0, otherwise it is decremented by one. NCR is again added to POS (X position) so POS is the same after this last plot as it was after the last message was plotted.

V is also set to 0 whenever a note, rest, bar line, marker or cursor note is found. Even in the messages, R is incremented before the plot of a chosen voice message, and if it reaches 0, V is set to 0 and the cursor is drawn. In this case the present E1,3 is stored in Y and Z is not set to -1.

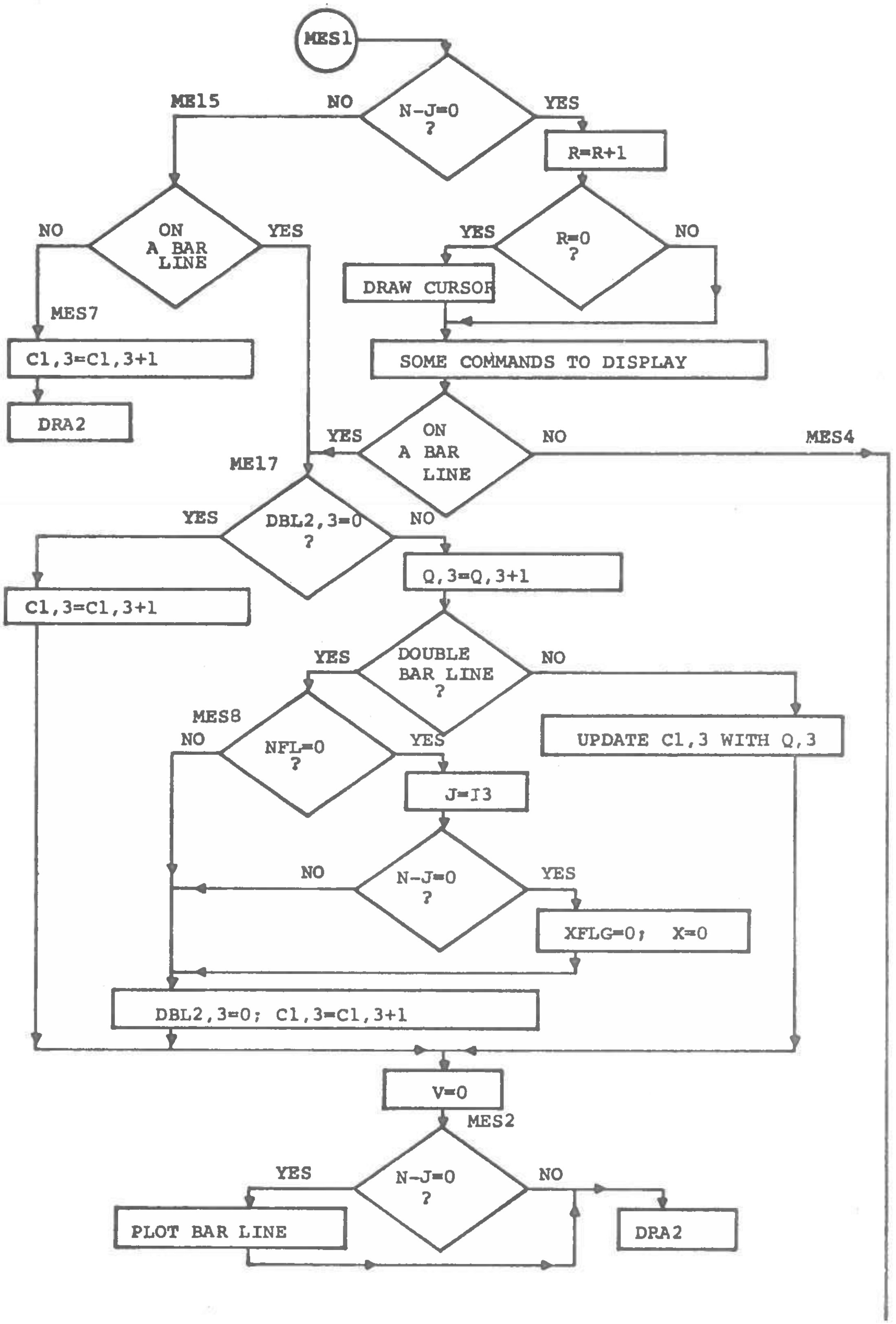


Figure 20. (a) Messages

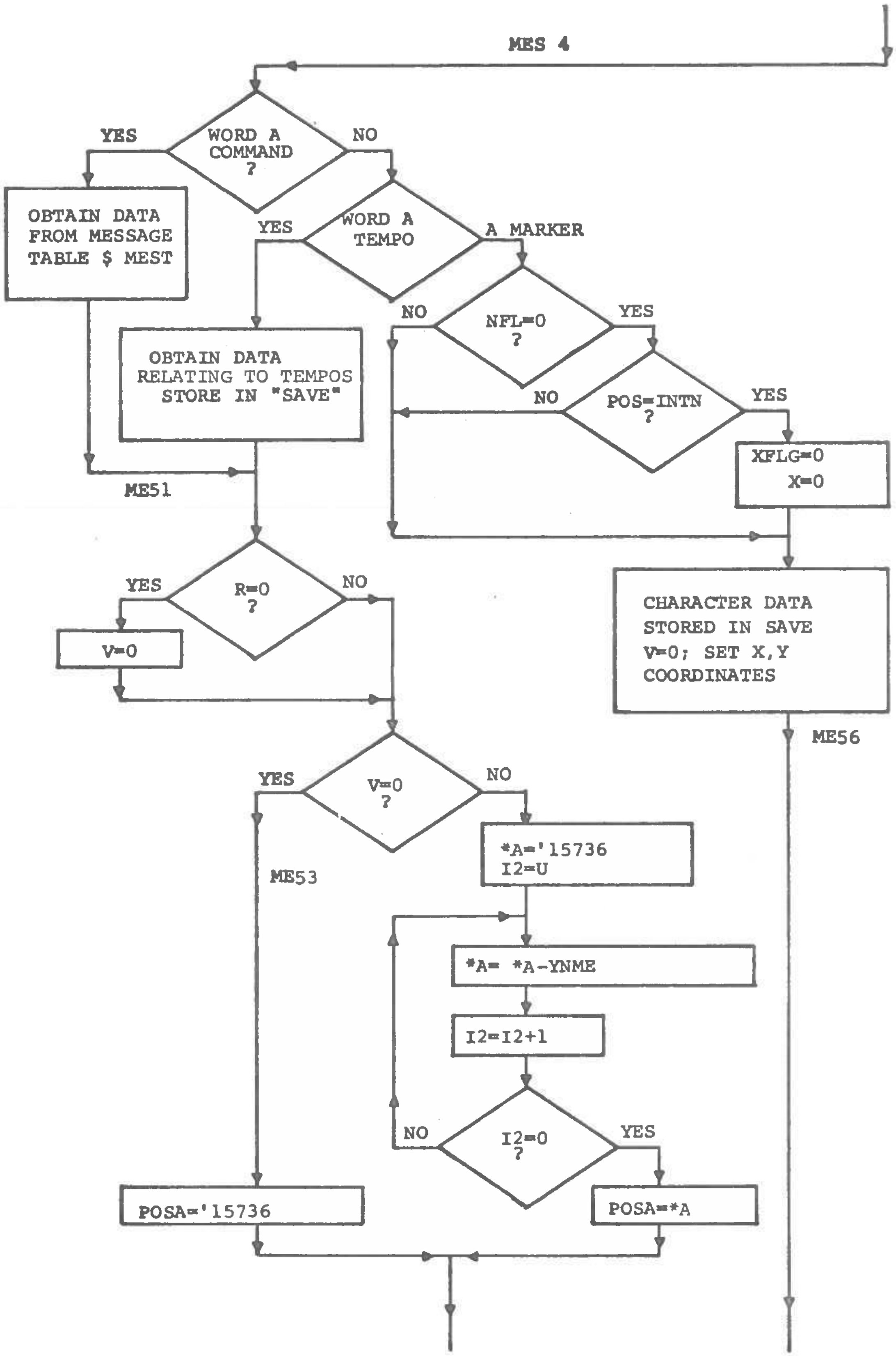


Figure 20. (b) Messages

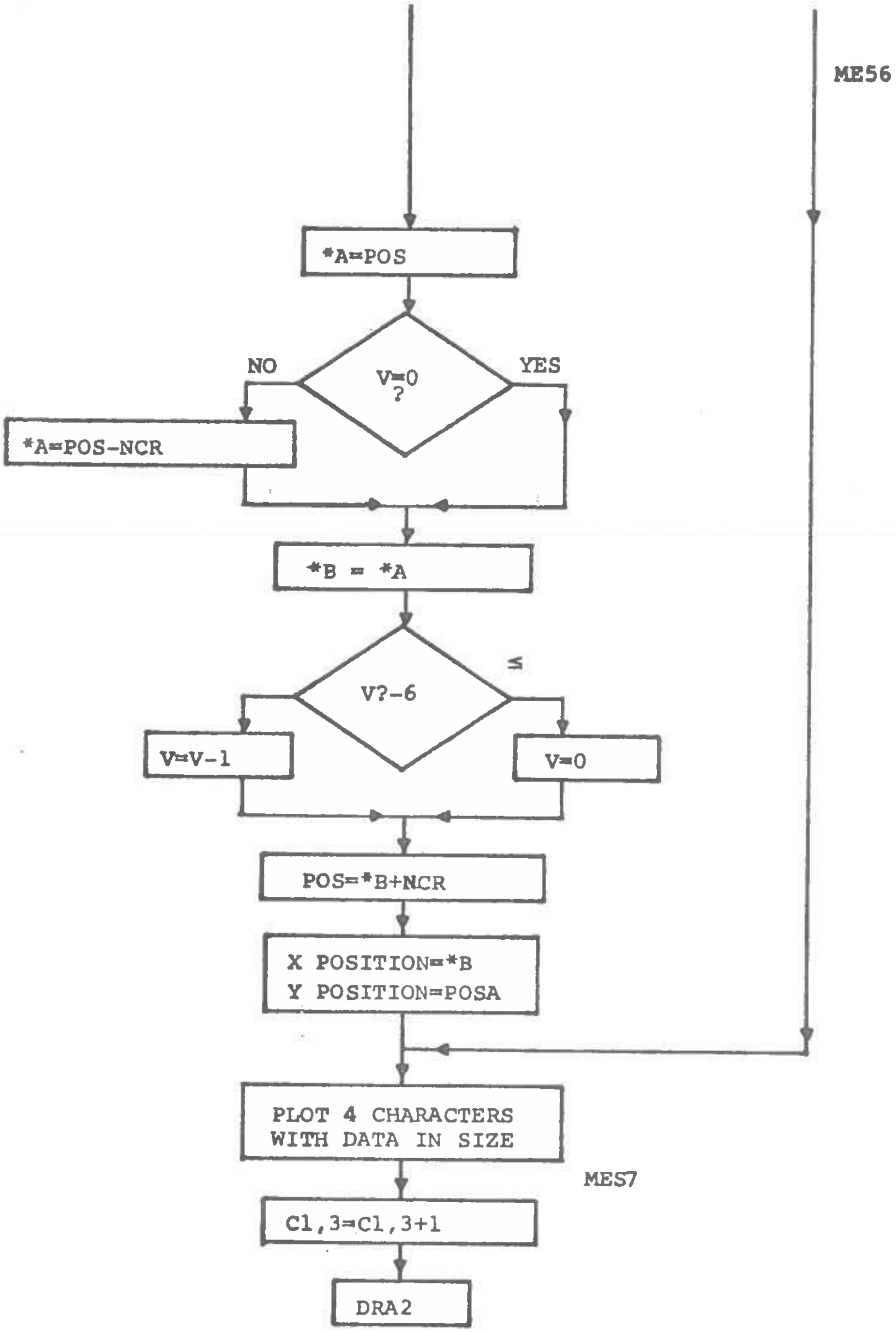


Figure 20. (c) Messages

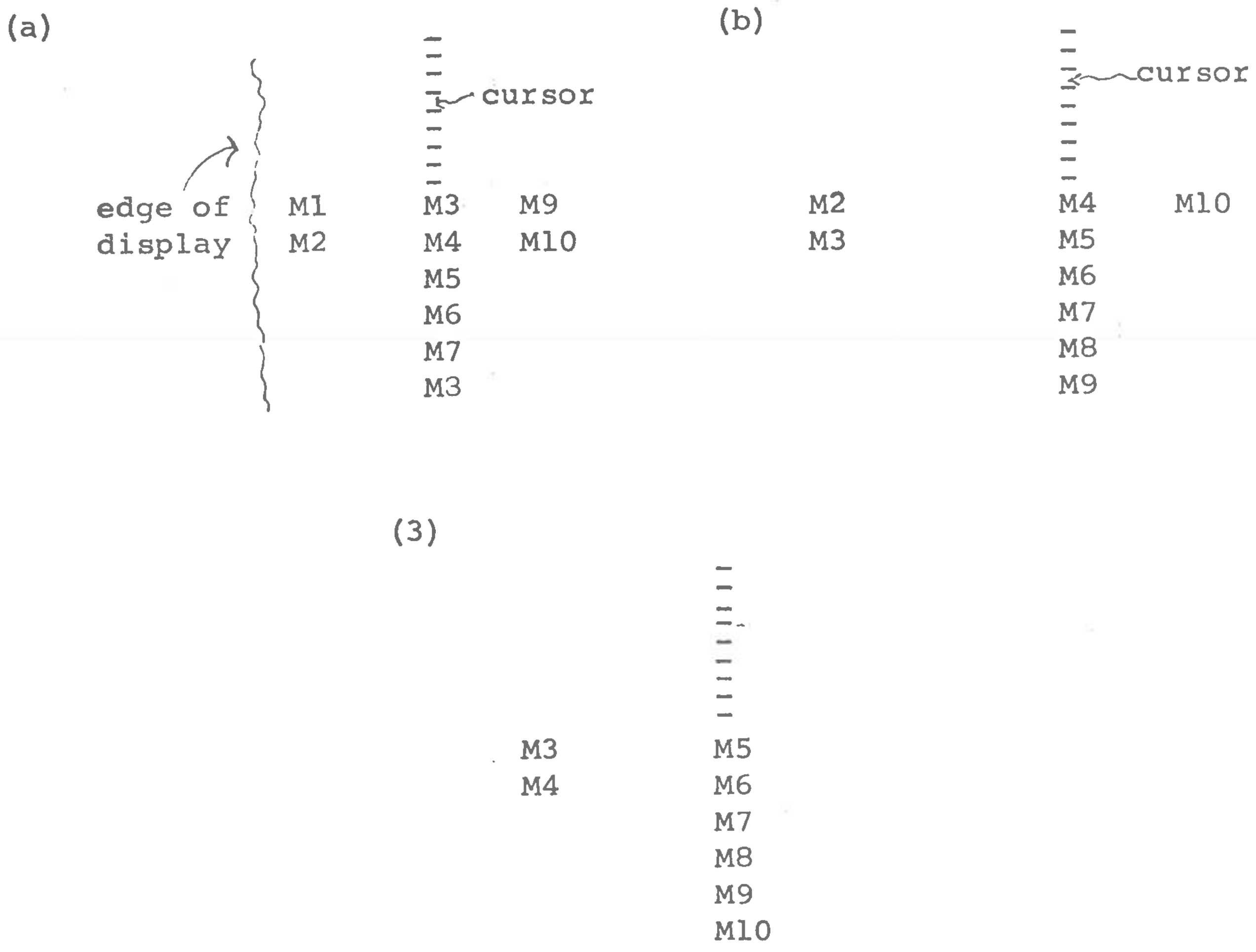


Figure 21. The cursor position lies on the 1st message directly below it. Each successive picture shows the result of the cursor moved one step forward.

The effect, though, of setting V to 0 for the cursor is that the cursor "points" to the top message in the stack. See Fig.

For marker messages, the letter is plotted, and V is set to D. If we are in LOOK and the X position of the marker is zero then all movement flags are deactivated and the marker is plotted.

INTERRUPT PROCESSING

When a button is pressed, an interrupt is enacted as in DOLY, which stores the contents of the keyboard register in BUTA and sets the keyboard flag, KFLG, to 0 and then returns. At the beginning of the Display Loop, just after the staff is drawn, a branch is made to the Input Processing routine. If KFLG has not been set to 0 since the last time the Input Processing routine was used, a branch is made to INPT, the second half of the routine.

