Orchestra-80

DISTRIBUTED BY

Software Affair, Ltd.
473 SAPENA CT. SUITE 1
SANTA CLARA, CA 95051

TABLE OF CONTENTS

INTRODUCTION

ORCHESTRA-80 is a unique combination of hardware and software that can turn any **16K TRS-80* level II** into a high quality musical instrument.

The hardware is a single printed circuit board that plugs into the expansion connector on the back of the **TRS-80** keyboard or the screen printer connector on the expansion interface. The board contains a precision digital-to-analog converter and associated electronics to change the binary computer output into a high fidelity audio signal. The output of the board is connected to the **AUX/TAPE/TUNER** input of any stereo or monophonic audio amplifier. (Cables not supplied.)

The software consists of five major components integrated into a single, easy to use, program.

1. Digital Synthesizer

The synthesizer has a six octave range and can produce up to four simultaneous notes or voices. Each voice can be assigned to any of four user-defined tone color registers. The default tone color registers mimmic the sound of a trumpet, oboe, clarinet and pipe organ. The registers are easily altered to imitate other instruments or to make strange new sounds.

2. Music Language Compiler

The music language was designed to allow the direct transcription of virtually any written music to a symbolic form used by the computer. Non-musicians will find the language simple to learn and easy to use since no previous musical training or knowledge is required. Yet, in spite of its simplicity, the language has all the features and capabilities required by the advanced computer musician.

The compiler will accept music written in any key or time signature and any note value within the synthesizer's range from whole notes to sixty fourth notes. Notes may be single, double or triple dotted and/or played as triplets. All forms of accidentals (single and double) are supported as are staccato, pizzicato and two forms of articulation. There is also the capability of repeats, second ends (with or without retard) and modulation.

3. Editor

A full function, 'full screen' text editor is provided to ease the task of entering and modifing music language programs. A full screen of text is viewable at all times and a blinking cursor can be positioned anywhere within the file. All cursor motion keys automatically repeat when held down. Additional functions include insert, delete character, insert, delete line, and global character string search.

4. File Manager

The file manager provides the orderly storing and retrieval of named program files on tape or disk.

5. Initialization

The initialization routines allow the user to alter the tone color registers, select between the standard four voice synthesizer and a higher resolution, three voice version, and choose either the standard (1.77 MHz) or enhanced (2.6 MHz) CPU clock rate.

Because all of the software components are integrated into a single program, a great deal of inter-module communication is possible. For example, when syntactical error is encountered during compilation, an error message is displayed and the editor is called to display the file with the cursor positioned exactly where the error was detected. The user simply corrects the error and returns to the compiler. There is also a multi-function command that allow a sequence of music programs to be read into memory, compiled and played, one after another.

ORCHESTRA-80 is available in versions for tape or disk. Both versions are supplied on cassette along with several sample music programs and a utility program to transfer music files between tape and disk. The minimum system required to run either version is a 16K level II.

*TRS-80 IS A TRADEMARK OF THE TANDY CORPORATION.

INSTALLING THE INTERFACE BOARD

This section describes how to connect the interface board to your **TRS-30**.

WARNING The **ORCHESTRA-80** interface board contains static sensitive devices that can be damaged by careless handling. Handle the board as little as possible. Always **TURN OFF** the **TRS-80** and **expansion interface** before connecting or removing the interface board. Always **TURN OFF** the **audio amplifier** before making or breaking its connection to the interface board.

Observing the warning above, connect the interface board to the **expansion connector** on the back of the **TRS-80** keyboard or to the **screen printer connector** on the side of the expansion interface. When correctly positioned the board will be pointing upward and the printing on the face of the board will be right-side up.

The audio socket on the board will mate with a standard 'RCA type' phono plug. (Audio cables with this type of plug are readily available at Radio Shack and most Stereo stores.) Connect one end to the phono socket on the interface board. Connect the other end to the **AUX, TAPE, TUNER** or other high level input of an amplifier or receiver. **DO NOT use the PHONO, MIC, or other low level input.** After all connections are secure, turn on the computer and the amplifier.

Certain machine instruction combinations, encountered during normal system operation, can trigger the interface board and may result in unexpected pops and clicks. Therefore, always keep the amplifier at minimum volume when not actually playing music.

There are no harmful voltages or currents on any of the exposed parts on the interface board. However common sense and the warnings above suggest that the board not be touched, particularly with metal objects, when the computer is turned on. Doing so may damage the components on the board and possibly the computer.

When disconnecting the interface board, be sure both the computer and amplifier are turned off.

4

TRANSFERRING THE PROGRAMS TO DISK

This section describes how to transfer the programs from the cassette to a disk. It applies to the disk version of the program only.

Insert and rewind the cassette containing the programs for the disk version of **ORCHESTRA-80**. Using the **TRSDOS** utility **TAPEDISK**, transfer the first two files on the cassette to a formatted disk. Consult the **TRSDOS** manual for general information of the use of **TAPEDISK**.

A typical **TAPEDISK** sequence follows:

```
DOS READY
TAPEDISK [Enter]                              (executes tapedisk utility)
?C [Enter]                                    (command to load
                                               from tape)
?F ORCH80/CMD:0 5575 6CEF 5575 [Enter]        (saves file on disk 0)
?C [Enter]                                    (load second file
                                               from tape)
?F ORCHCOPY/CMD:0 5575 5B69 5575 [Enter]      (save on disk 0)
?E [Enter]                                    (exit to Trsdos
DOS READY
```

The music files on the tape are recorded with a special format and must be read with **ORCHCOPY**. (That's one of the programs you just read in.) See the section describing **ORCHCOPY** for more general information.

A typical abbreviated **ORCHCOPY** sequence follows:

```
DOS READY
ORCHCOPY [Enter]
ORCHCOPY etc                                  (sign on and copyright notice)
TYPE THE LETTER etc                           (prompt and function menu)
> R [Enter]                                   (select read file function)
> T [Enter]                                   (source is tape)
ENTER INPUT FILE NAME > * [Enter]             (* means read next file)
LONE                                          (echos name of file read)
TYPE THE LETTER etc etc                       (function menu again)
> W [Enter]                                   (write the file just read)
> D [Enter]                                   (destination is disk)
ENTER OUTPUT FILE NAME >
LONE:0 [Enter]                                (save on disk 0)
TYPE THE LETTER etc etc                       (function menu again)
```

Repeat the sequence above three more times to transfer the music files named **SONATA**, **GIGUE** and **FUGUE**. After the last file is transferred, select function **Q** to quit and return to **DOS**.

CONFIGURING THE SOFTWARE

This section describes how to configure the software to your specifications

The **ORCHESTRA-80** program is extremely versatile but very complex. In order to minimize the size of the running program, and thereby maximize the memory available for music, certain functions are performed only once when the program is first executed

During this configuration phase you specify either the standard **four-voice synthesizer** or the higher resolution **three-voice** version. If your system has been modified for 2.66 MHz operation, you may instruct the program to take advantage of the faster clock. You may also modify the standard sine wave tables to produce different kinds of sounds.

The generation procedure consists of a programmed sequence of steps. At each step the program will ask a question and wait for your answer. The range of acceptable answers will be displayed with each question. After typing your answer, press the **ENTER** key. If your answer is accepted, the program will proceed to the next step. If not, the question will be repeated until an acceptable answer is obtained. Your answer should not contain any leading or embedded blanks. Pressing the **BREAK** key at any time will clear the input line and allow you to retype your answer.

If you have not already done so, connect the interface board to your computer. If you have a disk system, go to **STEP 3**. **the first steps apply to tape systems only**

STEP 0 Power up the system. Insert and rewind the cassette containing the programs for the tape version of **ORCHESTRA-80**

A typical loading sequence follows

MEMORY SIZE? [Enter]	(no reserved space needed)
RADIO SHACK LEVEL II BASIC	
READY	
>SYSTEM [Enter]	(the tape is system format)
***? ORCH8T [Enter]**	(name of tape version file)
***?/ [Enter]**	(begin execution)

When the program starts, the screen will be cleared, a copyright notice will appear on the bottom line and the first question in configuration dialog will appear in the middle of the screen with a blinking cursor on a line below it

STEP 1
DUPLICATE? <Y/N> At this point you may request the program to make a back up copy of itself. If you answer **N**, the program will proceed to **STEP 4**. If you answer **Y** continue to the next step

STEP 2
CASSETTE #2? <Y/N> At the previous step you requested the program to duplicate itself. If your system has two cassette decks and you wish the copy to be written on the auxiliary deck answer **Y**. Otherwise answer **N** and the standard cassette port will be used

The program will begin writing as soon as you press **ENTER**. be sure the cassette deck is ready. Once the file is written, the program will return to **STEP 1**

STEP 3 **This step applies to disk systems only.**

Power up the system and load the program. A typical loading sequence follows

---

DOS READY	
ORCH80 [Enter]	(load and execute the program)

When the program starts, the screen will be cleared, a copyright notice will appear on the bottom line and the first question in configuration dialog will appear in the middle of the screen with a blinking cursor on a line below it

STEP 4
FAST CLOCK? <Y,N>

There are two ways to activate the clock speed up modification hardware usually a switch, or software usually outputting a certain value to a certain port. If your speed up modification is controlled by software answer **Y**. Otherwise answer **N** and proceed to **STEP 8**.

STEP 6
ENABLE CODE?

To make optimum use of a software controlled clock modification the program needs to know the instructions to execute to enable the fast clock. Enter those instructions up to 8 bytes in hexadecimal.

One popular mod is enabled from **BASIC** with **OUT 254,1**. In assembly language it is:

```
LD    A,1
OUT   (254),A
```

The hex machine code would be **3E01D3FE**.

STEP 7
DISABLE CODE?

It is usually necessary to disable the fast clock before reading or writing to tape or disk. If you tell it how, the program will switch to the standard clock rate before doing any I/O. Enter those instructions up to 8 bytes in hexadecimal.

One popular mod is enabled from **BASIC** with **OUT 254,0**. In assembly language it is:

```
LD    A,0
OUT   (254),A
```

The hex machine code would be **3E00D3FE**.

If you wish to do I/O with a fast clock, enter a **NOP** instruction or **00**.

STEP 8
FOUR VOICES? <Y,N>

If you want the standard four voice capability answer **Y**. Otherwise answer **N** for the higher resolution three voice synthesizer. Music arranged for four voices can be played on a 3 voice system; the fourth voice is silent. Three voice systems also require less memory.

STEP 9
TEST SCALE? <Y,N>

If you answer **Y** the program will play a sample scale with each of the four different tone colors (registers **A**, **B**, **C** and **D**). This step will be repeated until you answer **Y** to proceed to the next step.

The next five steps allow you to modify the wave forms of the different registers. If you are not already familiar with the use of the system, answer the next step **N** to bypass the wave form dialog.

STEP 10
ALTER SINE TABLES? <Y,N>

If you answer **Y**, the program will proceed to the next step where you are required to define a new register. Answer **N** to proceed to **STEP 15**, the end of the system configuration procedure.

STEP 11
REGISTER NAME? <A—D>

Enter the name of the register you wish to modify.

STEP 12
NUMBER OF PARTIALS? <1—16>

The wave form of a register is defined by a sum of sine waves or partials. The frequency of each partial is an integer multiple of the fundamental frequency. Partial # 1 is the fundamental, partial # 2 is the first harmonic and is two times the fundamental frequency, partial # 3 is the second harmonic and is three times the fundamental, etc. Enter the number of the highest partial you wish to define.

Unless you are after special effects, the number of partials should not exceed 4 for a 1.77 Mhz clock, or 6 for a 2.66 Mhz clock. This will keep aliasing and high frequency distortion to a minimum. Refer to the How It Works section for a complete explanation.

STEP 13
PARTIAL #01, WEIGHTING FACTOR? <0—255>

This step will be repeated for each partial to be defined. Enter a number that represents the relative strength of the named partial. A partial with a weight of 200 will be twice as prominent (twice as loud) as a partial with a weight of 100. A partial with a weight of 0 makes no contribution to the final wave form. All weights are normalized so that a register defined with all partials of weight 1 will be exactly the same as a register defined with all partials of weight 255.

STEP 14
VOLUME? <1—256>

Enter the overall amplitude of the register. The higher the number, the louder it will be.

The program will now generate the register you have just defined. When it has finished the calculations (it may take several seconds), the program will return to **STEP 9** to demonstrate the results.

STEP 15
SAVE PROGRAM? <Y/N>

The configured program is now in memory and is ready for use. Answer **Y** if you wish to save this configured program image and avoid the lengthy dialog in the future. Otherwise, answer **N** to begin execution of the program: **STEP 18**

STEP 16
PROGRAM NAME?

Enter the file name to be used to save the program. A three voice synthesizer based on a fast clock might be named **ORCH3F**

TAPE version: maximum length is six characters.

DISK version: maximum length is eight characters followed by drive number; the extension **/CMD** will be supplied by the program; ie: **ORCH3F:0**

STEP 17

This applies to **TAPE** systems only.

See **STEP 2** for explanation. Be sure to replace the music cassette before proceeding to the next step.

STEP 18

If you have not used the **ORCHESTRA-80** before, you are probably anxious to hear what it can do. Type:

GET LONE [Enter]

to load and play the music called **LONE**. In a few seconds you'll be on your way. Hi-yo Silver, away!

USING THE SYSTEM

This section describes the command structure.
First, some definitions

Command Line	The first line on the video screen
Status Line	The second line
Editor	Used by the remainder of the screen

There are two modes of program operation: **Command Mode** and **Edit Mode**.

Command Mode	Distinguished by a blinking line cursor on the top line of the video screen
Edit Mode	Distinguished by a blinking block cursor in the middle of the screen

In either **Command** or **Edit Mode** the following control keys are available:

BREAK	Stops current process and enters **Command Mode**
CLEAR	Erases the character at the cursor and everything to the right of it on the line
ENTER	Accepts the current line as command or edit input
Left Arrow	Moves cursor to the left
Right Arrow	Moves cursor to the right Note: Left and right cursor movement will wrap around at the logical end of line
Shift Left Arrow	Deletes the character under the cursor and shifts the remainder of the line to the left. The cursor does not move
Shift Right Arrow	A space is inserted under the cursor by shifting the character at the cursor and the remainder of the line to the right. The cursor does not move and any character shifted off the end of the line is lost

In **Edit Mode** these additional control keys are available:

Up Arrow	Moves the cursor to the line above the current line towards the top or beginning of the file
Down Arrow	Moves the cursor to the line below the current line, towards the bottom or end of the file
Shift Up Arrow	Joins the current line to the one above it making one long line. Effective only at the beginning of a line
Shift Down Arrow	Forms a new line beginning with the character at the cursor
Question Mark	Makes a copy of the current line and inserts it at the end of the file
Exclamation Mark	Moves the current line up one position towards the top or beginning of the file

The latter two control keys are used to rearrange and or duplicate parts of a file.
All keys will repeat at a rate of about ten per second if held down for more than a half second.

COMMAND MODE

All system commands consist of a keyword optionally followed by one or more operands. Operands are separated from each other and from the keyword by one or more spaces. The keyword may be abbreviated (only the first character is significant) but may not contain any

leading or embedded spaces. Unrecognized commands are answered with a ? in the status line.

NEW Delete current file in memory, reset all pointers and begin new file.

TOP Move file pointer to the top or beginning of the current file and display it.

BOTTOM Move file pointer to the bottom or end of the current file and display it.

EDIT Enter **Edit Mode** with current file pointer.

/<string> Where **<string>** is any character string. Search the entire file for a matching character string starting with the character to the right of the file pointer. If a match is found, position the file pointer there. If no match is found, display a **?** in the status line and file pointer does not move.

READ filename **TAPE version:** The tape is searched for a matching filename. When found, the file is read into memory, replacing any previous file there. An asterisk (*) in the file name will match any character. **READ *** will read the next file on the tape, regardless of its name. **READ xx*** will read the next file with a name beginning with **xx**. During the search, the name of each file encountered will be displayed in the status line. The operation may be terminated at any time by holding down the **BREAK** key. (The tape must be non-blank and moving.)

DISK version: The disks are searched for a matching filename. When found, the file is read into memory, replacing any previous file there. If not found, an error code is displayed in the status line. The filename may be followed by a drive number. The file extension is always **/ORC**.

WRITE filename **TAPE version.** The current file is written with the filename given.

DISK version. The current file is written with the filename given. The filename may be followed by a drive number. The file extension is always **/ORC**.

DRIVE n **TAPE version only.** Allows switching between the standard (**DRIVE 1**) and alternate (**DRIVE 2**) cassette deck for file operations.

DIR n **DISK version only.** Displays directory of disk **n** (**0,1,2,3,**) listing all files with an extension of **/ORC**.

QUIT Returns to **TRSDOS** (**DISK**) or **LEVEL II BASIC** (**TAPE**).

LIST Performs a **Form Feed** on the line printer and prints the current file beginning with the current line. Printing may be terminated at any time by holding down the **BREAK** key.

SCORE **Compiles** the current file into the binary form required by the synthesizer. After successful compilation an internal **TOP** function is performed. If an error is encountered, compilation is terminated, an error code is displayed in the status line and the file is positioned with the file pointer at the place where the error was detected.