If the keyboard flag has been set, there are two possible courses of action depending on whether or not LOOK was last chosen. If something other than LOOK was last chosen, then the buttons will be interpreted as a WRITE, INSERT, DELETE, MODIFY, TEMPO or MARKERS input command and the data in the voice is changed as it would be in DOLY. A branch is then made to INPT.

If LOOK had been the entry to the program, a check is made to see if any of the voice selection buttons are set. (Appendix B, Table 6). If they are, a negative integer would be stored in N, 3 for the chosen voice. If only one voice was chosen, N would be set to that voice, otherwise it is set to 0.

If none of the voice selector buttons had been chosen the palm and thumb buttons are checked. If the thumb button alone is pressed, forward motion is indicated, if the palm button alone is down, backward motion is indicated and if both buttons have been pressed, all motion is cancelled. X and XFLG are set to correspond with the choice selected. N is then checked, and if it is 0, more than one voice is being displayed, and the motion is cancelled so X and XFLG are set to 0. All cases are followed by a branch to INPT.

INPT, the second part of the Input Processing routine, first checks the value of X. If X is -1, a branch is made to the Going Forward routine and if X=+1, a branch is made to the Going Backward routine. If X = 0, the shaft encoder word is checked. The Y position is stored in POSB for use by the cursor drawing routine as the Y position of the dot and a translated version of the Y position is stored in MWRD to be used as the pitch of the dot. The horizontal wheel position is loaded into ENC and if it differs from the old position, (ENCO), a branch is made to the Going Forward routine or the Going Backward routine, depending on which way the wheel was turned. If ENC = ENCO, a branch is made to start the display loop.

GOING FORWARD

The purpose of the block of instructions labelled Going Forward is to find the first word that is displayed on the screen and, by advancing the voice pointers, move the displayed segment of music forward so that this note is no longer in this segment.

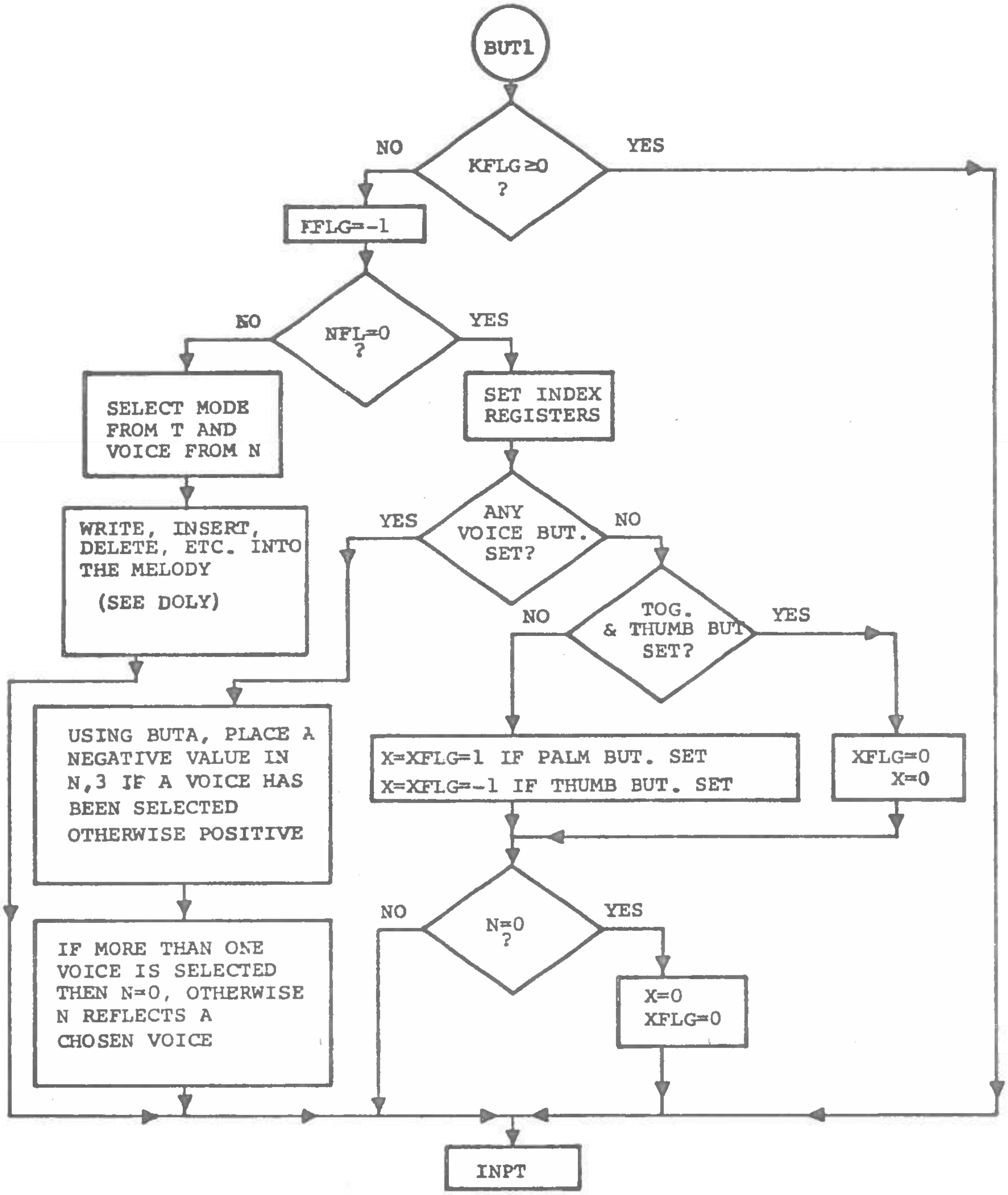

BUT1
KFLG ≥0 ?
NO
YES
FFLG=-1
NFL=0 ?
NO
YES
SELECT MODE FROM T AND VOICE FROM N
SET INDEX REGISTERS
WRITE, INSERT, DELETE, ETC. INTO THE MELODY (SEE DOLY)
ANY VOICE BUT. SET?
YES
NO
TOG. & THUMB BUT SET?
NO
YES
USING BUTA, PLACE A NEGATIVE VALUE IN N,3 IF A VOICE HAS BEEN SELECTED OTHERWISE POSITIVE
X=XFLG=1 IF PALM BUT. SET
X=XFLG=-1 IF THUMB BUT. SET
XFLG=0
X=0
N=0 ?
NO
YES
IF MORE THAN ONE VOICE IS SELECTED THEN N=0, OTHERWISE N REFLECTS A CHOSEN VOICE
X=0
XFLG=0
INPT

Figure 22. Interrupt Processing

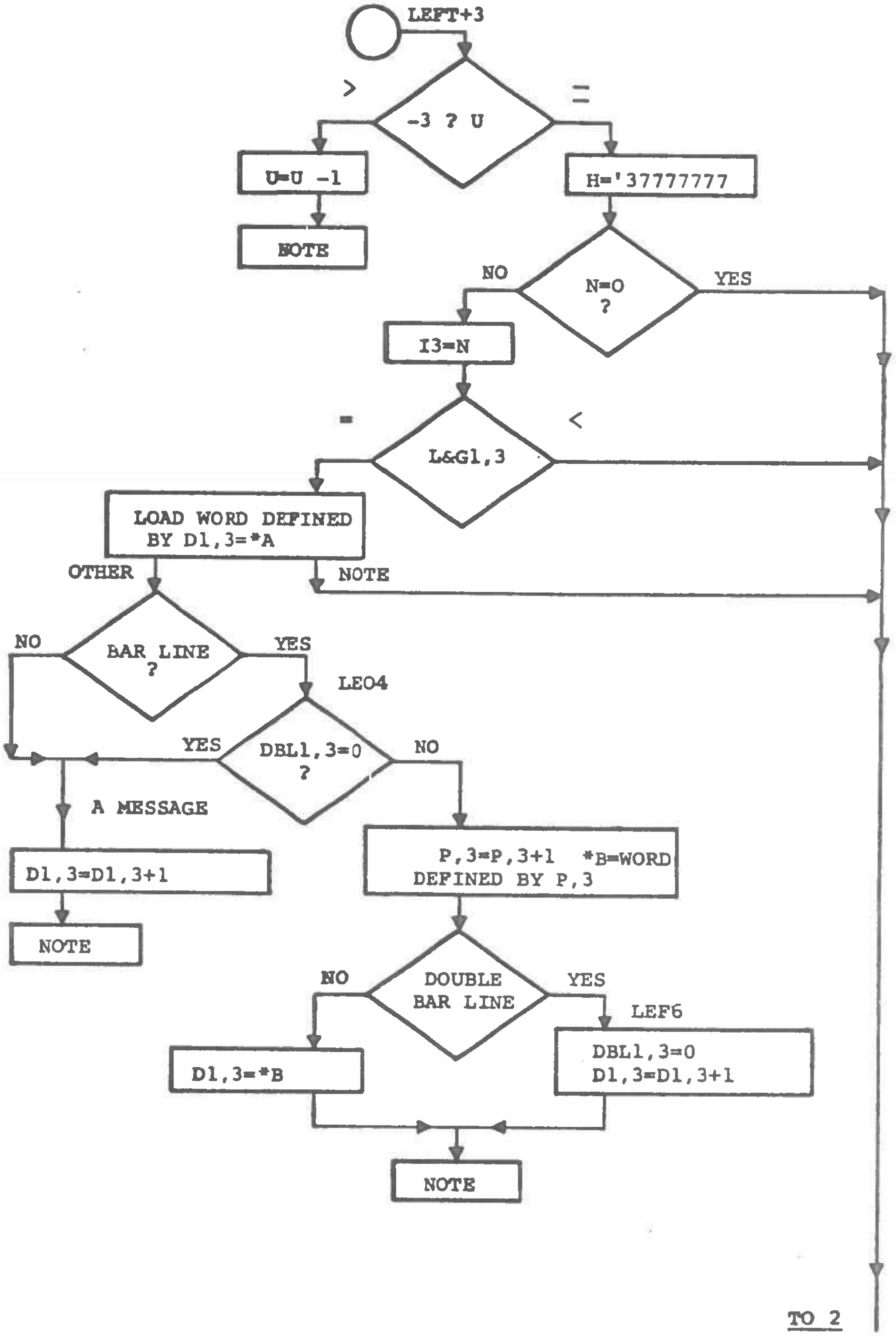


Figure 23. (a) GOING FORWARD

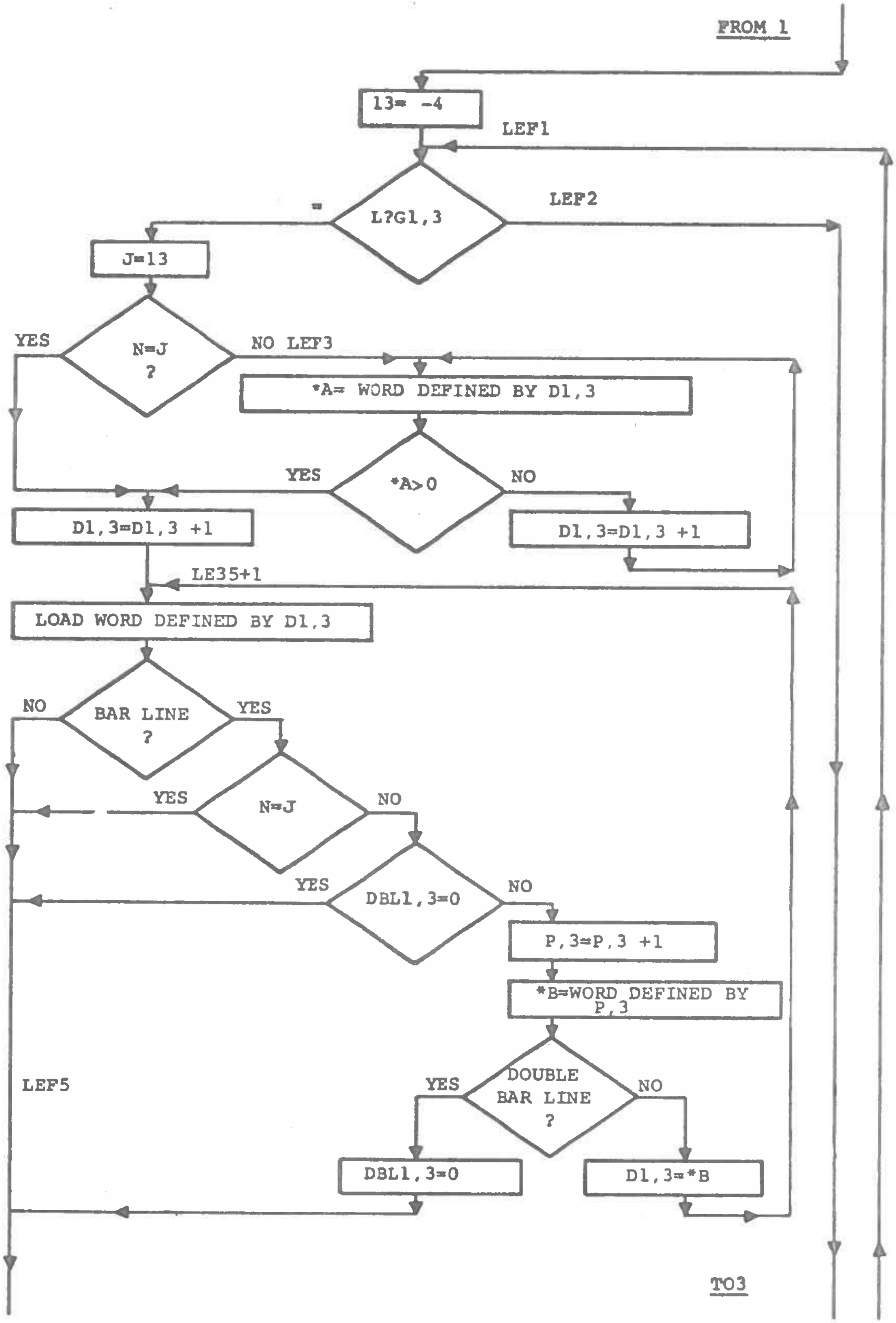


Figure 23. (b) GOING FORWARD

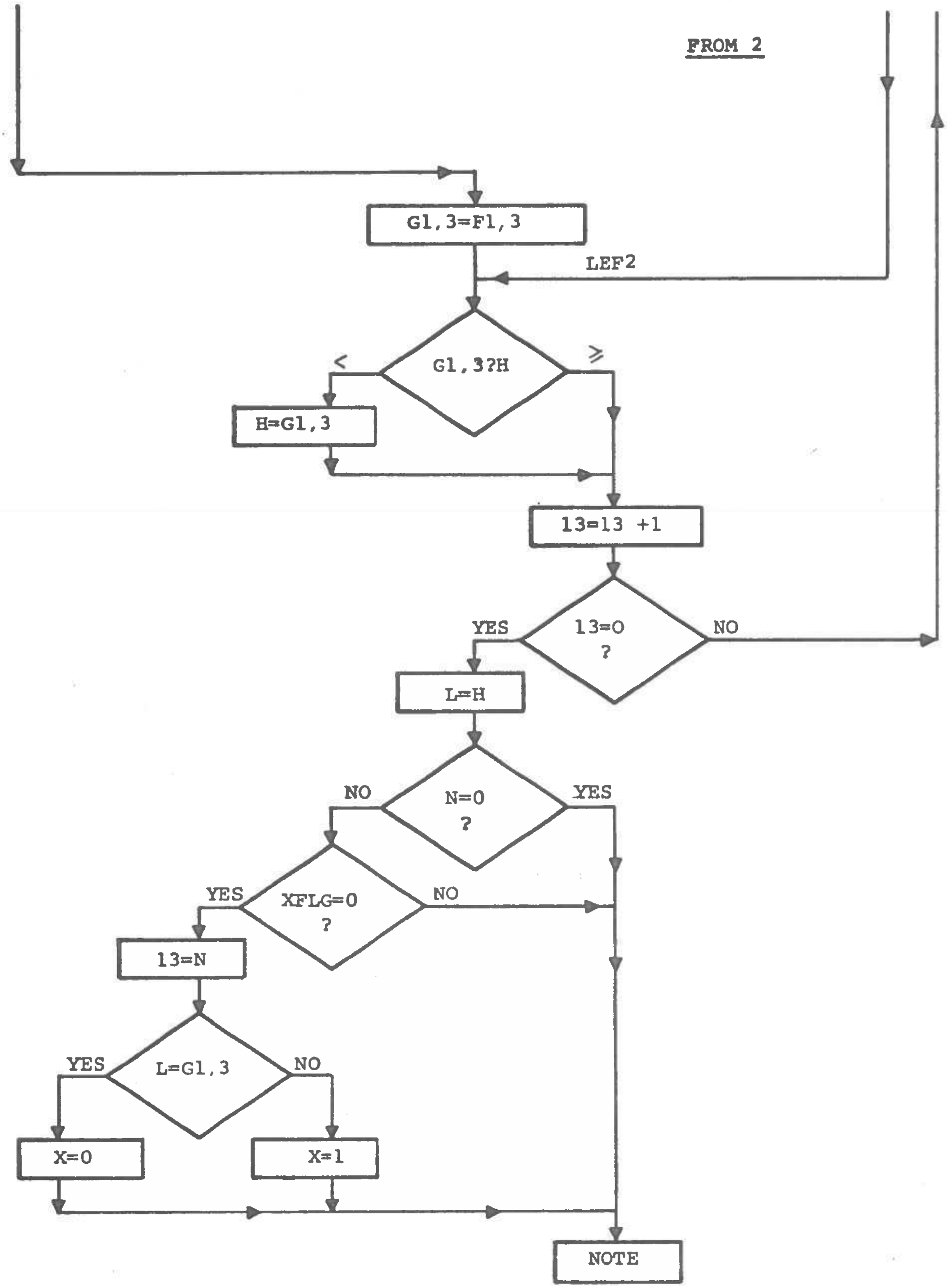


Figure 23. (c) GOING FORWARD

Following through the actual programming, the first thing the routine does is to check which note the cursor is on. If it is not on the third note yet, it is moved forward and a return is made to the display loop.