If there is not enough memory to hold both the source file and the compiled object code at the same time, the program will pause and display the question **OVERLAP?** in the **status line**. If you answer **Y**, compilation will resume and portions of the source file will be **OVERLAPed** by the object code. Answering anything else will cancel the compilation and enter the **command mode.**

This function is useful when compiling large files as the end of the object code can **OVERLAP** the beginning of the source code. **DO NOT OVERLAP** a file that has not been saved on disk or tape since this function implies the probable destruction of the current file. Do not expect to be able to use it in any way once this command is given.

PLAY nn Directs the synthesizer to **PLAY** the most recently **SCORED** object code starting with **PART nn**. If **nn** is omitted, the entire piece is played. An error message will be displayed if the **PART** cannot be found or if the text file was not recently **SCORED**.

The numeric keys on the keyboard can be used to control the operation of the synthesizer while a piece is being **PLAYED**. Keys 1-7 control the **TEMPO**. Holding

down any combination of these keys form a binary weighted value which replaces the default tempo programmed in the music. Key 7 is the most significant bit and key 1 is the least. See the section on Tempo Conversion for more details.

The 0 key will stop the synthesizer and enter command mode.

GET file1 file2 file3... This is a multi function command. It will perform a **READ SCORE** and **PLAY** for each of the files named. **OVERLAP** will be performed automatically if necessary.

DISK version Filenames may be followed by a drive number just as in the **READ** command.

TAPE version Filenames may contain * as in **READ**. In addition, an at-sign (@) will match any character, repetitively. **GET @** will **READ SCORE** and **PLAY** all files on the tape.

EDIT MODE

Perhaps the easiest way to learn how to use the **Text Editor** is to experiment with it. Review operation of the cursor control keys. Since all the keys repeat when held down, try to use a light touch; the keyboard is fully debounced. Accidentally holding down the **CLEAR** key can erase an entire file in a few seconds!

Notice that:

1. The cursor always remains on the same screen line and that the file moves up and down around it.
2. The insert character function (**Shift Right Arrow**) adds spaces to the current line and will shift characters off the end of the line.
3. The **Editor** always knows how long each line is and will not go beyond the last character.
4. A **Shift Up Arrow** will un do only one previous **Shift Down Arrow**.

CONTROL KEY	FUNCTION
BREAK	Return to **command mode**
CLEAR	Erase to end of line
ENTER	Accept the current line
Left Arrow	Move cursor left
Right Arrow	Move cursor right
Shift Left Arrow	Delete character
Shift Right Arrow	Insert space
Up Arrow	Move cursor up
Down Arrow	Move cursor down
Shift Up Arrow	Join line
Shift Down Arrow	New line
Question Mark	Duplicate line
Exclamation Mark	Move line up

COMMAND	OPERAND	FUNCTION
/	**<any string>**	Search for next occurance of string
BOTTOM	(none)	Position file pointer at the end of the file
DIR	**0 1 2** or **3**	List all **/ORC** files on the disk named in the operand. Default = **0**
DRIVE	**1** or **2**	Use the cassette deck named in the operand for future tape operations
EDIT	(none)	Enter **Edit mode**
GET	**filename ...**	**Read Score** and **Play** each of the files named in the operand
LIST	(none)	Print the current file on the printer
NEW	(none)	Erase current file
PLAY	**Part number** or **blank**	Play the current piece starting with the part number in the operand or at the beginning if no operand
QUIT	(none)	Return control to **TRSDOS** or **LEVEL II**
READ	**filename**	Read the named file into memory
SCORE	(none)	Compile the current file
TOP	(none)	Position the text pointer at the beginning of the file
WRITE	**filename**	Write the current file with the name in the operand

TRANSCRIBING MUSIC

This section introduces the notation of the music language and compares it with standard musical notation.

The music language provides the means by which standard musical notation is transcribed into a symbolic form usable by the computer.

The use of the language can be shown best by example **Figure 1** is an excerpt from **J. S. Bach's Capriccio Figure 2** is the same music transcribed for a three-voice synthesizer. While the two figures may appear to have little in common, they both contain essentially the same information.

In **Figure 2** each line begins with a three digit number followed by a space. These line numbers are added for the purposes of this discussion only and have no bearing on the music. Remember, the contents of each line effectively begin after the first space

Lines 10, 20 and 30 are informational only. A slash (/) appearing anywhere in a line causes the remainder of that line to be ignored.

For the purposes of transcription, a piece of music is considered to be subdivided into **parts**, **measures** and **voices**. A **part** defines one or more **measures** that are played in the same key, tempo and with the same registration and usually correspond to a portion of the piece that may be repeated, i.e. a phrase, stanza, chorus etc. **Parts** are defined by the letter **P** followed by either a two digit number or a space. Numbered **Parts** can be repeated; un-numbered **Parts** cannot.

Measures are indicated by a character string beginning with the letter **M** and ending with a space. The characters following the **M**, usually a **measure number**, are ignored by the compiler and only serve as a reference between the printed musical score and the transcribed music language text.

A **voice** is a separate 'strand' of music, in harmony or counterpoint. A trio has **three voices** and a quartet **four**. Up to four **voices** may be defined and they are identified by **V1**, **V2**, **V3** and **V4**. As a convenience, each **measure** begins with an implied **V1**. Music arranged for four voices can be played by a three-voice synthesizer except **Voice 4** will be silent.

In the example the Aria consists of two repeated sections, one 5 measures long and the other 7 measures long. The first section is defined in line 40 as **P50**. The choice of the two-digit number is arbitrary and serves only to identify the part. However, each **part** defined must have a **unique number**. Line 200 defines a **Repeat** of **Part 50**. Likewise, line 390 is a **Repeat** of **Part 52** (line 210).

Line 50 defines the **key signature**. You need only specify the number of **sharps** (**#**) or **flats** (**&**); the compiler will know which notes are affected. If the key signature is not specified, C-major (no sharps or flats) is assumed.

Line 60 defines the **time signature** and **tempo**. **NQ** indicates that each **quarter note** gets a **beat** (**H**=half, **Q**=quarter, **I**=eighth, **S**=sixteenth). =**C0** indicates the relative length of a beat. Normally, the tempo parameter is determined experimentally by holding various combinations of 1-7 keys while the piece is playing. The key pattern is then translated, using the **Tempo Conversion Tables**, to a two digit number which is entered into the source file. It may also be necessary to alter the time signature parameter to achieve to desired tempo.

Line 62 defines **transposition**. The character **<** or **>** followed by a number defines the direction and number of **semi-tones** the piece is to be transposed (**<=down, >=up**). It is common practice to transpose a piece of music as it is transcribed to accommodate the range and fingering of the new instrument. While **ORCHESTRA-80** should not pose any 'fingering' problems, it is often desirable to transpose a piece down a few semi-tones to avoid the distortion and aliasing present in the higher notes. This is especially true with 1.77 MHz, 4-voice synthesizers.

Line 64 defines the **tone color registers** to be used by the different **voices**. **Voice 1** uses register **C** (clarinet), **voice 2** and **3** use register **B** (oboe) assuming, of course the registers have not been altered during the configuration phase. The default register is **D** (organ).

The music proper begins in line 70 with the definition of **measure 1**.

Each of the symbols representing a note in standard musical notation imply two pieces of information:

1. **Its shape, along with the time signature, defines how long, relative to a beat, the note is to be held.**
2. **Its position on the staff, along with the key signature and clef define the note to be played.**

Transcribing this two-dimensional form into a single line requires two characters to represent each note. The **time value** or **shape**, is defined by a single letter. **W=whole, H=half, Q=quarter, I=eighth, S=sixteenth, T=thirty-second, X=sixty-fourth.** The **time value** may be modified by additional symbols. **Dotted notes** are indicated by a **period** (.) following the letter. **Triplet** time values are indicated by a **colon** (:) following the letter. e.g. **Q** means all the notes following are **quarter notes**. **I.** means all the notes following are **dotted eighth notes**. **S:** means all the notes following are **sixteenth note triplets**. Note length modifiers may be combined. **H:..** means all the notes following are **double dotted half note triplets**. (Dot modifiers cannot precede a triplet modifer, i.e. **Q.: is not valid.**)

The **staff position** of a note is defined by its relationship to a fixed point on the staff, **Middle-C**. Middle-C is always location **0** and notes above it are defined by a positive (+) displacement while notes below it are defined by a negative (−) displacement. See **Figure 3**. The displacement uses a hexadecimal-like number scale with the letters **A** to **G** representing the numbers **10** to **16**. The maximum positive displacement is **+G**, the maximum negative displacement is **−F**. A **dollar sign** (S) defines a **rest**.

Coding note displacements can be simplified by specifing a default displacement sign or clef. An **asterisk** (*) sets the default to **+** or **treble clef**, and all unsigned notes following are assumed to be **+**. An **at-sign** (@) sets the default to **−** or **bass clef**, and all unsigned notes following are assumed to be **−**.

Accidentals are indicated with a **sharp** (#), **flat** (&) or **natural** (%) sign immediately following the note affected. Accidentals stay in effect until the end of the measure or are modified by another accidental. **Double sharps** (##) and **flats** (&&) and **natural sharps** (**#% or %#**) and **flats** (**&% or %&**) are allowed. **Double naturals** (%%) and **flatted sharps** (**&#**) or **sharp flats** (**#&**) will give unpredictable results and should be avoided.

You should now be able to decipher all the symbols in lines 70, 80 and 90 and relate them to the first measure in **Figure 1**.

Line 100 introduces several new symbols. The **paranthesis** indicate **reiterative compilation**. The number following the right paranthesis is the reiteration **count** which tells the compiler how many more times to compile the bounded sequence. Line 100 will be compiled as if it were written:

```
M2 *I6;SD6^16;SD6^16;SD6^16;SD6^16;SD6^
```

Reiteration should not be confused with Part **repetition.** Reiteration occurs at compile **time**; Part repetition directs the synthesizer to re-play a particular Part. Reiteration blocks may not be nested.

Notes may be further modified by **'expression'** modifiers which alter the way a note is played, but not it's pitch or overall duration, by **replacing a portion of the note with a rest.** There are two major types of expression, **staccato** and **articulation,** and each type has two forms, **long** and **short**.

Short staccato indicated by a **comma** (,), shortens the note by ½ and **adds a rest equal to ½ the notes duration Long staccato** indicated by a **semi-colon** (;), is similar except the note is **shortened by ¼ of its value and adds a ¼ rest. Articulation** is used to introduce a **small rest** after a note to **separate** it from the note following. **Short articulation**, indicated by an **apostophe** ('), **shortens the note by an amount equal to ⅛ of a 1/128 note and adds a rest of that value Long articulation**, indicated by a **quote** ("), is similar except **the amount shortened is double**, **⅔ of a 1/128 note**. In all cases, expression modifiers effect only the note they follow. Accidentals must precede expression modifiers.

Measure 5 (lines 170-190) presents a very common problem: **too many voices.** An experienced musician will probably have no trouble, for the rest of us, it will be a matter of blind luck or painstaking trial and error deciding which notes to use and which to ignore. Ultimately, the 'right' solution is the one that sounds best.

Thus far, the assumption has been that the music being transcribed is in bass 'treble clef notation. To take advantage of music arranged for different instruments in different clefs (soprano, alto, tenor etc.), each voice can be defined as belonging to a different clef. Voice clef definition is indicated by a **U** followed by a number that is the displacement from Middle-C in the clef being defined to Middle-C in the treble clef. For example:

```
V1 U-2 V2 U-6 V3 U-8 V4 U-C
```

defines **Voices 1-4** as **soprano alto tenor** and **bass**, respectively. Because all clefs are defined relative to the treble clef, each voice is transcribed exactly as if it were the treble clef, and no clef symbols (* or @) need be used.

Another useful transposition would be **V4U-7** to **lower Voice 4 one octave** to get a low bass sound.

Often, when transcribing part-music, the accidentals in one voice interfere with the accidentals in another voice. This is because the compiler applies every accidental to all voices. **OPTION 1**, specified **01**, will limit the application of accidentals to the voice in which they appear.

COMPILER ORGANIZATION

This section describes certain features of the Compiler in more detail. A little bit of technical information will help in understanding how the compiler works and how certain parameters are interpreted.

The output of the Compiler consists of two lists, a Part List and a Note List. Each element in the Part List contains a pointer to a Note List element and parameters, such as tempo and registration, that describe how the Note List is to be played. Each element in the Note List contains note frequencies and durations for each of three or four voices.

During compilation, the source file is processed and the data collected is placed into either a Part Buffer or a Note Buffer. At the appropriate time, the contents of the Buffers are processed into the corresponding List.

When compilation begins all Buffers are cleared and the default values are entered. If written out, the default values would look like this:

P00	un-numbered part
K0#	key of C-major no sharps or flats
<0	no transposition up or down
NQ	time signature based on a quarter note
N60	each beat has a value of 60
V1 YD U0	voice 1 uses register D no clef transposition
V2 YD U0	voice 2 uses register D no clef transposition
V3 YD U0	voice 3 uses register D no clef transposition
V4 YD U0	voice 4 uses register D no clef transposition
M V1 *	begin a measure with voice 1 in treble clef

The contents of the Part Buffer are added to the Part List whenever a PART or REPEAT symbol is processed or the end of the file is reached. The contents of the Note Buffer are added to the Note List whenever a MEASURE, PART or MEASURE symbol is processed or the end of the file is reached.

Referring to Figure #2, the PART symbol in line 40 will cause the contents of the Part Buffer to be added to the initially empty Part list and the Note Buffer to be added to the initially empty Note List. A new Note List element is started and the Note list pointer in the Part Buffer is changed to point to it.

Lines 50 to 64 modify most of the default values in the Part Buffer. In line 70, the MEASURE symbol will cause the Note Buffer to be processed and the Note frequencies to be transferred to the Note List. The Note Buffer is cleared and any accidentals are reset. (Since the Note Buffer is empty, nothing of interest has happened yet.) The notes defined in lines 70, 80 and 90 are then placed in the Note Buffer.

The MEASURE symbol in line 100 causes the Note Buffer to be processed using the key signature and transposition parameters in the Part Buffer, and the calculated note frequencies of Measure 1 are transferred to the Note List. The Note Buffer is cleared, accidentals reset and the rest of the line is transferred to the Note Buffer.

This process of accumulating notes in the Note Buffer and processing them only when a new measure is defined continues until line 200. The REPEAT symbol causes the Note Buffer to be processed and Measure 5 is added to the Note List. It also causes the Part Buffer to be added to the Part List and a new Note List element to be started. The Note List pointer in the Part Buffer is changed to point to the same Note List element that PART 50 pointed to.

The PART symbol in line 210 acts much the same as that in line 50. Similary, line 390 is like line 210 and the end of the file forces out the remaining buffer contents.

The result of the compilation is a Note List with the following elements

<1> [illegible]
<2> measures 1 2 3 4 5
<3> [illegible]
<4> measures 6 7 8 9 10 11 12
<5> [illegible]

and a Part List with elements and Note List pointers as follows

<00> points to <1>
<50> points to <2>
<00> points to <2>
<52> points to <4>
<00> points to <4>

Notice that Note List elements 3 and 5 are not pointed to. These are the elements created when the REPEAT symbol is processed. Any notes defined between a REPEAT and a PART symbol are processed into one of these 'un-claimed' elements and are never played. In all other respects, REPEATs are like un-numbered PARTs.

REPEATs can alter any of the parameters in the Part Buffer and thereby repeat sections of music with a different tempo and or registration. For example, the demonstration scale played during program configuration can be duplicated with the following:

```
P01 YA S0123432108     register A
R01 YB                 register B
R01 YC                 register C
R01 YD                 register D
```

The default values **NQ=60** and **V1** * are assumed

SUMMARY

Parameters take effect when data is processed from the buffer to the list. All parameters remain in effect until changed, explicitly or by default.

Parameters affecting the Key Signature (**K** and **O**) or Transposition (**<**, **>** and **U**) take effect at the beginning of the current Measure and stay in effect until changed.

Accidentals take effect immediately and stay in effect until changed or the end of the current Measure. (Accidentals are also reset by **K**).

Parameters affecting the tempo (**N** and **=**) or Registration (**Y**) take effect at the beginning of the current Part and remain in effect until changed.

Parameters affecting the Clef (* and @) or Note duration (**W**, **H**, **Q**, **I**, **S**, **T**, or **X** with or without modifiers. . or **:**) take effect immediately and remain in effect until changed.

Parameters selecting the current Voice (**V**) take effect immediately and remain in effect until changed or the beginning of the next Measure. Measures always begin with a default **V1**.

MUSIC LANGUAGE SUMMARY

The Music Language is processed a single free format, string of symbol groups. All spaces and line boundries between symbol groups are ignored. A symbol group consists of a single character symbol and is optionally followed by one or more symbol modifiers.