Next a check is made to see if the chosen voice contains a message or a bar line at the beginning of the display. If there is a message, D1,3 is incremented and a return is made. If there is a bar line, DBL1,3 must be checked to see if a double bar line has been passed. If it has,D1,3 is incremented and a return is made. If the double bar line is yet to come, P,3 is incremented, and if this does not indicate a double bar line, the contents of the address in P,3 are stored in D1,3 before the return. If a double bar line has just been reached, DBL1,3 is set to 0, D1,3 is incremented and a return is made.

If neither of the aforementioned possibilities were the case, a search has to be made for the words at the beginning of the display. These are the words with the lowest G1,3's and these G1,3's = L. Each voice is checked in turn, and if it is found that G1,3 ≠ L that voice is not touched, and the next voice is checked. When a voice is found that has a G1,3 equal to L, the voice is checked to see if it was the chosen voice. If it is not, D1,3 is checked to see if it points to a note or rest. If it does, D1,3 is incremented, F1,3 is stored in G1,3 and the next voice is checked. If D1,3 points to a bar line, Q,3 is incremented and D1,3 is set to the contents of Q,3 if the double bar line flag is not set, or if a double bar line is encountered, D1,3 is simply incremented and the double bar line flag is set. If the double bar line flag was set to begin with, D1,3 is simply incremented. If D1,3 points to a command or marker, it is incremented. In both cases, D1,3 pointing to a bar line, or D1,3 pointing to a command or marker, a branch back is made so the new D1,3 will be checked. In this way, D1,3 will be incremented until it has been incremented past a note or rest. F1,3 is then stored in G1,3 and the next voice is checked.

If G1,3 = L for the chosen voice, D1,3 is simply incremented by one, F1,3 is stored in G1,3 and on to the next voice.

When all the voices have been checked the smallest G1,3 is found and stored in L. If the drift mode is on, XFLG≠0, a return is made to the Display Loop. Otherwise, a check is made to see if the G1,3 of the chosen voice is equal to L. If so, a return is made to the Display Loop, and if not X is set to -1 before the return. This -1 in X will cause a branch to the Going Forward routine after the next time the staff is drawn. This will cause the music to keep going forward until a note from the chosen voice is at the left edge of screen.

GOING BACKWARDS

When the computer is moving back through the music, the notes before the one on the screen have to be checked. The first such check is in the chosen voice by decrementing D1,3 and seeing if it points to a message. If it does, a return is made to the display loop, and if it does not, D1,3 is incremented back to its original value.

As E1,3 and C1,3 are not being used at the moment, and will be reset at the beginning of the next Display Loop, the present values of D1,3 and P,3 are stored in them so they may be altered without losing the original values of D1,3 and P,3.

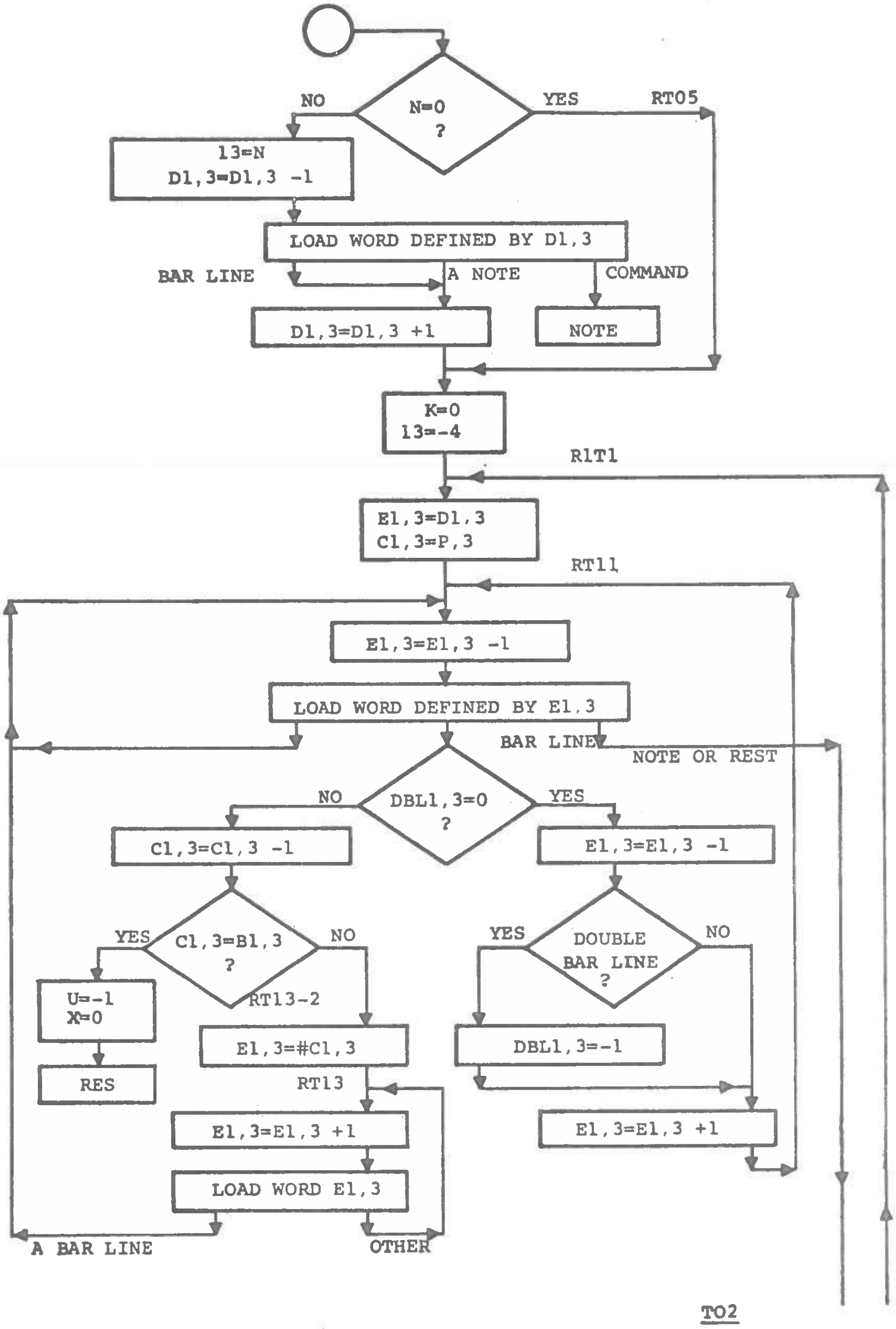


Figure 24. (a) GOING BACKWARD

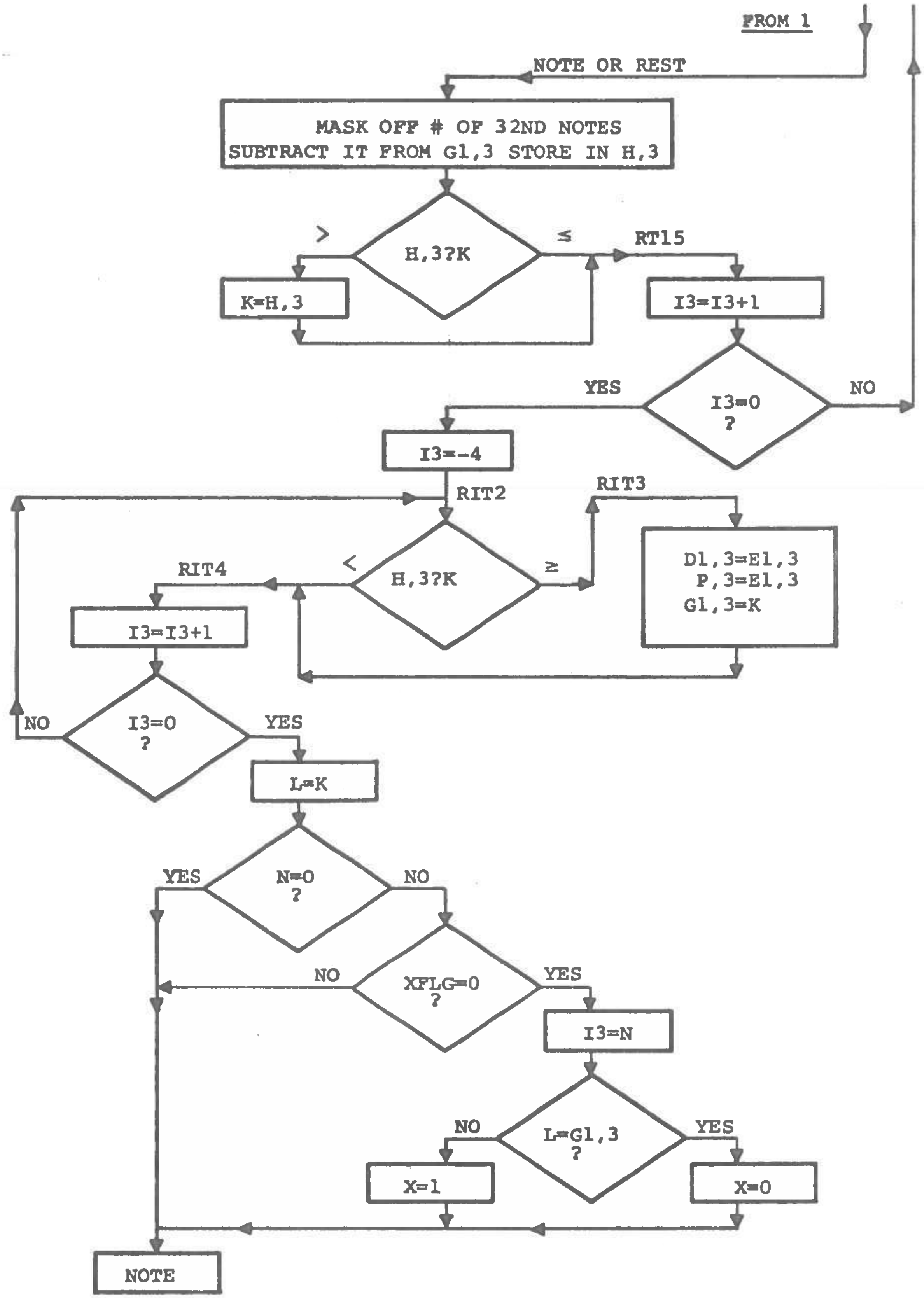


Figure 23. (b) GOING BACKWARD

The next segment of the routine is a loop which is repeated for each voice. This loop finds the note or rest previous to that displayed on the screen. E1, 3, which was set to equal D1, 3, is decremented and the word it points to is loaded. If this word is a command, a branch back is made to decrement E1, 3 again and load the previous word. If the word is a bar line and the double bar line flag is not set, C1, 3 is decremented to give the address of the first note of the last bar. If the 1st bar was already being displayed, the cursor position is decremented and a branch is made to reset all the pointers. If the first bar is not yet being displayed, the bar before the first one on the screen is scanned through to find its end and the address of the bar line at the end of the bar is stored in E1, 3 and again a branch is made to decrement E1, 3 to check the word before the bar line. If the double bar line has been set, and an E1, 3 pointed to a bar line, the word previous to the bar line is checked. If it is not a bar line, a branch back is taken so the word will be checked as the other words were, but, if there are two bar lines in a row, the double bar line flag is put back to -1, and a branch back is made so that the first bar line will be treated as a normal bar line. E1, 3 will be continually decremented in the above manner until a note or rest is found. When one is found, H, 3 is set to G1, 3 minus the duration of this note or rest. The largest H, 3 is stored in K.

When all four voices have been checked, K contains the largest H, 3, and any voice where H, 3 equals K will be one that is changed. This change will store the E1, 3 that was used in the routine in P1, 3 and C1, 3 will be stored in P, 3. K will contain the number of 32nd notes up to but not including the first note displayed and therefore will be stored in both L and the G1, 3's of the changed voices.

As in Going Forward this routine finishes off by setting the flag X if a note from the chosen voice, if there is a chosen voice, is not the first note to be displayed. This flag will cause a call to Going Backwards each time the Input Processing routine is called.

VII. SYSTEMS PROGRAMS

The programs in this chapter are those that are loaded with the music system. They are, on the whole, very short simple programs and can be described in a few sentences each. The descriptions of the programs are in the order that the programs are stored in the computer. The actual memory locations of the programs are given in Appendix A.

KLOK

This subroutine increments index register 2, and sets IFLG to D when register 2 becomes 0. It is used for 1ST PLAY, 6TH PLAY, 7TH PLAY, 8TH PLAY, and 10THPLAY.

LOOK

LOOK contains the NEXTPAGE list of choices (see Chapter 4) and either an external branch, or the actual programming for each choice. The four choices that deal with inverses and transposes are in LOOK. If one of these is chosen, two flags are set up. CFLG is set to -1 if a diatonic invert or transpose is chosen, and is set positive if a chromatic invert or transpose is chosen. FLG is set negative in the case of an invert, and positive in the case of a transpose.

The number of semitones of the requested transpose is then loaded into KOW, or in the case of an invert, the number of semitones that the chosen note is above low C is stored in KOW. Then if CFLG is positive, the pitch is loaded for each note and this pitch is used to load the number of semitones that the present pitch is above low C. This value, the number of semitones, is stored in a table. A calculation is then performed to obtain the number of semitones that the transposed or inverted note is above low C, and this value is used to obtain the new pitch value from a second table. The idea behind the diatonic operation is the same, only different tables are used so that the notes end up on members of the diatonic scale. This process is repeated for every note in the melody.

LOOK also contains a couple of reverse routines, one reverses the first '2000 words of the envelope (REV ENV) and the other reverses the melody (RETROGRD). Both of these load the beginning and ending addresses of the data table to be reversed, (the address of the double bar line in the case of RETROGRD) and then branch to a subroutine that does the actual interchanging.

SPDTA, DELEZE

These are two general systems programs (as opposed to music system programs). SPDTA changes numbers from binary to decimal and DELEZE (Reference 9) deletes leading zeroes. For example, if a tempo command of '00000113 was to be displayed on the music SPDTA would change the number to the decimal characters 0075, and DELEZE would delete the zeroes to make it 75.

GETTER

As mentioned before, data sets of different types are stored on disc. GETTER is the little program that is called when GET is chosen from the main list of choices. It enables the user to choose a data set and will then check the code of the chosen data set, and load it into the memory location that corresponds with the code (see Appendix E). If a melody has been loaded, a call is made to PATCH2 in DOLY to sort out the bar line tables for each voice. If envelope parameters were loaded GENTAB the subroutine which generates the actual envelope from the parameters is called. After all the necessary calls are made the data set is executed if it is a program, otherwise a branch is made to the sound generator. To fully understand GETTER and the next program, DUMPER, it is necessary to read and understand the program description of D$STOR Reference (2)

DUMPER

Choosing PUT AWAY from the main list of choices enacts the program DUMPER. This program then presents the user with a list of choices, each choice being a different type of data set that can be stored on the disc. After a choice is made, the data set is stored with an appropriate code (see Appendix E).

In the case of the melodies, the data set that is stored ends at the first double bar line. For example, if the user chose to store 2 VOICES, the data set that is stored would extend from '14457 (the beginning of the melody) up to and including the first double bar line in VOICE 2, or if there were no double bar lines, the melody would be stored up to the end of VOICE 2 ('17204).

RHYTHM

The percussion is controlled by the contents of the first eight bits of a word output. Each bit represents a percussion instrument, and if the bit is one, the instrument is sounded. There is a table which has a percussion control word for every natural pitch value, i.e., if the percussion is on, and middle C is played, the word corresponding to middle C (which happens to sound the bass and tambourine) is loaded, and a call is loaded to RHYTHM. RHYTHM outputs the word to the percussion generator, waits about 700 microseconds, clears the percussion generator, and returns control to the sound generator.

ARRAN

ARRAN is given control when ARRANGE is chosen from the main list of choices. It takes the address in THUD (the address of the top of the arrangement table), subtracts 79 (the length of the table), and then calls I$DECR Reference (9),which allows the typing of numbers into the arrangement table. A call is then made to DAMEL and TRAN (see below) and the control is returned to the sound generator.

I$DECR(7)

This is a general systems program. It translates ASCII decimal character input from the typewriter into binary numbers and stores it in the arrangement table. All non-numeric characters are eliminated but serve as numerical delimiters.

DAMEL/TRAN

The arrangement table consists of the numbers typed in after ARRANGE has been chosen. DAMEL is called to make another temporary table (PTBL) which contains the addresses of the first note of each bar in sequential order.

To do this, DAMEL first enters the address of the first note in PTBL, then it scans through the music looking for bar lines. When the first bar line is found, the address of the note after it is stored in PTBL+1, the address of the note after the second bar line is stored in PTBL+2 and so on. When a double bar line or the end of the music is found, the word '40000000 is stored in the table to indicate the end of the table. A call to TRAN always follows the call to DAMEL. TRAN uses the arrangement table and DAMEL's sequential order bar line table to rewrite the bar line table that is used by the sound generators. This table contains the addresses of the first note of the bars in arranged order, that is, the order that was typed in under the ARRANGE choice. TRAN takes each element of the arrangement table, uses it to find the corresponding element of the temporary table, and stores it in the final bar line table. For example, if the arrangement table contained the numbers 1 3 4 2 7 0, TRAN would load the address in PTBL (the first note of the first bar) and store it in the first member of the bar line table, the address from PTBL+2 which is the address of the first note of the third bar would be stored in the second member of the bar line table, PTBL+3 in the third member, then PTBL+1, PTBL+6 and finally PTBL-1 which is the number '40000000 and indicates the end of the melody. If the melody had had a double bar line after the third bar, PTBL+3 would have had a '40000000 which would have been stored in the third member of the bar line table to indicate the end of the piece. ARRAN, DAMEL, and TRAN only work on the current voice. They all have memory locations which store outside addresses, like the addresses of the bar line table and the starting address of the music, and these memory locations are externally referenced so that they may be changed when a different voice is chosen.

FORM

FORM, FORM1 and FORM2 are the entries to the program that display the 3 waveforms and accept modifications to them from the tablet. The three entries are for the three waveforms. The program is in three segments, one segment displays the waveform, the second changes the data in the waveform if the tablet is being used, and the third clears the waveform by storing - '40000000, in every point of the waveform.

The first segment uses an index register to load successive points of the waveform to display them. Until it is interrupted, it is a continuous loop. When the tablet pen is placed on the tablet, an interrupt activates the second segment of the program. An input word from the tablet is read, and if the clear button has been pushed, this word will be negative, and a branch will be made to the 3rd segment of the program.

If the input word is positive the Y position of the tablet pen is stored in a word in the waveform table, the X position of the tablet deciding which word is the table. The last or '2000th word of the waveform table has the greatest X which is '2000. A branch is then made back to the first segment of the program.

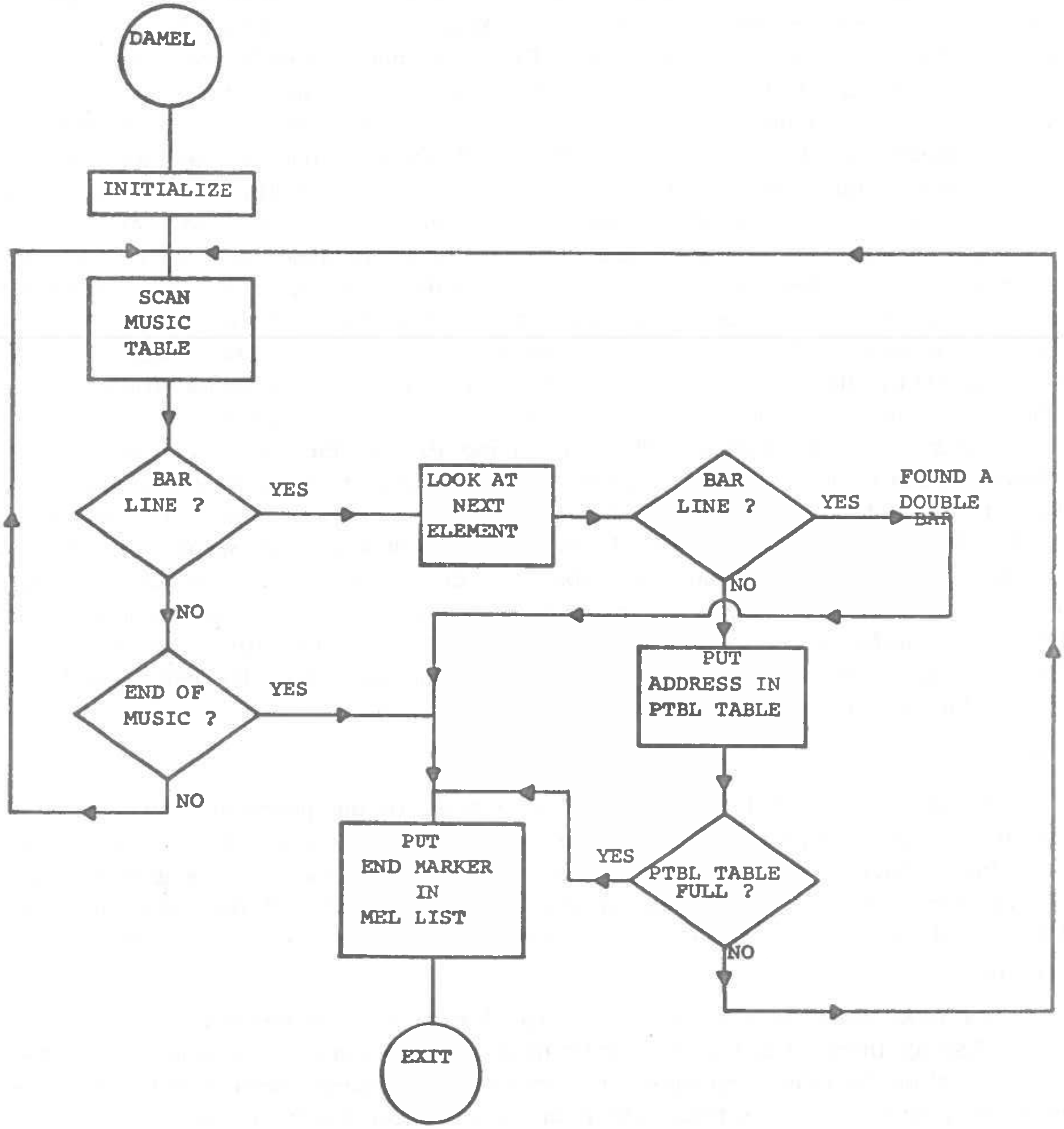


Figure 25. DAMEL

DYNA

DYNA is the program that generates, modifies and controls the envelope. It has three entries, all from the NEXTPAGE list of choices, SEE ENV which displays the envelope and allows modification, GENTAB which generates the envelopes from the parameters and then branches to SEE ENV and TRANSFRM which combines WAV1 with the envelope, and then branches to SEE ENV. DYNA also contains 2 tables, NENVT which is the table of the seven envelope parameters, and ENVT which is the '5000 word long envelope table.

The principle behind DYNA is to allow the user to create a smooth curve to control the amplitude of the individual notes. Elaborate envelopes could be drawn into WAV2 and copied into the envelope table using COPY, but these curves are normally too rough. DYNA, though, allows 3 smooth curves, an exponential attack, and two exponential decays to be entered into the envelope. The rate of these attacks and decays are parameters, and as such, can be changed by the user. The other parameters that are user-controlled are the initial amplitude of the envelope (starting level), and the beginning addresses of the attack and the decays.

On entry to DYNA, the red button command is changed so that pushing the red button will cause a branch to INTP in DYNA. The user is then presented with a list of choices including the word GET, and the six envelopes. There is a table of envelope parameters associated with each envelope. Choosing one of the envelopes, moves these parameters into the main section of the program so that they may be modified. A branch is then made to the display routine. If GET was chosen, no change takes place in the parameters in the program, so GET is a direct branch to the display routine.

Later, when the red button is pushed, the parameters in the main section of the program are restored in the parameter table of the last chosen envelope. The original red button command is then executed.

As in FORM, a segment of the program displays the first '2000 words of the table on the CRT. After each of the '2000 points has been plotted, a check is made to see if the left foot pedal is depressed. If it is not, a return is made to display the envelope again.

If the foot pedal is down, a subroutine is called. This subroutine checks the shaft position encoder (the horizontal wheel) and if it has been changed significantly, a small addition or subtraction is made to the parameter that was last chosen. (The buttons are used to choose which parameter is to be modified).

As one of the parameters may have been changed, a subroutine GEN has to be called before a return can be made to the display segment of the program. GEN takes the starting amplitude (S) and stores it in successive points of the envelope until it reaches the beginning point of the attack curve (D1) or one of the decay curves (D2) or (D3). If D1 was reached first, the amplitude that is stored in the next set of points is calculated by multiplying the difference between the present amplitude, CAMP, and the maximum possible amplitude, by the attack rate K1 which is a number just short of 1. The result is subtracted from the maximum amplitude and stored in CAMP. This is repeated until the beginning address of the first or second decay is reached (D2 or D3). If D2 is reached first, the amplitude is multiplied by the first decay rate, K2, another number just short of 1. This exponential reduction of the curve continues until the

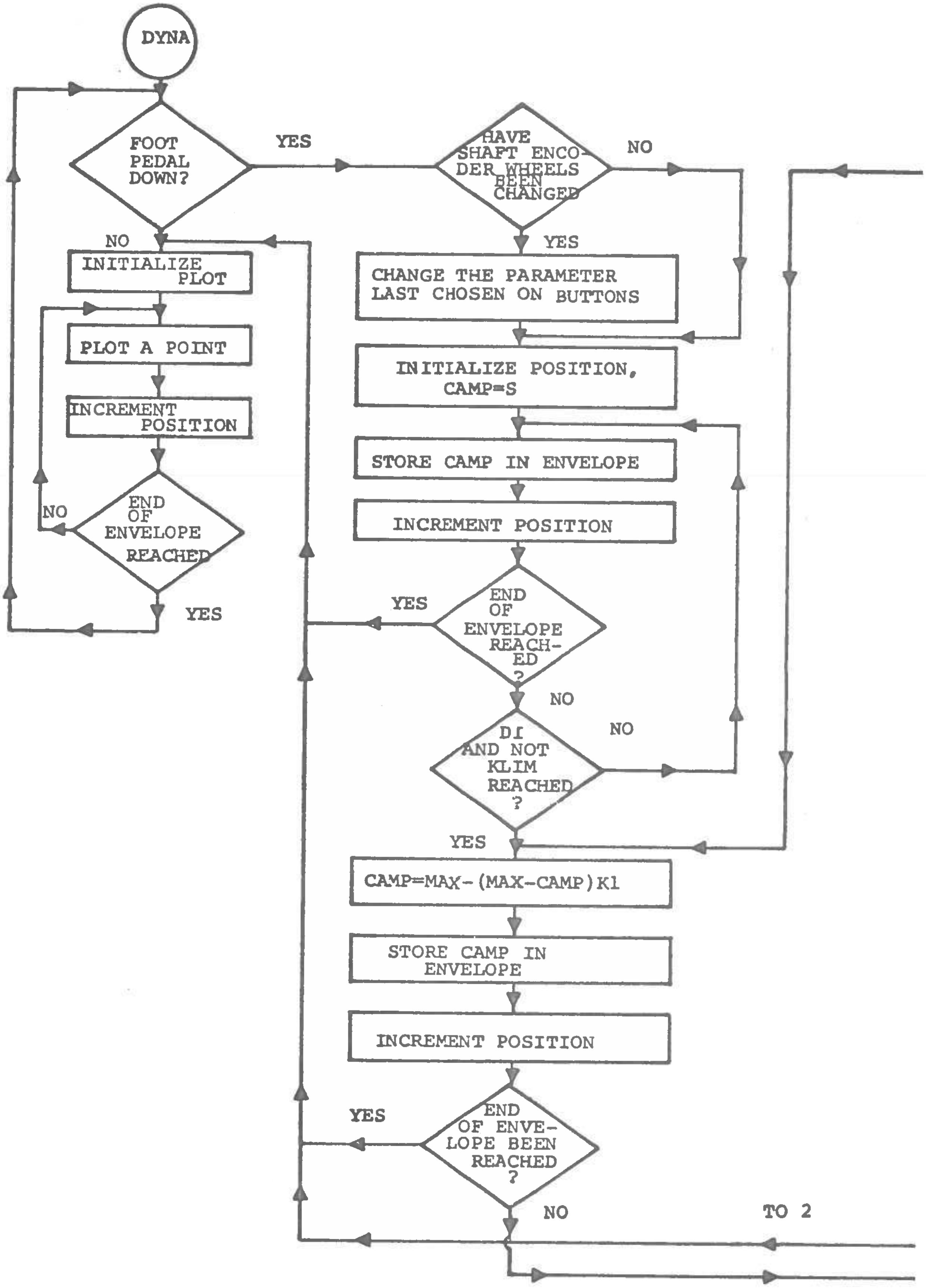
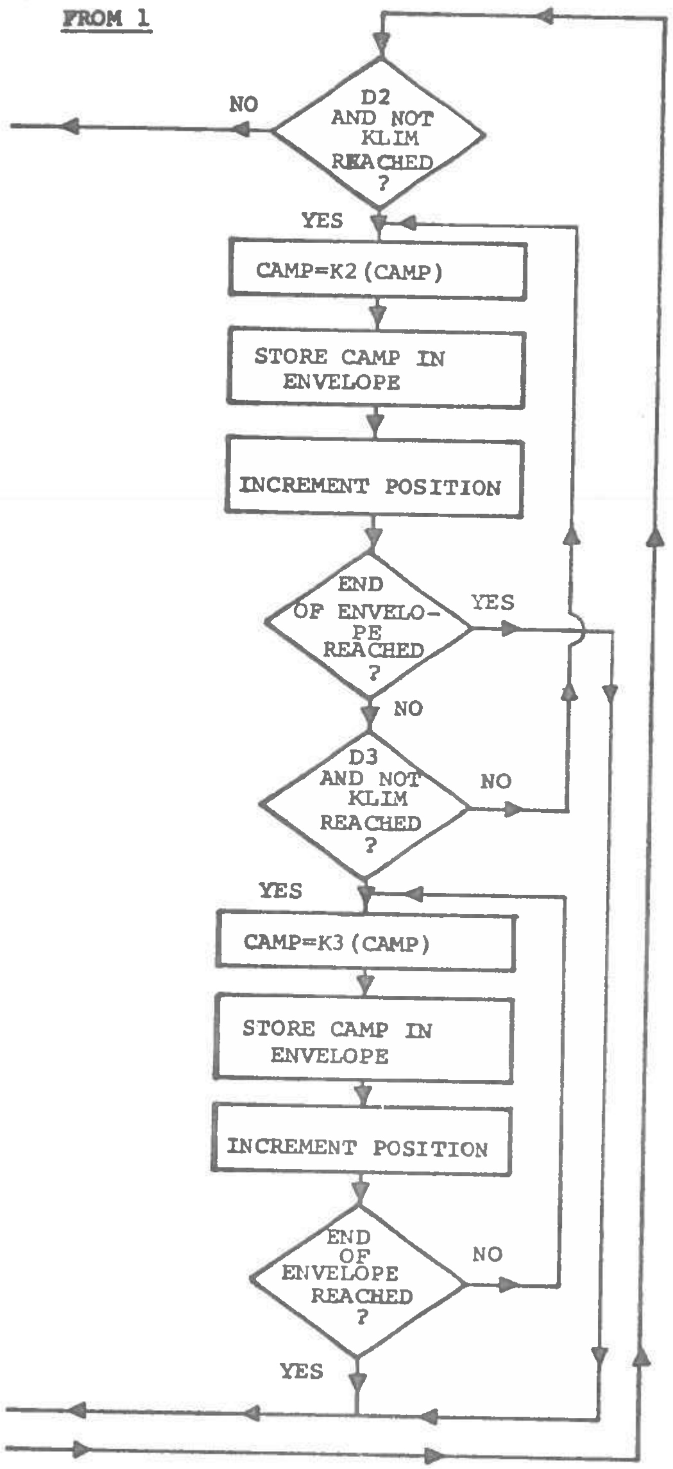


Figure 26. (a) DYNA Envelope Control



S - STARTING AMPLITUDE

CAMP - CURRENT AMPLITUDE

K1 - ATTACK RATE

K2 - FIRST DECAY RATE

K3 - SECOND DECAY RATE

D1 - BEGINNING ADDRESS OF ATTACK

D2 - BEGINNING ADDRESS OF FIRST DECAY

D3 - BEGINNING ADDRESS OF SECOND DECAY

KLIM - ADDRESS OF '2000th WORD OF THE ENVELOPE.