Unless otherwise specified, all numeric data is hexadecimal

NOTE SYMBOL	MUSICAL NOTE NAME	STAFF POSITION (see Figure 3)
+G	E	
+F	D	
+E	C	
+D	B	
+C	A	
+B	G	
+A	F	
+9	E	
+8	D	
+7	C	
+6	B	
+5	A	
+4	G	
+3	F	
+2	E	
+1	D	
0	C	Middle C
-1	B	
-2	A	
-3	G	
-4	F	
-5	E	
-6	D	
-7	C	
-8	B	
-9	A	
-A	G	
-B	F	
-C	E	
-D	D	
-E	C	
-F	B	
S	REST	(any)

NOTE MODIFIER	MUSICAL EXAMPLE	NAME
#	♯	ACCIDENTAL SHARP
&	♭	ACCIDENTAL FLAT
%	♮	ACCIDENTAL NATURAL
# #	♯♯	DOUBLE SHARP
&&	♭♭	DOUBLE FLAT
%#	♮♯	NATURAL SHARP
%&	♮♭	NATURAL FLAT

EXPRESSION MODIFIER	EXAMPLE	NAME
, (comma)		SHORT STACCATO
; (semicolon)		LONG STACCATO
' (apostrophe)	(none)	SHORT ARTICULATION
" (quotation mark)	(none)	LONG ARTICULATION

TIME VALUE SYMBOL	MUSICAL EQUIVALENT	NAME OF NOTE	REST ($)
W		WHOLE note	
H		HALF note	
Q		QUARTER note	
I		EIGHT note 	
S		SIXTEENTH note 	
T		THIRTY-SECOND note 	
X		SIXTY-FOURTH note 	

TIME VALUE MODIFIER	MUSICAL EQUIVALENT	NAME
• (period)		DOTTED note
: (colon)		TRIPLET

SYMBOL	MODIFIER	DEFINITION
(	(none)	Begin REITERATION
)	**hex digit**	REITERATE the number of times specified by the modifier
*	(none)	Unless otherwise indicated, all the NOTES following are assumed to be + or TREBLE clef
/	(none)	Comments. The remainder of the line is ignored
<	**hex digit**	All the NOTES in the current and following MEASURES are transposed DOWN the number of semitones specified by the modifier
=	**2-digit hex**	Set TEMPO to value of modifier
>	**hex digit**	All the NOTES in the current and following MEASURES are transposed UP the number of semitones specified by the modifier
@	(none)	Unless otherwise indicated, all the NOTES following are assumed to be – or BASS clef
K	**digit 0–7** and **#** or **&**	Define KEY SIGNATURE by number and type specified by the modifier
M	**any string** and **a space**	Define the beginning of a MEASURE. The current MEASURE is ended; ACCIDENTALS are dropped; the KEY SIGNATURE is restored
N	**H Q I S** or **T**	Define TIME SIGNATURE; set NOTE type in modifier to one BEAT
O	**hex digit**	Set the OPTIONS to the value of the modifier
P	**2-digit hex**	Defines the beginning of the PART named by the modifier. The current MEASURE is ended; The current PART is ended
R	**2-digit hex**	REPEAT the PART named by the modifier. The named PART must be previously defined. This symbol group may be followed by a TEMPO group and or REGISTER group
U	**signed hex**	TRANSPOSE all the NOTES following that belong to the current VOICE up or down the number of whole steps indicated by the modifier
V	**1 2 3** or **4**	All the NOTES following are added to the previous NOTES of this MEASURE belonging to the VOICE named by the modifier
Y	**A B C** or **D**	Set the current VOICE in the current PART to the REGISTER specified by the modifier

HOW IT WORKS

The synthesizer uses a sampling technique commonly used in professional digital synthesizers

The Sampling Theorem tells us that any wave form, no matter how complex, can be reconstructed from a rapid succession of discreet voltages. The reconstructed wave form will actually be a stepped approximation of the original but if the steps are small enough they will not be noticed. The size of the steps is determined by the sample rate and the frequency of the wave being reproduced.

Consider one cycle of a simple sine wave. If we were to measure its amplitude at 100 equally spaced intervals and record the values in a table we could reproduce it by serially setting a voltage equal to each entry in the table. Since the table represents exactly one cycle, the frequency of the synthesized wave will be equal to the sample rate divided by 100 (the number of entries in the table). At 1000 samples per second the entire table will be accessed 10 times per second, producing a 10 Hertz sine wave. doubling the sample rate will double the frequency of the synthesized sine wave

Using the same table and sample rate we could effectively double the frequency of the synthesized wave by accessing every other table entry. This gets us through the entire table twice as fast. Similarly, taking every third or fourth table entry will triple or quadruple the frequency. This is the basis of the synthesizer, except to get musically accurate frequencies, it is necessary to skip a fractional number of table entries

Theoretically, the highest frequency that can be synthesized is equal to half the sample frequency. This is called the Nyquist frequency. As this limit is approached, the synthesized wave form becomes more and more distorted since there are fewer samples in each cycle. As the distortion increases, a secondary tone, or alias, is produced. If the Nyquist frequency is exceeded, the synthesized frequency is replaced by its alias. This phenomenon is not limited to the fundamental frequency. It affects each component of the synthesized wave form

Aliasing cannot be filtered out, it can only be avoided. If aliasing is a problem, redefine the registers with fewer partials or transpose the entire piece down a few semi-tones. An alternative solution would be to speed up your CPU

ORCHCOPY UTILITY

This section describes the music file copy utility program and applies to the **disk version only** The file structure used by **ORCHESTRA-80** is unique and music files cannot be transferred to or from tape by the usual means.

ORCHCOPY is a self-prompting, menu-driven program that must be used to transfer music files to or from tape or disk. To use it, simply type the letter corresponding to the function desired.

Tape file names may be from one to six characters long. An asterisk (*) in the file name will match any character(s). Therefore, requesting **ORCHCOPY** to read a tape file named * will read the next file on the tape.

Disk file names may be from one to eight characters long and may be followed by a drive number, i.e., **LONE:1** File extensions are not allowed as the program will supply an extension of **/ORC**.

The program will respond with an error message when the requested function could not be completed successfully.

UNRECOVERABLE I/O ERROR means **TRSDOS** rejected or otherwise did not complete the requested disk operation. Either the file name is illegal or contains an invalid character, the file was not found, the disk or directory is full or some other error occurred.

The other error messages are self-explanatory.

ERROR MESSAGES

When an error is detected an error number will be displayed and, whenever possible, the file positioned with the file pointer at, or near, the place where the error was encountered.

ERR 1 **MEMORY OVERFLOW**	There is not enough memory to **SCORE** the current file even after **OVERLAP**ing There is not enough memory to **EDIT** the current file. Consider expanding your system or splitting the file into two or more smaller sections. There is not enough memory to **READ** or **GET** the requested file. Expand the memory in your system.
ERR 2 **SYMBOL OUT OF CONTEXT**	The command operand contains an invalid hexadecimal digit.
	The **TEMPO** symbol = is not followed by two hexadecimal digits.
	The compiler was expecting a hexadecimal digit or a note symbol (**0-9** or **A-G**) but none was found.
ERR 3 **PARAMETER ERROR**	The number of sharps or flats in the **KEY signature** is too large or is not followed by **SHARP** (**#**) or **FLAT** (**&**) symbol.
	The note value in the **TIME** signature is not **H Q I S** or **T**.
	The **VOICE** specified is not **1 2 3** or **4**.
	The **REGISTER** specified is not **A B C** or **D**.
ERR 4 **INVALID PART NUMBER**	The **PART** or **REPEAT** specified is not a two digit hexadecimal number or the **PART** is not blank. **P00** is considered an **un-numbered part**.
	The **PART** is already defined.
	The **REPEAT** is not previously defined.
	The piece cannot be **PLAYED** because the **PART** cannot be found or because the piece needs to be **SCORED**.
ERR 5 **MEASURE OVERFLOW**	The measure being compiled contains more than 64 notes per voice. Split the measure in half and carry forward any accidentals.
	Note: Staccato, pizzicato and articulation generate a note and a rest so count them as two notes.
ERR 6 **NOTE OVERFLOW**	Dotted whole notes, 7 dotted triplets and other unusual combinations cannot be compiled.
ERR 7 **FILE I/O ERROR**	The host operating system reported an I/O or other error during the last file operation.
	READ or **GET**: File not found.
	WRITE: Disk or directory full.
	READ WRITE GET or **DIR**: Unrecoverable I/O error.
	DRIVE or **DIR**: Invalid device number.

STANDARD REGISTERS

The standard tone color registers are based on the spectral analysis of orchestral instruments and are generated with the following recipes. (Systems running with the standard clock rate of 1.77 MHz do not use partials 5 and 6.)

	Partial Weighting Factor						
Register	1	2	3	4	5	6	Vol.
A trumpet	224	240	240	160	80	64	224
B oboe	64	128	240	128	240	32	240
C clarinet	224	0	80	0	240	0	160
D organ	240	64	0	128	0	0	176

TEMPO CONVERSION TABLE

The numeric keys 1-7 are used to set or alter the tempo of a piece while it is playing. When none of the keys are depressed the piece will play at the tempo defined when it was compiled. Otherwise, the binary weighted value of the keys held down is used to set the tempo. The following table shows the hexadecimal and decimal value for each key combination.

Once the correct tempo has been established by experimentation, the corresponding hexadecimal value should be transferred to the source file. Tempo settings below **80** (hex) may cause an undesirable shift in frequency and should be avoided. It may be necessary to adjust the time signature parameter to get the tempo between **80** and **FE**. Increasing the note value of the time signature will allow you to double the value of the tempo, e.g., **NQ=C0** is preferable to **NI=60.**

The **0** key is used as a stop switch. When that key is depressed, the synthesizer will stop and the program will enter the command mode.