NB - D1 AND NOT KLIM REACHED? MEANS THAT IF D1 HAS BEEN REACHED, BUT KLIM HAS NOT BEEN REACHED, THEN TAKE THE YES ARROW

Figure 26. (b) DYNA Envelope Control

beginning address of the second decay, D3, is found and then the second decay rate K3 is used. This causes the envelope table to be filled with the appropriate values. The flow diagram is for the specific case where the starting points for attack and decay curves are in the order D1, D2, D3.

When GEN reaches the '2000th word of the envelope, it deactivates any starting addresses that are left so that the last '3000 words do not have a change in attack or decay rate in them.

TRANSFRM takes every point of the envelope, and it with the corresponding point from WAV1.

GENTAB branches to the GEN subroutine before it starts displaying the envelope so it would overwrite any change made using TRNSFRM.

In the monophonic sound generators, the rate of progress through the envelope is controlled by a real time clock. A knob beside the CRT controls the frequency of this clock interrupt. If the clock interrupts, say '500 times during a quarter note, only the first '500 words of the envelope will be used when quarter notes are being played. If set switch 1 is up, the number of interrupts per 16th note is counted every time the envelope is displayed. This number is multiplied by 4 to give the number of interrupts per quarter note. An arrow is placed on the screen when the number of points plotted equals the number of interrupts per quarter note. This arrow points to the end of the envelope segment used when a quarter note is played. As calculating the position of the arrow takes time, and slows down the display rate, it can be turned on or off. It is on only when set switch 1 is up.

VIII. OVERLAYS

It was found that musicians desired more capabilities in the system than it was possible to store in the computer at one time. For this reason, a series of overlay programs were written. These are programs that are loaded on top of and replaces something else. For example, sound generators are all overlay programs; when a sound generator is loaded, it replaces the sound generator presently in the system.

Most overlay programs are loaded on top of the waveform tables. After these programs have been loaded, they can be used again without reloading by choosing BRANCH on the NEXTPAGE list of choices for a program beginning in WAVE1, or choosing copy for a program in WAVE 3. Appendix A contains a chart of all the overlay programs and where in memory they are loaded.

COPY

COPY was designed primarily to move data from one area to another area in core. COPY has its own list of choices, the first four of which are responsible for copying parts of the melody to the end of one of the voices. For example, if the cursor was left in the first bar of VOICE 1, and COPY ALL was chosen, the computer would search for a double bar line in VOICE 1, and when it is found, the address of the second bar would be stored in FROT. If a double bar line did not exist, the address of the first note of the voice would be stored in FROT. In any case, the address of the first note in the voice is also stored in FROM. After this initialization has been done, a list of the four VOICES is presented on the screen. If, say, VOICE 2 was chosen, VOICE 2 would be scanned for a double bar line. If there was one before the end of VOICE 2, the address of the second bar line is stored in DEND, if not, the address of the beginning of VOICE 2 is stored DEND. Now the computer has the beginning and ending address of VOICE 1, (FROM and FROT) and the address of the end of the present melody in VOICE 2 (DEND). The transfer subroutine is called, and a word of the melody is copied from FROM to DEND, FROM and DEND are incremented, FROM is checked to see if it is equal to or greater than FROT, DEND is checked to see if it is equal to or greater than the address of the end of VOICE 2, and if neither of these is the case, the next word is copied from address FROM to address DEND. When FROM does reach FROT or DEND reaches the last address of VOICE 2, a double bar line is placed at the end of VOICE 2. A call is then made to DAMEL and TRAN and a branch is made to the WRITER program.

The other 3 choices are all very similar. If COPY BAR had been chosen, the address of the beginning of the bar that cursor is in is stored in FROM, and the address of the end of the bar is stored in FROT. The rest is the same as COPY ALL. C TO END puts the address of the note where the cursor was left into FROM, and the address of the end of the piece is FROT (as in COPY ALL). Again the rest is the same as COPY ALL.

C ARRANG does everything that COPY ALL does, but it also copies the arrangement table from the initial voice to the chosen voice.

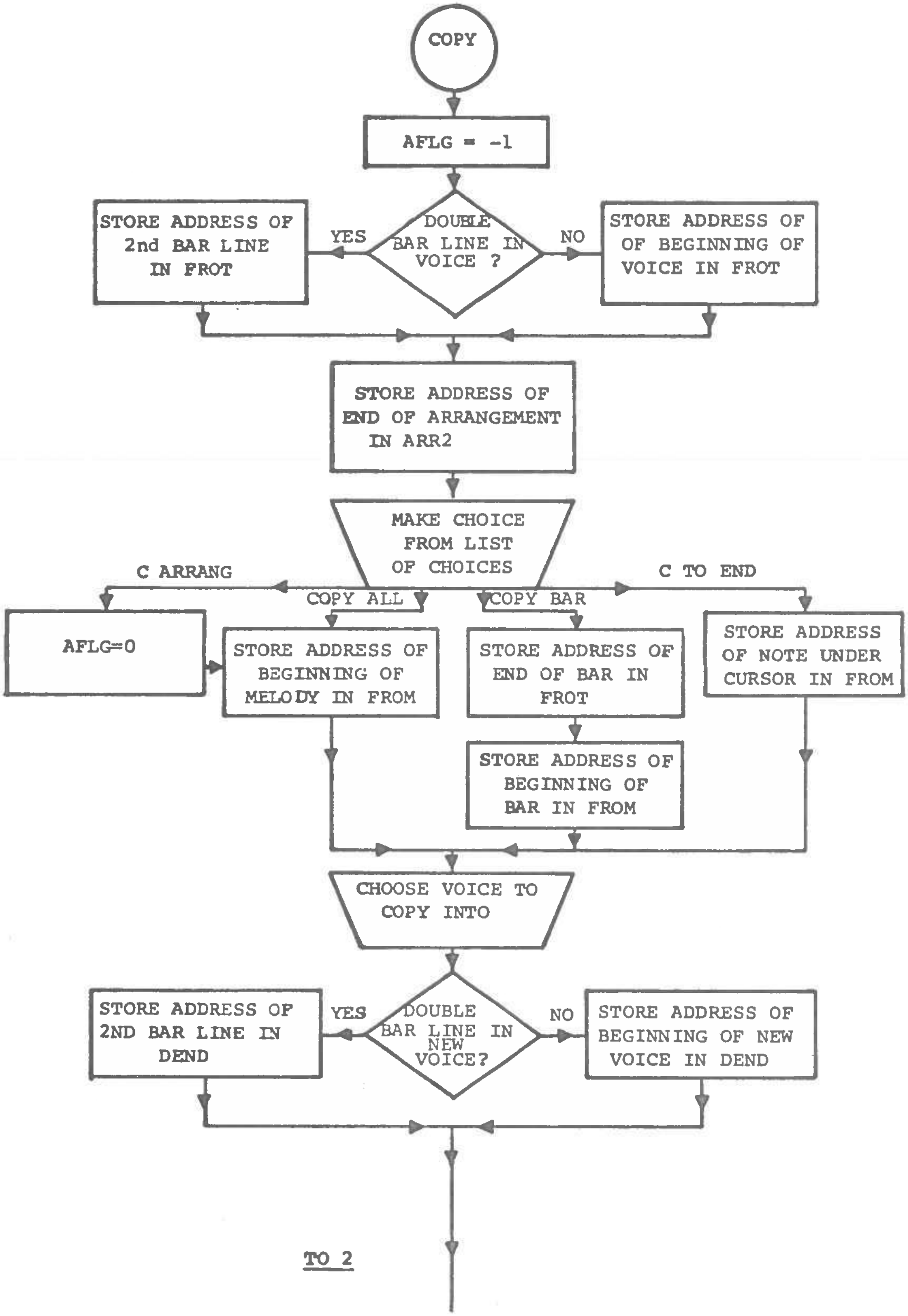

COPY
AFLG = -1
DOUBLE BAR LINE IN VOICE ?
YES
NO
STORE ADDRESS OF 2nd BAR LINE IN FROT
STORE ADDRESS OF OF BEGINNING OF VOICE IN FROT
STORE ADDRESS OF END OF ARRANGEMENT IN ARR2
MAKE CHOICE FROM LIST OF CHOICES
C ARRANG
COPY ALL
COPY BAR
C TO END
AFLG=0
STORE ADDRESS OF BEGINNING OF MELODY IN FROM
STORE ADDRESS OF END OF BAR IN FROT
STORE ADDRESS OF NOTE UNDER CURSOR IN FROM
STORE ADDRESS OF BEGINNING OF BAR IN FROM
CHOOSE VOICE TO COPY INTO
DOUBLE BAR LINE IN NEW VOICE?
YES
NO
STORE ADDRESS OF 2ND BAR LINE IN DEND
STORE ADDRESS OF BEGINNING OF NEW VOICE IN DEND
TO 2

Figure 27. (a) COPY

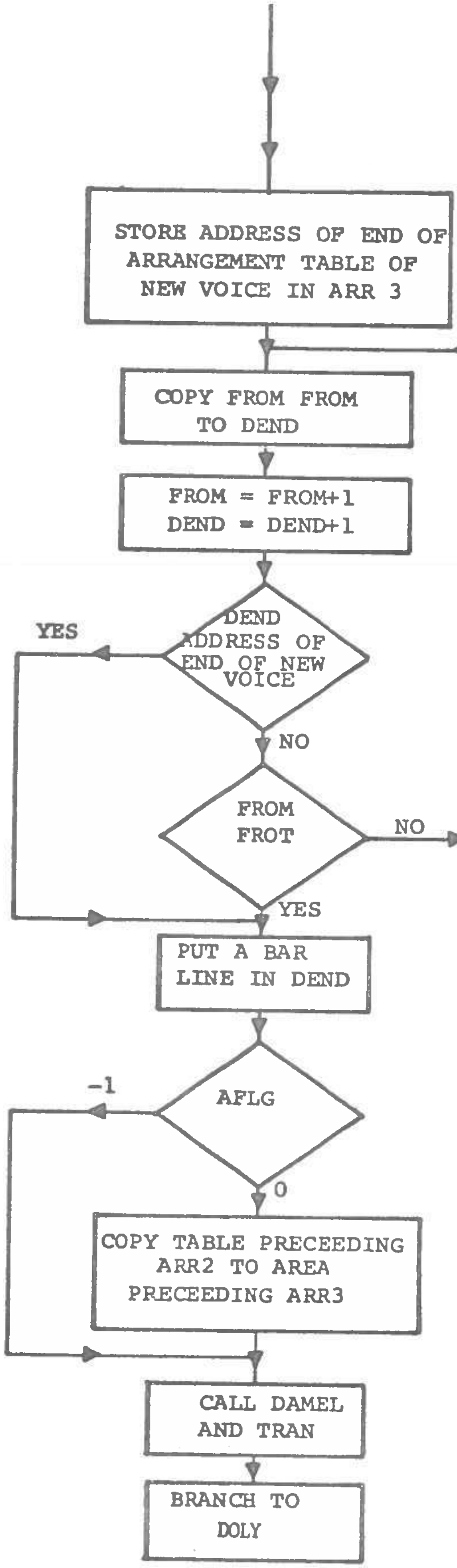


Figure 27. (b) COPY

Next on the COPY list are three choices that copy data from the waveform tables to the first '2000 words of the envelope table and back again. These choices are WAV1>ENV, WAV2>ENV and ENV>WAV2. Copying from the one table to the other involves a slight translation (see Chapter 3).

Next we have AUGM BAR and DIMN BAR standing for augment bar and diminish bar. If one of these is chosen, the duration of every note in the bar where the cursor was left is multiplied by two (for augment bar) or divided by two (for diminish bar).

The last two choices on the COPY list are branches to two other overlay programs that are loaded with COPY. Choosing KEY SIG branches to the program KEY SIG, and SAVE IT branches to the program DUMPO.

KEY SIG

The purpose of the KEY SIG program is to scan through the melody, and add accidentals to all the notes that normally have accidentals in a specified key signature. For example, if the key of B is chosen, all the E's in the melody will be changed to D sharps (as there is no facility for flats) and all the B's will be changed to A sharps.

The KEY SIG program has two tables that are used. The KEY table contains a word for each of the 12 possible key signatures. The first bit of the word states whether the key has sharps or flats. Each of the next seven bits represents one of the 7 notes from C to B. If one of these bits is 1, the note that it represents needs to be changed, and if a bit is 0, the note will not be changed. For example, the first eight bits of the E key signature word are 011 011 00, the first zero indicating that it is a sharp scale, the second bit indicating that C's will be changed to C sharps, the third bit indicating that D's will be changed to D sharps, etc.

When KEY SIG is loaded, it sets up the second table. This table is 32 words long and is initially set to contain the integers (15 bit) in sequence from -11 to 20. These are the numbers that represent the natural pitches. A list of 12 different key signature choices is then presented, and after choosing one of the numbers of the table that represent notes that are not to be changed are set to -1 (24 bits). The table now contains only the numbers representing the pitches that are to have accidentals added. The melody is then scanned through, with the pitch of every note checked against those in the table. If it is there, a '37 is added to the pitch for a flat or a '40 is added for a sharp (see Appendix D). After this is done, a branch is made to DAMMU.

DUMPO

DUMPO is the same as the systems program DUMPER except that different data sets are stored on disc (see Appendix E).

AUTOSAVE and -SAVE IT

These programs enable the dumping of file 13 (AUTO SAVE) or individual melodies (-SAVE IT or ADBOT) on magnetic tape. These are systems programs for the whole computer system and are documented in the DSS Program Library (1147 and 1148).

TUNER(12)

Each of the sound generators contains a table containing a member for each pitch. In the monophonic sound generators a number from the table is stored in an index register which is then counted down to 0 to time each step of the waveform. The number in this table is X where

$$X = \left(\frac{C/1.75}{B} - A \right) - D$$

C is the length of the period in microseconds, 1.75 microseconds is the cycle time of the computer, A is the number of computer cycles in the initialization of every period, B is the number of steps in the waveform, and D is the time of the output.

In the polyphonic sound generator, a number from the table is added to a memory location every time through the play loop causing this memory location to alternate between positive and negative. Therefore this number X, depends on F, the number of cycles in the play loop, C, the number of microseconds in the period and '80,000,000, the number that must be added to the memory location during one period. $X = ('80{,}000{,}000) \frac{F}{C}$.

The TUNER program was written so that a new table could be placed in the sound generator that would produce a different set of pitches. The program consists of an assembly language main program calling a FORTRAN subroutine. When TUNER is loaded, the assembler part of the program presents the user with a list of sound generators. After one of these is chosen, a choice has to be made between tempered and equitempered scales. If a tempered scale is requested, a base frequency is typed in by the user. The FORTRAN subroutine then takes over, and calculates the pitches based on the base frequency and the tempered scale for C.