DIGITS	HEX	DEC	DIGITS	HEX	DEC	DIGITS	HEX	DEC
1234567	**FE**	254	12345-7	**BE**	190	123456-	**7E**	126
-234567	**FC**	252	-2345-7	**BC**	188	-23456-	**7C**	124
1-34567	**FA**	250	1-345-7	**BA**	186	1-3456-	**7A**	122
--34567	**F8**	248	--345-7	**B8**	184	--3456-	**78**	120
12-4567	**F6**	246	12-45-7	**B6**	182	12-456-	**76**	118
-2-4567	**F4**	244	-2-45-7	**B4**	180	-2-456-	**74**	116
1--4567	**F2**	242	1--45-7	**B2**	178	1--456-	**72**	114
---4567	**F0**	240	---45-7	**B0**	176	---456-	**70**	112
123-567	**EE**	238	123-5-7	**AE**	174	123-56-	**6E**	110
-23-567	**EC**	236	-23-5-7	**AC**	172	-23-56-	**6C**	108
1-3-567	**EA**	234	1-3-5-7	**AA**	170	1-3-56-	**6A**	106
--3-567	**E8**	232	--3-5-7	**A8**	168	--3-56-	**68**	104
12--567	**E6**	230	12--5-7	**A6**	166	12--56-	**66**	102
-2--567	**E4**	228	-2--5-7	**A4**	164	-2--56-	**64**	100
1---567	**E2**	226	1---5-7	**A2**	162	1---56-	**62**	98
----567	**E0**	224	----5-7	**A0**	160	----56-	**60**	96
1234-67	**DE**	222	1234--7	**9E**	158	1234-6-	**5E**	94
-234-67	**DC**	220	-234--7	**9C**	156	-234-6-	**5C**	92
1-34-67	**DA**	218	1-34--7	**9A**	154	1-34-6-	**5A**	90
--34-67	**D8**	216	--34--7	**98**	152	--34-6-	**58**	88
12-4-67	**D6**	214	12-4--7	**96**	150	12-4-6-	**56**	86
-2-4-67	**D4**	212	-2-4--7	**94**	148	-2-4-6-	**54**	84
1-4-67	**D2**	210	1-4--7	**92**	146	1-4-6-	**52**	82
-4-67	**D0**	208	---4--7	**90**	144	---4-6-	**50**	80
123--67	**CE**	206	123---7	**8E**	142	123--6-	**4E**	78
-23--67	**CC**	204	-23---7	**8C**	140	-23--6-	**4C**	76
1-3--67	**CA**	202	1-3---7	**8A**	138	1-3--6-	**4A**	74
--3--67	**C8**	200	--3---7	**88**	136	--3--6-	**48**	72
12---67	**C6**	198	12----7	**86**	134	12---6-	**46**	70
-2---67	**C4**	196	-2----7	**84**	132	-2---6-	**44**	68
1----67	**C2**	194	1-----7	**82**	130	1----6-	**42**	66
-----67	**C0**	192	------7	**80**	128	-----6-	**40**	64

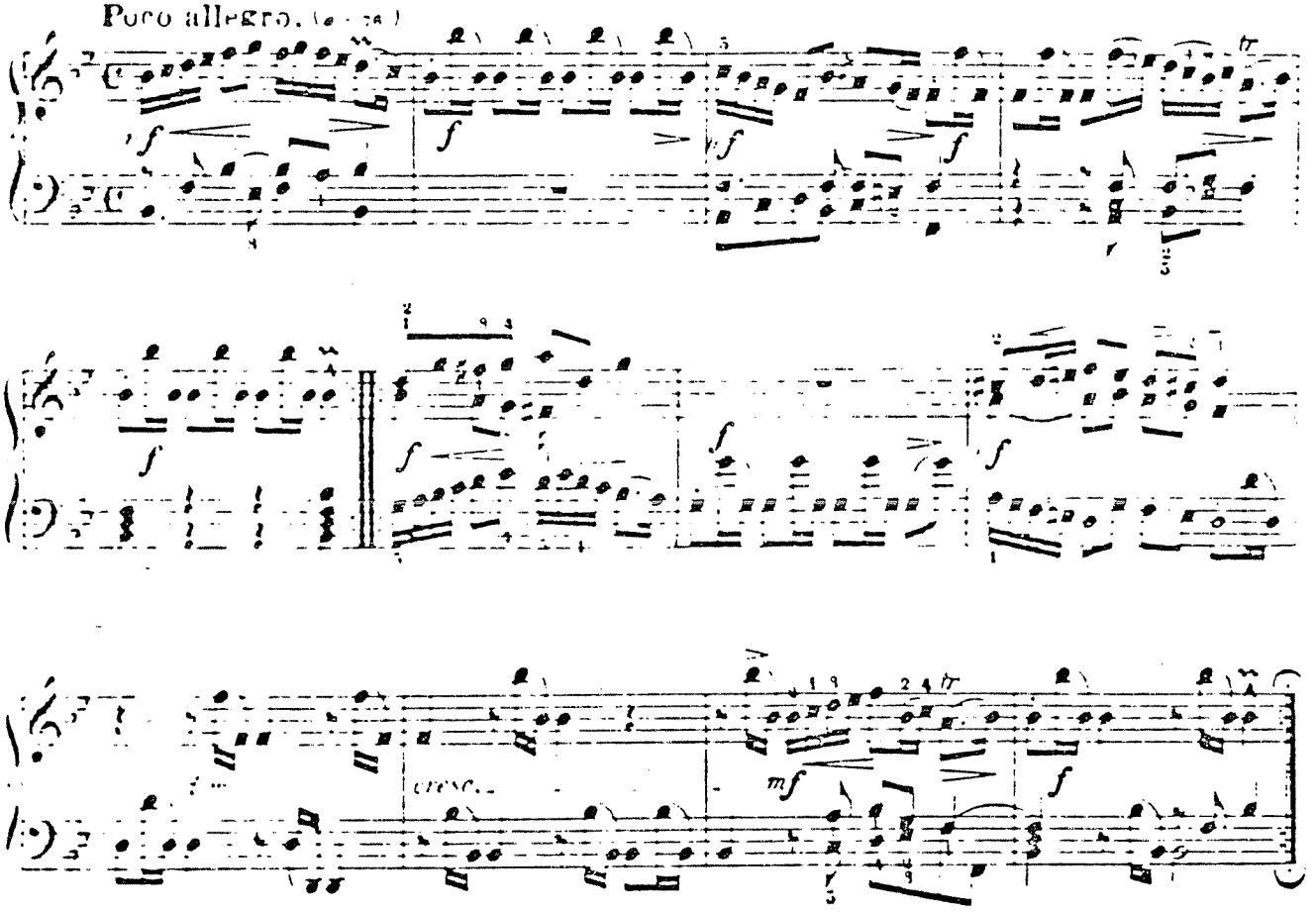
FIGURE 1.

Nº 3. Capriccio

sopra la lontananza del suo fratello dilettissimo.

Ist eine Schmeichelung der Freunde, um denselben von seiner Reise abzuhalten.

Aria di Postiglione.

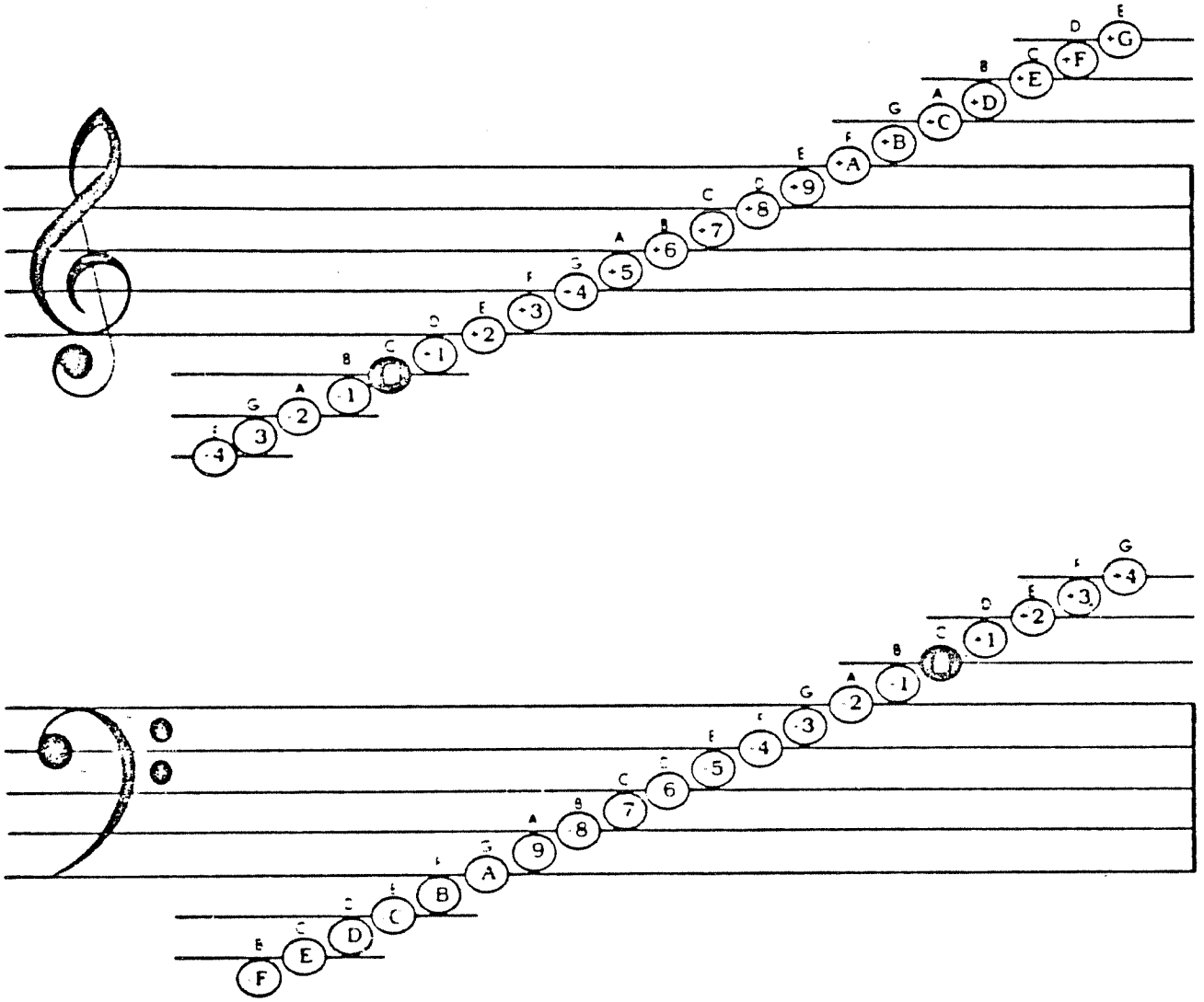


26

FIGURE 2.

```
010   / CAPRICCIO
020   / SOPRA LA LONTANANZA DEL SUO FRATELLO DILETTISSIMO
030   / J.S. BACH
040   P50 / ARIA DI POSTIGLIONE
050   K2&
060   NQ=E0 / POCO ALLEGRO
070   >?
080   V1YC V2YB V3YB
090   M1 *S6789IABSABA9I.8S7
100      V2@IS4Q1I12Q1
110      V3@Q.8I5Q48
120   M2 *(I6:SD6")3
130   M3 *S7654I365S43"I3:SA3"
140      V2@Q.4"I445%Q4
150      V3@I9768Q7B
160   M4 *I3:SA3"I3:SA9&8767I.5S6"
170      V2@QSIS4"43Q4"
180      V3@QSIS785&Q4
190   M5 *(I6:SD6")3
200      V2@Q4SS4
210      V3@Q6SS6
220     R50
230     P52
240   M6 *I8BA#BC8QB
250      V2*Q6I54S3#
260      V3@S3210*I12S1210@I.1S2
270   M7  V3@(I3;S+43")2I3;+4
280   M8 *I7#S89%IA987#Q8
290      V2*Q5I5654Q3
300      V3@S2345%I63456:S+16"
310   M9 *QSISSA3"Q3ISSA3"
320      V2@I6;S+16"Q6ISS6D"QD
330   M10 *Q3ISSD6"Q6
340      V2@ISS18"Q8ISS18"I8;S18"
350   M11 *ISSD6"6789IAS67I.5S6"
360      V2@Q8IS213Q4
370      V3@QSIS7654B
380   M12 *I6:SD6"Q6ISSD6"Q6"
390      V2@Q4ISS18"H8
400      V3@Q6SIS4Q1
410     R52
```

FIGURE 3.



Note: The symbols inside the notes represent the orchestra-80 scale. The small letters above the notes represent the musical scale and are for reference only.

INTERPRETING THE MUSIC LANGUAGE

This section will describe in detail the process of transcribing written music to **ORCHESTRA-80** notation. In order to interpret sheet music correctly we must understand musical notation. Included in this section are several extra charts and examples to help explain a few problems encountered during music system entry.

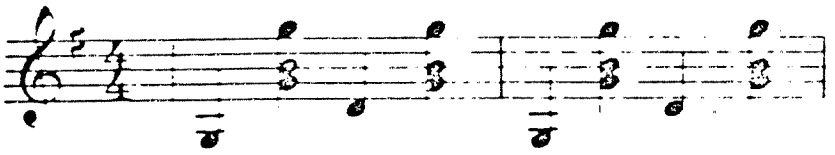
Your **ORCHESTRA-80** program (4-voice) should be loaded and executed according to the instructions previously explained in the manual. You should have a blinking cursor in the upper right hand corner of the screen and a copyright notice at the bottom.

To enter music we type:

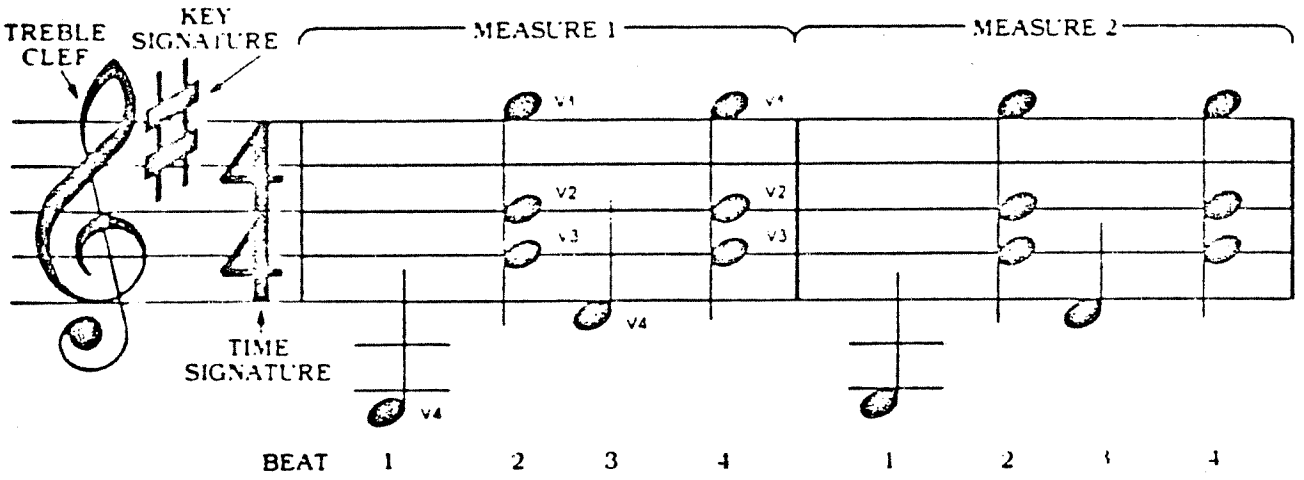
EDIT [Enter]

You now have a blinking white square in the center left section of the screen. The **E** in the upper left corner indicates **EDIT MODE**

MUSIC SAMPLE 1, ACTUAL SIZE



MUSIC SAMPLE 1, ENLARGED



The long horizontal lines are called **Ledger Lines**. Together they are known as the **Staff**. Most of the information we are looking for may be found on the staff or on additional ledger lines added above or below the staff for extra high or extra low notes.

Reading from the left side of the ledger line we first encounter the **Treble Clef** symbol. All of the music in this example will be in the treble clef. The note symbols for the treble clef are found in **Figure 3**. The upper half of the diagram represents the **Treble Clef**, the bottom half is the **Bass Clef**

To the immediate right of the treble clef we find a single **#** mark known as a **Sharp**. This one sharp tells us a very important fact about the piece of music, we now know what **Key** it is

to be played in. One sharp is the key of G. We tell the system to play in the key of G by simply entering.

```
K1# [Enter]
```

If you are wondering where the line numbers are, don't worry, we never use line numbers in this program. The next piece of information could be placed on the same line as the key signature, but for the sake of clarity we will use separate lines for each new item.

The next piece of information on the staff is the **Time Signature**. This consists of two numbers, one on top of the other, like a fraction.

The top number tells us the number of **Beats** in a **Measure**. A **Measure** is a section of the staff between two vertical lines which extend from the top line of the staff to the bottom line. This example has four beats per measure. 1 - 2 - 3 - 4. It's as simple as that. You will get the feel of the music beat by tapping your foot and counting out loud 1 - 2 - 3 - 4.