If an equal tempered scale is chosen, the typewriter prompts for a base frequency, and the number of intervals per octave. The computer then calculates the true pitch table using the formula C x 2A/TONE where C is the base frequency, TONE is the number of intervals per octave, and A is the number of intervals from the base frequency.

After the true pitch table is calculated in both the cases of tempered or equitempered scale, the table for the sound generators is calculated using one of the formulae given previously. This table is overlaid on the old one inside the sound generator.

NEW ENVL

NEW ENVL is a program that writes a Gaussian distribution curve repeatedly in the envelope. The horizontal shaft encoder wheel controls the width of the curve and the vertical wheel controls the amplitude. For example, if the horizontal wheel was set so the curve was '50 words long, the curve would be repeated '100 times in the '5000 word long envelope.

When the program is loaded, the present envelope is stored in the external memory. A list of four choices, ENVEL 1, ENVEL 2, etc. , is then presented on the screen.

ENVEL 1 creates a one tailed Gaussian curve, ENVEL 3 makes a two-tailed curve. ENVEL 2 and ENVEL 4 are versions of ENVEL 1 and ENVEL 3, respectively, but they start the curve one half cycle later.

After a choice is made, the routine which displays the envelope is enacted. Pressing the foot pedal allows changes in the horizontal and vertical parameters as mentioned before. Pressing one of the black buttons activates a subroutine which multiplies the present envelope by the old envelope that was stored in the external memory. Every point of the envelope is loaded, multiplied by the corresponding point of the old envelope (a number between 0 and 1) and then restored in the envelope.

PAR>SLID

As the pitch representation of the notes in SLIDER is slightly different than in DOLY, PAR>SLID takes every word of the melody, checks to see that it is not a command, bar line, or a rest, and using the last 15 bits of the word as a reference, replaces these 15 bits with a value from a table. This new value represents the same pitch in the SLIDER version of WRITER. After the translation has been completed, the SLIDER synthesizer sound generator and the SLIDER version of WRITER are loaded.

SLID>PAR

SLID>PAR is the reverse of PAR>SLID. The last 15 bits of each note are checked against a table of increasing numbers.

When an element of the table is found that is greater than this last 15 bits, we have found the pitch of the note. For example, if the first element of the table that was greater than the pitch was the 17th element, the pitch of that note is 17 semitones above low C. The number representing this pitch for the normal WRITER program, DOLY, replaces the old 15 bits in the note. The synthesizer sound generator PARAPLAY and DOLY are loaded after a call to SLID>PAR.

PITCHGEN

PITCHGEN, stored as a sound generator, allows the use of random numbers, and envelope and waveform tables for melody input. A list of the possible melody inputs is given in Appendix B, Table 9. PITCHGEN also contains a simple monophonic synthesizer sound generator.

The portion of PITCHGEN which generates the new melody is based on the sound generator. When the choice GENERATE is chosen from the PITCHGEN list of choices, this sound generator is modified so that pitches and durations are loaded from the waveform, envelope, melody, or random number generator, instead of solely from the melody. Then, instead of playing these notes, the computer stores them in the melody. The command GEN PART starts PITCHGEN at the present position of the cursor.

Going through this in more detail, when GENERATE is chosen, a -1 is stored in T2 (a 0 is stored if PITCHPART was chosen, then a minus one is stored in T3 (a 0 would have been stored had PLAYER or PLAYPART been chosen). Next the beginning addresses of the 3 waveforms, the envelope, and the melody are stored in pointers. If GEN PART had been chosen (T2 = 0) the melody pointer would be loaded with the address of the note under the cursor.

The buttons are then checked for a command. If the buttons that have been pushed represent a command in Table 9 of Appendix B, a change is then made in the program. For example, if a "Melody from Wave 1" command had been chosen, the instruction that loads the pitches is changed so that it loads from WAVE 1. In this way either the command that loads the pitch or the command that loads the duration is changed. In the case of a duration command that is a combination duration and random, i. e. , "duration wholes and random" or "duration as written and random", two memory instructions have to be changed, one is changed from a "load the duration command" to a "branch to the random generator" command, and the second change is the command which adds the required duration to the random duration, i. e. , for "duration wholes and random", this command will add a whole note to the random duration.

Now that the pitch and duration of the note have been calculated they are stored in the melody. All the pointers are incremented by one, and the next word in the melody is checked for bar lines or commands. If it is a command, it is treated as a button control command. If it is a bar line, the next word is checked for a bar line and if it is one as well, the computer branches to the WRITE program. If this next word in the melody is neither a bar line nor a command, the pitch and duration of the next note are calculated and stored in the next word of the melody.

IX. LIMITATIONS OF THE MUSIC SYSTEM

The NRC computer music system requires that a minimum of 13 interconnected programs and 6 large data tables be in core at any one time. A system of this size is cumbersome and very difficult to modify. A program cannot be lengthened without changing addresses in all the programs after it to make room. The address of an externally addressed memory location may not be changed without modifying the programs that reference it. In brief there is much inertia in a system of this sort. Furthermore, a new program cannot now be added to the group of systems programs as there is no unused space in memory. New programs have to be overlaid on a waveform or envelope table. The latter limitation could be removed by increasing core memory. However, such extension is difficult and expensive because the SEL 840 computer is no longer manufactured.

This chapter, then, is to give suggestions on things that could be done differently in a new system. These changes should be incorporated while the new system is in its early stages, and modifications are more easily implemented.

The timing in our system is based on 32nd notes. It might be desirable to change this basis to 96th notes which would allow triplets to be written into the music. For example, a quarter note would have a duration of twenty-four (96th notes) but a quarter note in a triplet would have a duration of 18 (96th notes). An alternate method of allowing for triplets is a command that would cause the present tempo to be multiplied by 3/2 for triplets, or 5/4 for five notes in the time of four. The initial tempo could be stored while the triplet is played and reloaded when the triplet is over. An example this idea is in the listing for PLAIN 2 (18). A clock interrupt would be necessary for each voice being played, so a change in tempo for one voice would not affect the others.

One of the main limitations in the present hardware is the small amount of core memory (16K). In a computer where 1/2 words are addressable, it could be possible to store two envelope or two waveform tables in the same core area, one table using the most significant 8 bits of each word, and the record table using the least significant 8 bits.

If there is space available in the data words for the notes, it might be possible to add a flat sign capability. In tempered scales, a flattened note is not necessarily the same as a sharp of a note one tone down, i. e. , F sharp is not the same as G flat. This feature may be worthwhile only if it is simple to add, as it is not likely to be used often, though it will make the music more readable.

In any new system using software sound generators, the inclusion of both a square wave polyphonic sound generator and a multi-timbre monophonic sound generator in the basic system would be of great use. It would relieve the musician from the task of constantly changing sound generators.

An attempt could also be made to improve TONE 2 (18) so that it could play four lines of music at once. The trouble with TONE 2 at the moment is the length of time spent in one pass through its play loop. With a modern computer with a shorter cycle time and more hardware registers, all the data for the play loop could be stored in registers, and the cycle time through the play loop could be shortened considerably. If this were done, a four voice TONE 2 and an elaborate 2ND PLAY could be the standard polyphonic sound generator and monophonic sound generator respectively and both could be stored in core at the same time.

In a similar vein, an extended version of 4 WRITE could easily contain all the capabilities of DOLY, making DOLY obsolete. Therefore 4 WRITE could become the main core resident WRITER program, and DOLY could be discarded.

These suggestions are intended to indicate some improvements that could easily be made and to illustrate some advantages of more modern computers. They show that further development of the system is possible and desirable. Since changes become progressively more difficult to implement as the system is developed we feel that a great deal of thought should go into the early stages of a new system and that the advantages and shortcomings of this one should be carefully evaluated before proceeding with an improved version.

X. CONCLUSIONS

When this system was being created, the objective of the project was to open new avenues of man-computer communications. This means that the computer had to communicate with the musician in a manner that was acceptable to the musician. It would have been easy to overlook this aim and to program the computer so that it was easy for the programmer or engineer to use. To avoid this fundamental error a wide variety of musicians were invited to work with the music system with an invitation to make suggestions for improving or modifying it. These modifications were either to simplify the usage of the computer or to extend its facilities. It was considered very important to have a variety of musicians. Quite often one musician would brand as useless a facility that is used extensively and praised by another.

In general, there have been two categories of musicians, commercial and experimental. The commercial musicians have composed successfully for films, radio and TV. Many experimental pieces have also been produced. The computer music system seems to have been accepted as a valuable musical tool by many musicians who have seen and used it. Whilst we feel that the potential of an interactive system has been convincingly demonstrated, there is scope for a great deal more development and refinement. We believe that this kind of computer-aid-to-composers system has a place in the future development of music and that it will be accepted more and more readily by musicians as it is improved and tailored to their needs.

ACKNOWLEDGEMENT

I would like to acknowledge the help and direction received from Mr. J. K. Pulfer, the inventive genius behind the music system. I would also like to thank Mr. Frank Cairns and the Data Systems Section whose constructive criticism made this report possible.

REFERENCES

DATA SYSTEMS PROGRAM LIBRARY, REED, M-50, NRC

1. No. 1030C, LISTER, M. WEIN, 1970
2. No. 1085B, D$STOR, J.K. PULFER AND M. JAVOR, 1970
3. No. 1087, CALCAR, M.D. DUGGAN, 1968
4. No. 1099, MELODY SUBROUTINE, J.K. PULFER, 1969
5. No. 1100, D AMMU, J.K. PULFER, 1969
6. No. 1101, EXPERIMENTAL MUSIC PROGRAMS, J.K. PULFER, 1969
7. No. 1107A, I$DECR, J.L. WOLFE, 1969
8. No. 1108, MUSIC PROGRAM PACKAGE, J.K. PULFER, 1969
9. No. 1118, DELEZE, J.L. WOLFE, 1969
10. No. 1138, DRAMU, K.M. WILSON, 1970
11. No. 1139, POLYFONY, P.P. TANNER, 1970
12. No. 1146, TUNER, T. TRICKER, 1971
13. No. 1147, AUTOSAVE, J.K. PULFER, 1971
14. No. 1148, ADBOT, J.K. PULFER, 1971

PUBLICATIONS

1. Pulfer, J.K. Man-Machine Interaction in Creative Applications, International Journal Man-Machine Studies, 3/1/1971
2. Pulfer, J.K. Programmers Reference Manual for a Digital CRT Display, ERB-788, REED, NRC., Ottawa, 1968.
3. Pulfer, J.K. and Tanner, P.P. Marvelous Music Machine Manual, unpublished user's manual, REED, NRC, 1970, 1971 and 1972.
4. Tanner, P.P. Some Programs for the Computer Generation of Polyphonic Music, ERB-862, REED, NRC, 1971.
5. Reference Manual for the SEL 840A General Purpose Digital Computer. System Engineering Laboratories, Incorporated, Fort Lauderdale, Florida, 1966.

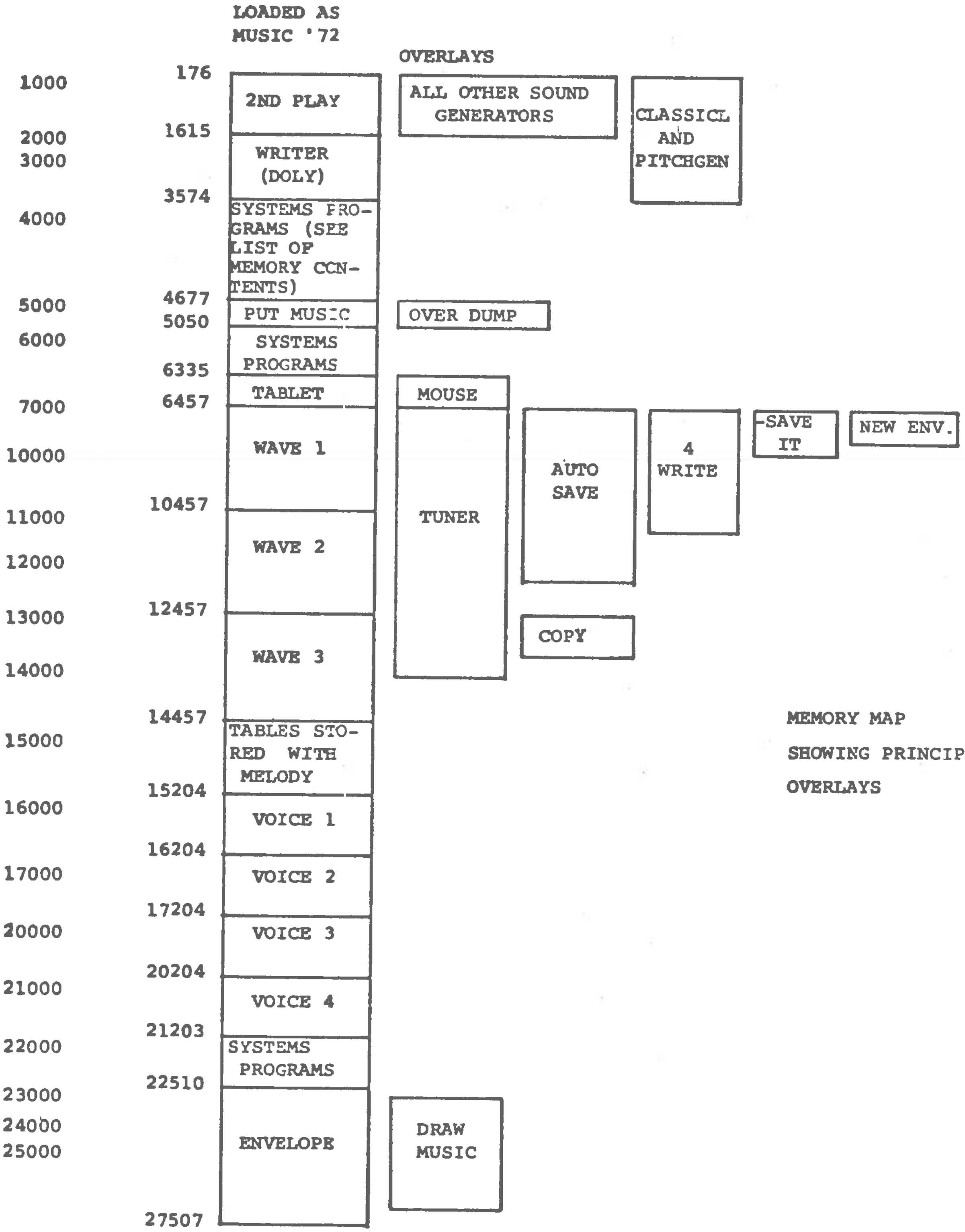
APPENDIX A - LIST OF MEMORY CONTENTS

	ADDRESS	NAME	CODE	MEANING
	(176		E	BEGINNING OF SOUND GENERATORS
SOUND	(200		E	PLAY IT
GENERATOR	(1516	NTB	D	LIST OF CHOICES
	(1556	MEST	D	MESSAGES TABLE
	1616	KLOK	E	CLOCK INTERRUPT
	1623	IFLG	P	LENGTH OF NOTE LEFT
	(1624 (	MARMU	E	BEGINNING OF MUSIC WRITING PROGRAM, "MARKERS"
	(1632	DITMU	E	"TEMPO"
	(1640	DAIMU	E	"MODIFY"
	(1650	DADMU	E	"DELETE"
	(1660	DIMMU	E	"INSERT"
	(1666	DAMMU	E	"WRITE" (BYPASSES LIST OF VOICES)
WRITER	(2444	MPTR	P	CURRENT CURSOR ADDRESS
	(2471	TEMPO	E	'SET TEMP'
	(2574	MWRD	P	PITCH INDEX VALUE
	(3220	DUMMU	E	"WRITE" (POLYPHONIC)
	(3316	FG	P	CHECK FLAG
	(3362	COUNT	P	NO OF 32NDS TO CURSOR (COUN)
	(3442 (	FIRST	P	CURRENT ARRANGED BAR LINE POINTER (FIRS)
	(3451	HERE	P	COUNT UP TO HERE
	(3453	KFOUR	P	FLAG FOR COUNT (K4)
	(3511	PATCH	E	"WRITE" (MONOPHONIC)
	(3515	PATCH2	E	SET DAMEL/TRAN COUNTERS
	3574	LOOK	E	"NEXTPAGE"
	4461	SPDTA	E	Change Library to decimal
	4553	DELEZE	E	Delete reading zeros
	4635	GETTER	E	"GET"
	4677	DUMPER	E	"DUMP"
	5051	RHYTHM	E	PERCUSSION SUBROUTINE
	5077	RYTAB	D	TABLE OF PERCUSSION CHOICES
	5124	ARRAN	E	"ARRANGE"
	5137	THUD	P	ADDRESS OF END OF ARRANGEMENT TABLE
	5142	I$DECR	E	INPUT NUMBERS FROM TYPEWRITER
	5426	DAMEL	E	STORES ADDRESSES OF BAR LINES
	5473	TUTAB	P	ADDRESS OF END OF VOICE
	5474	TUMEL	P	ADDRESS OF BEGINNING OF VOICE
	5615	TRAN	E	REARRANGE TABLE OF BAR LINE ADDRESSES

	ADDRESS	NAME	CODE	MEANING
	5621	TULUP2	P	ADDRESS OF END OF ARRANGEMENT TABLE
	5624	TULUP4	P	ADDRESS OF END OF ARRANGED TABLE OF BAR LINE ADDRESS
	5634	ULTA	D	MELTAB LESS 1
	5635	MELTAB	D	VOICE ONE ARRANGED TABLE OF BAR LINE ADDRESSES
	5744	ULTA1	D	END OF VOICE ONE TABLE
	6074	ULTA2	D	END OF VOICE TWO TABLE
	6214	ULTA3	D	END OF VOICE THREE TABLE
	6334	ULTAB	D	END OF VOICE FOUR TABLE
	6335	FORM	E	"SEE WAV1"
	6343	FORM1	E	"SEE WAV2"
	6351	FORM2	E	"SEE WAV3"
WAVE 1	6457	FORMTB	D	FIRST WAVEFORM TABLE
WAVE 2	10457	FORMT1	D	SECOND WAVEFORM TABLE
WAVE 3	12457	FORMT2	D	3RD WAVEFORM TABLE
	(14457	AMPT		TABLE OF AMPLITUDE VALUES (NOT USED)
	(14470	DMPT		TABLE OF DECAY VALUES
	(14501	SPED	P	INITIAL TEMPO
	(14622	ARA1	D	END OF ARRANGEMENT TABLE FOR VOICE FOUR
	(14742	ARA2	D	END OF ARRANGEMENT TABLE FOR VOICE TWO
	(15062	ARA3	D	END OF ARRANGEMENT TABLE FOR VOICE THREE
MELODY	(15202	TREAD	D	END OF ARRANGEMENT TABLE FOR VOICE ONE
	(15204	MUTAB	D	BEGINNING OF VOICE ONE
	(16203	MUT1	D	END OF VOICE ONE
	(16204	MUT12	D	BEGINNING OF VOICE TWO
	(17203	MUT2	D	END OF VOICE TWO
	(17204	MUT13	D	BEGINNING OF VOICE THREE
	(20203	MUT3	D	END OF VOICE THREE
	(20204	MUT14	D	BEGINNING OF VOICE FOUR
	(21202	AUTAB	D	2ND FROM END OF VOICE FOUR
	(21203	UTAB	D	END OF VOICE FOUR
	21204	PLAPIC	E	ENTRY FOR PLAY PICTURE SUBROUTINE PICTURE
	22054	PLBPIC	P	DISTANCE FROM START OF PLAY PICTURE
	22055	PLCPIC	P	ADDRESS OF START OF PLAY
	22164	DYNA	E	"SEE ENV"
	22345	GENTAB	E	"GENERATE"
	22451	SHIFT	E	"TRNSFORM"

	ADDRESS	NAME	CODE	MEANING
PARAMETERS	22477	NENVT	D	ENVELOPE PARAMETERS TABLE
ENVELOPE	22510	ENVT	D	ENVELOPE
	27553	ENVP2	D	ADDRESSES OF ENVELOPE PARAMETER TABLES-15
	27654	NTB2	D	LIST OF CHOICES (SYNTHESIZER PROGRAM)
SYSTEMS	30400	D$STOR	E	DISC HANDLER SUBROUTINE
PROGRAMS	31600	LISTER	E	SUBROUTINE TO CREATE LIST OF CHOICES

D - Beginning or end of a data table
P - Externally accessed pointer or flag
E - Entry points



MEMORY MAP

SHOWING PRINCIPAL

OVERLAYS

APPENDIX B - LIST OF BUTTON COMMANDS

BUTTONS	PARAMUS	1ST PLAY	2ND-5TH PLAY	PLAIN4+NORM2	DKAY4	TONE2
- - - X -	DK1	DK1	DK1	*1	DK1	DK1
- - X - -	DK2	DK2	DK2	*2	DK2	DK2
- - X X -	DK3	DK3	DK3	*3	DK3	DK3
- X - - -	DK4	DK4	DK4	*4	DK4	DK4
- X - X -	DK5	DK5	DK5	*5	DK5	DK5
- X X - -	DK6	DK6	DK6	*6	DK6	DK6
- X X X -	DK7	DK7	DK7	*7	DK7	DK7
X - - - -	DK8	DK8	DK8	*10	DK8	DK8
X - - X -	EN>2	TIM1	TIM1	*11	*11	TIM1
X - X - -	WA>2	TIM2	TIM2	*12	*12	TIM2
X - X X -	EN>3	TIM3	TIM3	*13	*13	TIM3
X X - - -	WA>3	SLON	TIM4	*14	*14	TIM4
X X - X -	SKIP	SLOF	TIM5	TCK1	*15	TIM5
X X X - -	O'S	*16	TIM6	TCK2	*16	TIM6
X X X X -	T1	*17	TIM7	TK12	*17	TIM7
- - - - X	T2	STAN	TIM8	UOCT	UOCT	TIM8
- - - X X	T3	DOCT	TIM9	DOCT	DOCT	TIM9
- - X - X	T4	*22	TM10	*22	*22	TM10
- - X X X	T5	*23	TM11	*23	*23	TM11
- X - - X	T6	PERC	PERC	PERC	PERC	TM12
- X - X X	T7	NOPC	NOPC	NOPC	NOPC	TM13
- X X - X	PERC	ENV	ENV	*26	*26	*26
- X X X X	NOPC	NENV	NENV	*27	*27	*27
X - - - X	GLIS	GLIS	GLIS	*30	*30	*30
X - - X X	PP	PP	PP	PP	PP	PP
X * X - X	P	P	P	P	P	P
X - X X X	MP	MP	MP	MP	MP	MP
X X - - X	MF	MF	MF	MF	MF	MF
X X - X X	F	F	F	F	F	F
X X X - X	FF	FF	FF	FF	FF	FF
X X X X X	NORM	NORM	NORM	STAN	STAN	*37
PALM BUTTON	SLUR	SLUR	SLUR	*40	SLUR	SLUR

TABLE 2

EXPLANATION OF BUTTON CONTROL COMMAND CODES

DK1 - DK8	8 PRESET DECAY LEVELS (LEGATO - STACCATO)
TIM1 - TM11	DIFFERENT TIMBRES
SCON	PHASE SLIDE ON
SLOF	PHASE SLIDE OFF
TCK1	TRACK ONE
TCK2	TRACK TWO
TK12	BOTH TRACKS
DOCT	PITCH DOWN ONE OCTAVE
STAN	PITCH AS WRITTEN
UOCT	PITCH UP ONE OCTAVE
PERC	PERCUSSION ON
NOPC	PERCUSSION OFF
ENV	ENVELOPE ON
NENV	ENVELOPE OFF
GLIS	GLISSANDO ON
NORM	GLISSANDO OFF
PP,P,MP,MF,F,FF	6 PRESET AMPLITUDE LEVELS
SLUR	MAKE DECAY APPLY TO FOLLOWING GROUP OF NOTES
EN>2	ENVELOPE OUTPUT TO PARAMETER 2
WA>2	WAVEFORM OUTPUT TO PARAMETER 2
EN>3	ENVELOPE OUTPUT TO PARAMETER 3
WA>3	WAVEFORM OUTPUT TO PARAMETER 3
SKIP	NO OUTPUT
O's	OUTPUT OF O
T1	ENVELOPE 1 OR WAVE 1
T2	ENVELOPE 2 OR WAVE 2
T3	ENVELOPE 3 OR WAVE 3
T4	ENVELOPE 4
T5	ENVELOPE 5
T6	ENVELOPE 6
T7	FULL ENVELOPE TABLE

TABLE 3

SPECIAL COMMANDS FOR "6TH PLAY", "8TH PLAY", AND "10THPLAY"

X X - - -	MOD	Use modified version of waveform
X X - X -	UNMD	Use waveform as drawn

TABLE 4

SPECIAL COMMANDS FOR "7TH PLAY"

- - - - X	NPRT	PORTAMENTO OFF
- - - X X	PORT	PORTAMENTO ON

TABLE 5

SPECIAL COMMANDS FOR "22NDPLAY"

X - - - X	ARON	ARTICULATION ON
X X X X X	AROF	ARTICULATION OFF

TABLE 6

"PLAY+SEE" COMMANDS FOR POLYPHONIC SOUND GENERATORS AND 4 WRITE

- - - X 0	DISPLAY	VOICE	1
- - X - 0		VOICE	2
- X - - 0		VOICE	3
X - - - 0		VOICE	4

TABLE 7

COMMANDS FOR "KEYBOARD"

PALM BUTTON	THUMB BUTTON	Lines of Music Played by Computer	Lines of Music Played by Keyboard
-	-	0	4
-	X	3	1
X	-	2	2
X	X	1	3

TABLE 8

ENVELOPE GENERATOR COMMANDS

-	-	-	X	-	SET ATTACK RATE
-	-	X	-	-	SET FIRST DECAY RATE
-	-	X	X	-	SET SECOND DECAY RATE
-	X	-	-	-	SET STARTING LEVEL
-	X	-	X	-	SET BEGINNING ATTACK TIME
-	X	X	-	-	SET BEGINNING OF FIRST DECAY
-	X	X	X	-	SET BEGINNING OF SECOND DECAY

TABLE 9

SPECIAL COMMANDS FOR PITCHGEN

X	-	-	X	-	MAW	MELODY AS WRITTEN
X	-	X	-	-	MW1	MELODY FROM WAVE 1
X	-	X	X	-	MW2	MELODY FROM WAVE 2
X	X	-	-	-	MW3	MELODY FROM WAVE 3
X	X	-	X	X	MENV	MELODY FROM ENVELOPE
X	X	X	-	-	MAWP	MELODY AS WRITTEN + RANDOM
X	X	X	X	-	MR	MELODY RANDOM
-	-	-	-	X	DAW	DURATIONS AS WRITTEN
-	-	-	X	X	D32	DURATIONS 32ND NOTES
-	-	X	-	X	D4	DURATIONS QUARTER NOTES
-	-	X	X	X	DWH	DURATIONS WHOLE NOTES
-	X	-	-	X	D4R	DURATIONS QUARTERS + RANDOM
-	X	-	X	X	DWHR	DURATIONS WHOLES + RANDOM
-	X	X	-	X	DAWR	DURATIONS AS WRITTEN + RANDOM
-	X	X	X	X	DR	DURATIONS RANDOM
X	-	-	-	X	DW1	DURATIONS FROM WAVE 1

APPENDIX C - PERCUSSION CONTROL

	F	BRUSH	CYMBAL	RIM
	E	BRUSH	CYMBAL	BASS
	D	BRUSH	CYMBAL	SNARE
	C	CYMBAL	TAMBOURINE	
	B	CYMBAL	BRUSH	
	A	SILENCE		
	G	CYMBAL	CLAVE	
	F	CYMBAL	RIM	
	E	CYMBAL	SNARE	
	D	CYMBAL	CONGA	BASS
	C	CONGA	TAMBOURINE	
	B	CONGA	BRUSH	
	A	CONGA	CYMBAL	
	G	CONGA	CLAVE	
	F	CONGA	RIM	
	E	CONGA	SNARE	
	D	RIM	BASS	CYMBAL
MIDDLE	C	BASS	TAMBOURINE	
	B	BASS	BRUSH	
	A	BASS	CYMBAL	
	G	BASS	CLAVE	
	F	BASS	RIM	
	E	BASS	SNARE	
	D	BASS	CONGA	
	C	TAMBOURINE		
	B	BRUSH		
	A	CYMBAL		
	G	CLAVE		
	F	RIM		
	E	SNARE		
	D	CONGA		
	C	BASS		

APPENDIX D - CONTENTS OF BITS 9 - 23 OF THE WORDS REPRESENTING NOTES OR RESTS

NOTES	BITS 9-23	NOTES	BITS 9-23
F	'24	F#	'64
E	'23	E#(F)	'63
D	'22	D#	'62
2 Octaves Above Middle C	'21	C#	'61
B	'20	B#(C)	'60
A	'17	A#	'57
G	'16	G#	'56
F	'15	F#	'55
E	'14	E#(F)	'54
D	'13	D#	'53
1 Octave Above Middle C	'12	C#	'52
B	'11	B#(C)	'51
A	'10	A#	'50
C	'7	C#	'47
F	'6	F#	'46
E	'5	E#(F)	'45
D	'4	D#	'44
Middle C	'3	C#	'43
B	'2	B#(C)	'42
A	'1	A#	'41
G	'0	G#	'40
F	'-1	F#	'37
E	'-2	E#(F)	'36
D	'-3	D#	'35
1 Octave Below Middle C	'-4	C#	'34
B	'-5	B#(C)	'33
A	'-6	A#	'32
G	'-7	G#	'31
F	'-10	F#	'30
E	'-11	E#(F)	'27
D	'-12	D#	'26
2 Octaves Below Middle C	'-13	C#	'25

APPENDIX E - MAXIMUM LIMITS FOR STORING

PUT AWAY COMMAND	CODE	BEGINNING ADDRESS	MINIMUM LAST ADDRESS		MAXIMUM LAST ADDRESS
1 VOICE	1	14457	15206		16204
2 VOICES	1	14457	16206		17204
3 VOICES	1	14457	17206		20204
4 VOICES	1	14457	20206		21204
PUT WAV1	2	6457		10456	
PUT WAV2	3	10457		12456	
PUT WAV3	4	12457		14456	
PUT ENV	5	22477		22507	
-	-			-	
1 ARGMT	1	14457		14622	
2 ARGMTS	1	14457		14742	
3 ARGMTS	1	14457		15062	
4 ARGMTS	1	14457		15202	
MONO MEL	1	14457	15206		21204
ALL WAVS	2	6457		14456	
WHOL ENV	7	22510		27507	
PLAYER	0	200		1615	
SCALE	6	574		673	

APPENDIX F - LIST OF SEL 840A INSTRUCTIONS

MNE-MONIC	OP CODE	FUNCTION	PAGE	MNE-MONIC	OP CODE	FUNCTION	PAGE
AAM	31	Add (A) to (M)	2-14	HLT	00-00	Halt	2-45
AIP	172	(Unit) to A	2-55				
AMA	05	Add (M) to (A)	2-13	IAB	00-06	(A) to B, (B) to A	2-32
AMX	61	Add (M) to (X)	2-14	IAM	44	(A) to M; (M) to A	2-12
AOP	170	(A) to Unit	2-54	IIB	34	(X)+1, Branch if not 0	2-23
ASC	00-20	Comp. A sign	2-26	IMS	14	(M)+1;Skip if 0	2-20
BAN	23	Branch if (A) neg.	2-22	LAA	01	(M) to A	2-8
BAP	24	Branch if (A) pos.	2-22	LBA	02	(M) to B	2-9
BAZ	22	Branch if (A)=0	2-21	LCS	57	Switches to A	2-10
BOF	25	Branch on O'FLOW	2-23	LIX	32	(M) to X	2-9
BRU	11	Branch to M	2-18	LSA	00-11	Left Shift A, Arith.	2-36
				LSL	00-16	Left Shift A, Log.	2-41
CEU	130	Command Ext. Unit	2-52				
CLA	00-03	Clear (A) to 0	2-31	MAA	27	(M) AND (A)	2-28
CMA	15	Compare (A) and (M)	2-21	MEA	26	(M) Exclusive OR (A)	2-27
CNS	20	Convert No. System	2-26	MIP	176	(Unit) to M	2-57
CSB	00-07	Copy B sign	2-33	MOA	30	(M) OR (A)	2-29
				MOP	174	(M) to Unit	2-56
DIV	10	Divide	2-16	MPY	07	Multiply	2-15
EAB	21-03	(EA) to EB	2-62	NEG	56	2's comp. (A)	2-25
EAD	45	EAU add	2-65	NOP	00-22	No operation	2-45
EBA	21-02	(EB) to EA	2-61	NOR	00-32	Normalize (A) and (B)	2-43
EDP	21-12	EAU D. P. mode	2-59				
EDV	50	EAU Divide	2-68	PID	0.43	P.I. Disable	2-46
EFP	21-14	EAU F. P. mode	2-60	PIE	1.43	P.I. Disable	2-46
EFU	21-11	Un-normalize F. P.	2-58	PIR	36	P.I. Return	2-19
EIA	21-01	(EA) to EB, (EB) to EA	2-61	POF	63	Protect Bit Off	2-74
ELL	54	Load LSH of (EA)	2-72	PON	62	Protect Bit On	2-73
ELN	51	Load (M) in EA	2-69				
ELO	52	Load (M) in EA	2-70	RNA	60	Round (A) by (B)	2-17
EMU	47	EAU Multiply	2-67	RSA	00-10	Right shift A, Arith.	2-35
ENO	21-00	EAU Normalize	2-60	RSL	00-15	Right shift A, Log.	2-40
EPS	21-06	Skip if (EA) pos.	2-63				
ESL	55	Store LSH of (EA)	2-72	SAS	00-21	Skip on A sign	2-24
ESN	21-07	Skip if (EA) neg.	2-64	SMA	06	Subtract (M) from (A)	2-15
ESO	21-10	Skip on EAU O'FLOW	2-64	SMP	35	Skip if (M) pos.	2-24
ESP	21-13	EAU S. P. mode	2-59	SNS	134	Skip No Switch	2-20
ESR	00-23	Skip if EAU not ready	2-62	SPB	12	Store Place & Branch	2-19
EST	53	Store (EA)	2-71	STA	03	(A) to M	2-10
ESU	46	EAU Subtract	2-66	STB	04	(B) to M	2-11
ESZ	21-05	Skip if (EA)=0	2-63	STI	33	(X) to M	2-11
EXU	16	Execute (M)	2-44				
				TAB	00-05	(A) to B	2-32
FLA	00-13	Full Left Shift, Arith.	2-38	TAI	00-01	(A) to X	2-30
FLL	00-17	Full Left Shift, Log.	2-42	TBA	00-04	(B) to A	2-32
FRA	00-12	Full Right Shift, Arith.	2-37	TBI	00-02	(B) to X	2-31
FRL	00-14	Full Left Rotate, Log.	2-39	TEU	132	Test Ext. Unit	2-53

SELECTED CRT DISPLAY COMMANDS

USAGE	COMMAND TO CRT	COMMAND TO EXTERNAL MEMORY
Load scale factor register	'11	'12
Plot a point	'251	'252
Plot high speed by transferring scale factor	'71061	'71062
Blank point	'4251	'4252
Plot four characters	'701	'702
Load colour register	'501	'502
Plot adjacent blank points	'4351	'4352
Line by transferring scale factor	'1061	'1062
Input shaft encoder		'6
Plot single character	'401	'402
Plot character + Y	'441	
Start buffer to display		'100000
Set memory address		'30000000
Branch to 0		'20000000

N.B. The above numbers are stored in the data word following a C.E.U. command. For complete information see ERB-788. (16)

APPENDIX G - VARIABLES IN 1ST PLAY AND 2ND PLAY

1ST PLAY	2ND PLAY	CODE	MEANING
X	X	ENVF	= 0 if envelope is off, -1 if envelope on
X	X	T3	= 0 if percussion off -1 if percussion on
X	X	NFLG	= 0 if glissando off -1 if glissando on
X	X	LWDF	= 0 if slur on -1 if slur off
X	X	WFLG	positive if note, negative if rest
X	X	T2	= 0 if PLAYPART chosen, -1 otherwise
X	X	$FG	= 1 if PLAY+SEE chosen -1 if PLAYER chosen
X		FFLG	used for lowering pitch one octave
X		ZFLG	= 0 if phase shift off = -1 if phase shift on
X		PFLG	= 0 if pitch down one octave -1 if standard pitch
	X	KM12	No. of steps in waveform
X	X	$IFLG	No. of 32nd Notes left to play (negated)
X	X	AAA	No. of notes until PLAPIC must be recalled (negated)
X	X	CAMP	current amplitude
X	X	DAMP	decay amplitude
X	X	HERE	address of note under cursor (for PLAYPART)
X	X	POIN	address of present position in bar line table
X	X	PEEC	address of present position in melody
X	X	NPCH	present pitch
X	X	AMP	initial amplitude (before decay)
	X	SAV	output pitch
X	X	SKIL	output amplitude
X	X	NOTE	Contents of word currently being played

APPENDIX H - KEYBOARD VARIABLES

CODE	MEANING
AAC 3	no. of 32nd time intervals left in note
COMQ	pitch for VOICE 4 (=CONT-1)
COMR	pitch for VOICE 3 (=CONT-2)
COMS	pitch for VOICE 2 (=CONT-3)
COMT	pitch for VOICE 1 (=CONT-4)
CONQ	sign of amplitude control for VOICE 4
CONR	sign of amplitude control for VOICE 3
CONS	sign of amplitude control for VOICE 2
CONT	sign of amplitude control for VOICE 1
CONT,3	pitches
CRUB,1	amplitude table
CRUB,3	present position is arranged bar line table
GOOO,3	present amplitudes
K1	present amplitude for VOICE 1 (=GOOO-4)
K1,4	initial amplitudes
K2	present amplitude for VOICE 2 (=GOOO-3)
K3	present amplitude for VOICE 3 (=GOOO-2)
K4	present amplitude for VOICE 4 (=GOOO-1)
NOT1	no. of 32nd time intervals left in note for VOICE 1 (=AAC-4)
NOT2	no. of 32nd time intervals left in note for VOICE 2 (=AAC-3)
NOT3	no. of 32nd time intervals left in note for VOICE 3 (=AAC-2)
NOT4	no. of 32nd time intervals left in note for VOICE 4 (=AAC-1)
OLDN	last input word from keyboard
ONT1,3	present note playing
P ,3	present position in melody
T1	choice in main list
T1,3	note playing during the last 32nd time interval
T2	choice in secondary list

APPENDIX I - DOLY VARIABLES

CODE	TYPE	INDEX INSTRUCTION	MEANING
ADDX	C	-	no. of notes to be plotted before cursor note
BEEP	P	3	address of the end of chosen voice less one
CNTR	C	-	current duration in COUN subroutine
CPTR	O	-	current contents of index 2
CRFL	F	-	=0 before cursor drawn, =-1 after cursor drawn
DEE	O	-	vertical position of rests
ENCO	O	-	old encoder position
ENC	O	-	new encoder position
FFLG	F	-	-VE if right foot pedal pressed, =0 otherwise
FG	F	-	-1 for WRITE+CH, =0 for WRITER
FIRS	P	-	address of beginning of arranged bar line table for chosen voice
HERE	P	-	address of cursor note for COUN
ADD	O	-	controls ADD X
JPTR	P	-	address of beginning of chosen voice
KEYB	O	-	contains button input data
KFLG	F	-	=0 when buttons pushed, =-1 after input processed
KPTR	P	2	address of beginning of chosen voice +36
K1,1	E	-	addresses of end of melodies
K2,1	E	-	addresses of beginning of melodies
K3,1	E	-	addresses of end of arrangement tables
K4	F	-	+VE if COUN counts to cursor, -VE if COUN counts to value in VAL
K5,1	E	1,STA	addresses of end of arranged bar line table
K6,1	E	1,STA	address of beginning of arranged bar line
K7	P	-	address of end of music
LFFL	F	-	-VE if left foot pedal pushed, =0 otherwise
MEEP	P	3	address of end of chosen voice
MPTR	P	-	current address of cursor note

CODE	TYPE	INDEX, INSTRUCTION	MEANING
MWRD	O	-	pitch of current dot position
M888	O	-	integer 2
NCR	O	-	X increment for messages
PEEP	O	-	storage for A accumulator
POIN	P	-	pointer to arranged bar line table for COUN
POS	O	-	X position
RPTR	P	2	address of end of chosen voice +31
UFLG	F	-	+VE if note, -VE if rest
UPTR	P	2	address of first displayed note +'40
VAL	O	-	duration where COUN places cursor
XNCR	O	-	X spacing
INCR	O	-	='40
YNC2	O	-	vertical spacing between staff lines
YTAB	O	-	pitch of present position of dot

VARIABLE TYPES

F	Flag
P	External Pointer
E	Fixed external address pointer
C	Counter
O	Other data

MODE FLAGS

	WRITER	INSERT	DELETE	MODIFY	MARKERS	TEMPO	USE
MAFL	-	-	-	-	0	-1	Tempo
TFLG	-1	-1	-1	-1	0	0	Marker or Tempo
MFLG	0	-1	0	-1	-1	-1	Insert
DFLG	0	0	-1	-1	-1	-1	Write or Insert
NFLG	0	0	-1	0	0	0	Delete

APPENDIX J - 4 WRITE VARIABLES

CODE	TYPE	MEANING
B1,3	E	Beginning of arranged bar line tables
B2,3	E	End of arranged bar line tables
BRIT,3	O	Intensity and scale (notes)
BRI,3	O	Intensity and scale (sharps)
BRIT	O	Bright intensity and large scale (notes)
BRI	O	Bright intensity and large scale (sharps)
BUTA	O	Contents of keyboard register
BLUE	O	Blue colour data
C1,3	P*	Address of current note
COLR,3	O	Colour of notes and rests
COLR	O	Red colour data
D1,3	P	Address of first note on display
DBL1,3	F	=0 if double bar line has passed through display, =-1 otherwise
DBL2,3	F	=0 if current note is after double bar line, =-1 otherwise
DARK	O	Dim intensity and small scale (notes)
DAR	O	Dim intensity and small scale (sharps)
E1,3	C*	Number of 32nd time intervals up to end of last plotted note
F1,3	C	Number of 32nd time intervals up to and including first note on display.
G1,3	C	Number of 32nd time intervals up to but not including 1st note on display
H	C	Number of 32nd notes to present position
H,3	F*	Flag for finding F1,3
HILO,3	F	+VE if note stems up, -VE if down
INTZ	O	Storage of original red button interrupt command
INTN	O	Initial X position
J	F	Voice of word currently being written
K	O	Various uses

CODE	TYPE	MEANING
KFLG	F	=0 when buttons are pushed, =-1 after buttons have been dealt with
L	C	Contents of smallest G1,3
N	F	=0 if no chosen voice; otherwise contains voice number of chosen voice
N,3	F	+VE if voice not chosen (LOOK), -VE if voice chosen
NFL	F	=0 if LOOK chosen, -VE otherwise
NCR	O	spacing for messages
P,3	P	address of position in arranged bar line table of first note on display
POS	O	X position
POSA	O	X position for sharps, and dots
POSB	O	Y position of dot
Q,3	P	Address of current position in arranged bar line table
R	C	Plot cursor when 0
S	P	Address of note under cursor
T	F	=0 for write; =1 for INSERT; =2 in DELETE; =3 in MODIFY; =4 for MARKERS; =5 for TEMPO
T,3	E	End of voices
U	C	Number of notes before cursor drawn
V	C	Number of piled up messages (negated)
W,3	E	Beginning of voices
X,3	E	End of arrangement tables
X	F	=-1, move forward; =0 no motion; =+1 move backwards
XFLG	F	=-1 drift forward; =0 no drift; =+1 drift backwards
XNCR,3	O	Spacing after sharps and dotted notes
XNCR	O	Large spacing for sharps and dotted notes
XNCS	O	Small spacing for sharps and dotted notes
Y	C	E1,3 of note after cursor
Z	F	=-1 if cursor has just been plotted, =0 otherwise

VARIABLE TYPES

F	Flag
P	Pointer to data tables
C	Counter
E	Fixed external address pointer
O	Other data
*	Used for a different purpose by the Going Backward routine

INDEX

ARRAN program description 7.2

ARRANGE 4.5

AUTOSAVE program description 8.4

Bar lines format 3.2
- use 4.6

BRANCH 4.7

CHR INV program description 7.1
- use 4.6

CHR TRAN program description 7.1
- use 4.6

COMMANDS format 3.2
- list of B.1
- Meaning B.2
- use 4.4

CONTINUE 4.9

COPY list of choices 4.11
- use 4.7, 4.11
- program description 8.1

DAMEL program description 7.3

DELETE use 4.3

DELEZE program description 7.1

DIA INV program description 7.1
- use 4.6

DIA TRAN use 4.6

DKAY 4 program description ERB862
- use 4.9

DOLY listing I
- program description 6.1

DRAMU program description 6.10
- use 4.11

DUMPER program description 7.2

DUMPO program description 8.4

DYNA program description 7.5

8TH PLAY commands B.3
program description 5.7
use 4.12

Envelope format 3.1
program description (DYNA) 7.6
use 4.9

5TH PLAY use 4.11

1ST PLAY commands B.1
program description 5.6
use 5.6

FORM program description 7.3

4 WRITE list of choices 4.10
listing J
program description 6.11
use 4.10

14THPLAY 6.11

4TH PLAY 4.11

GEN ENV 4.7

GENERATE 4.12

GEN PART 4.12

GET 4.5

GETTER program description 7.2

Hardware 2.1

HI PLAY 4.11

INSERT 4.3

INVERT program description 7.1
use 4.9

I$DECR program description 7.2

KEYBOARD listing H
  program description ERB 862
  use 4.9

KEY SIG program description 8.4

KLOK program description 7.1

LOOK (choice in 4 Write) 4.10

LOOK (NEXTPAGE Program) program description 7.1

Main List of Choices 4.1

MARKERS 4.4

Memory contents A.1
  format 3.1
  hardware 2.1
  map A.4

MODIFY 4.3

Monophonic sounds, generator program description 5.1
  use 4.11

NEW ENVL program descrition 8.5

NEXT PAGE list of choices 4.6
  program description (LOOK) 7.1
  use 4.5

NOTES format 3.2

PARAMUS commands B.1

PAR>SLID program description 8.6
  use 4.17

Percussion program description (RHYTHM) 7.3
  commands C.1

PLAIN 2 program description ERB 862
  use 4.9

PITCHGEN 8.6

PLAIN 4	commands B.1
	program description ERB 862
	use 4.9
PLAPIC	program description 6.10
PLAYER	4.4
PLAY KEY	4.9
PLAYPART	4.4
PLAY+SEE	4.4
POLYFON	4.11
PUT AWAY	program description (DUMPER) 7.3
	use 4.5
QUIT	4.4
RECORD	4.5
RETROGRD	4.7
REV ENV	4.7
RHYTHM	program description 7.2
SAVE IT	program description (DUMPO) 8.4
2ND PLAY	commands B.1
	listing G
	program description 5.1
	use 5.1
SEE ENV	commands B.8
	program description (DYNA) 7.6
	use 4.7
SEE WAVI	program description (FORM) 7.5
	use 4.6
SEE WAV2	4.6
SEE WAV3	4.6
SET TEMP	4.7
7TH PLAY	command B.4
	program description 5.8
	use 4.12

6TH PLAY commands kB.3
program description 5.8
use 4.12

SLIDER (writer program) 6.10

SLID>PAR use 4.17
program description 8.6

Sound Generators, monophonic, program description 5.1
use 4.7
polyphonic, program description ERB 862
use 4.8

SPDTA 7.1

STOR KEY 4.9

SWITCH ? 4.3

TEMPO 4.4

10THPLAY commands B.3
program description 5.8
use 4.12

3RD PLAY 4.11

13THPLAY 4.11

TONE 2 commands B.1
program description ERB 862
use 4.13

TONE 2 commands B.1
program description ERB 862
use 4.9

TRAN program description 7.4

TRANSFRM 4.7

TRANSPOSE program description 7.1
use 4.8

TUNER program description 8.5
use 4.11

12THPLAY 4.11

24THPLAY 4.11

22NDPLAY commands B.5
- use 4.11

Utility programs use 4.10

VOICES ? 4.1

Waveforms format 3.1
- program description (FORM) 7.5
- use 4.9

WRITER Listing I
- program description 6.2
- use 4.2

WRITE+CH 4.2

4 WRITE 4.10