Next we examine the bottom number in the time signature which is another **4**. This **4** means that each beat in the music will be equal to a **Quarter-Note**. We have established that we have 4 beats in each measure and each one is equal to a quarter-note. Refer to the chart in the **MUSIC LANGUAGE SUMMARY SECTION** and make sure you know what this note looks like. This example contains all quarter-notes.

Okay, we know that there are four beats to a measure and that each quarter-note receives one beat. This is called **4/4 Time** or **Common Time**. We add a line to describe the **Tempo** or how fast the music is to be played.

```
NQ=80 [Enter]
```

We just said that a note (quarter-note) will be equal to a time of **80** (hex). If you start out with this tempo value you can always change it later to speed up or slow-down the piece. The speed can also be determined by holding down the number keys during play as described in the **TEMPO CONVERSION SECTION**.

Well, we explained quite a bit and all you have entered so far is:

```
K1#
NQ=80
```

We can pick up the pace a little now that we've explained music theory.

This piece is ideally suited for four **Voices**. We have added some of our own symbols to help you visualize how we relate notes to voices.

We tell the system what we want each note to sound like. The first time through we will use a different instrument sound for each voice. We have four standard sounds available: **A, B, C, D**.

Try this.

```
V1YA V2YB V3YC V4YD [Enter]
```

We assigned sounds **A, B, C, D** to **Voices 1, 2, 3, 4** respectively.

Our music will begin with a **Part Number** which will contain one or more measures and can be repeated. Measures can not be repeated directly, only parts containing measures.

```
P01 [Enter]
```

Parts may consist of any two-digit numbers but must be unique

Type in the first measure symbol but don't hit **ENTER** because we'll be putting the actual notes on the same line

```
M01
```

All of the notes we are about to enter are in the treble clef. Refer to **Figure 3** for a reference chart of all the notes. The top section of the chart is the treble clef so we will use those numbers. Notice that all of the numbers are positive (+). To avoid putting a + in front of each number, we use an **Asterisk**, *, which tells the system that all numbers entered after this symbol will be positive unless specified otherwise. Add a space and this symbol to the first measure line.

```
M01 *
```

We now determine the notes for the first voice. When we tell the system that the notes belong to a specific voice, we start out with a voice number like **V2**. Again, as a convenience, the first line in a measure is implied **V1** and the **V1** is not required.

As you recall, this music has four beats. You must picture each voice as being present throughout the entire measure even when it doesn't appear. During the other two beats the voice is resting. Most music will show an actual symbol to indicate a **Rest**, but occasionally you will find music like this that does not show all the rests. Don't worry, you learned how to count to four a few paragraphs back and we can make use of your new knowledge to solve this problem.

V1 has notes on beats 2 and 4 only, so beats 1 and 3 are silent and must be entered as rests. If we add

```
QS
```

We have entered two pieces of information using two new symbols. The **Q** means that all notes or rests which follow have the time of a quarter-note. **Q**=quarter-note. The **S** is the symbol for **Any Rest**. In this case it follows a quarter-note time value symbol, so the first beat becomes a quarter-note rest.

Next in **V1** we find our first note, checking the scale in **FIGURE 3 we find the symbol for this note to be a B** so we add this to the line:

```
M01 *QSB
```

After that we find that **V1** does not appear in the third beat of the measure, so we add another rest **S**.

```
M01 *QSBS
```

And finally, the note on beat four is another **B** which completes the line:

```
M01 *QSBSB [Enter]
```

Let's define **V2** the same way. The rests occur in the same places, the only difference being that the notes are **6** instead of **B**. **Measure 1** now looks like this

```
M01 *QSBSB
    V2QS6S6
```

We put the voices on separate lines which requires no additional space when the file is saved. You could also enter the measure as follows:

```
M01 *QSBSB V2QS6S6
```

The result is the same either way. **Voice Three** has the same timing with the notes being **4** instead of **B** or **6**. Enter **V3**

```
M01 *QSBSB
    V2QS6S6
    V3QS4S4
```

We have defined 3 voices. On beats 1 and 3 the voices are silent. On beats 2 and 4 the voices will play simultaneously, each playing a different note. The musical term for this is a **Chord**.

Voice 4 has notes on beats 1 and 3. It has rests on beats 2 and 4. The first note is a **-3**. The note on beat 3 is a **1**. It looks like this:

```
M01 *QSBSB
    V2QS6S6
    V3QS4S4
    V4Q-3S1S
```

The measure is finished. At this point we want to listen to the music so we:

[Break] Press once to leave editor.
SCORE [Enter] Compile Music.
PLAY [Enter] You hear the finished music.

Get back into the editor:

```
EDIT [Enter]
```

We are going to enter the second measure. If you take a close look you will see that it is identical to measure 1. Position the cursor to the bottom of measure 1 and type:

```
R01 [Enter]
```

That tells the system to repeat **Part 01** (which contains measure 1), much easier than typing everything over. The example is now completely transcribed:

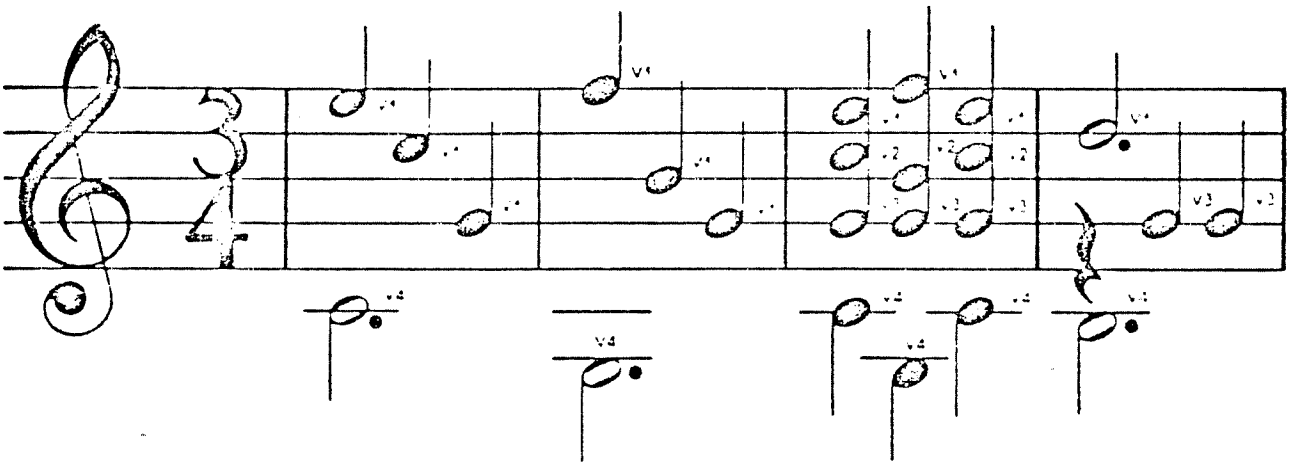
```
K1#
NQ=80
V1YA V2YB V3YC V4YD
P01
M01 *QSBSB
    V2QS6S6
    V3QS4S4
    V4Q-3S1S
R01
```

32

Remember, this example was chosen because of the problem with rests. Most music will have the rests written in and will be much easier. Correct transcription will depend on your awareness of timing, the number of beats per measure, the type of note that gets one beat, and the interpretation of rests.

At this point it would be helpful for you to change the tempo, alter the voices, and add repeats. Score and play the music after each change. This will also give you some practice with the editor.

MUSIC SAMPLE 2



Study the sample music and the finished transcription. Remember, this transcription is only one possible interpretation. The final decision should be based on what you feel sounds best.

Enter **SCORE** and **PLAY** this piece:

```
K0#
NQ=80
V1YA V2YB V3YC V4YD
P01
M01 *Q974
    V4H.0
M02 *QA64
    V4H.-3
M03 *Q9A9
    V2Q767
    V3Q444
    V4Q0-30
M04 *H.8
    V3QS44
    V4H.-1
```

Notice that there are no sharps or flats in the key signature. We enter **K0#**.

The time signature tells us that we have three quarter-note beats in each measure.

Measure 1 requires two voices. **V1** for the melody notes and **V4** for the bass notes. The melody notes are usually the highest notes in the measure. The bass notes are the lowest. **V2** and **V3** are not used and therefore not defined in **Measures 1** and **2**.

Measure 1 contains a dotted half-note, **H.**, followed by the note symbol which is **0**. A half-note lasts twice as long as two quarter-notes. A dotted half-note has its value lengthened

by 1 2. The total value of a dotted half-note is equal to three quarter-note beats. In 3 4 time, three beats fill the measure.

In **Measure 2** we find the same timing, but different note symbols

In **Measure 3** we are back to four-part harmony using all four voices. All the notes are quarter-notes.

In **Measure 4** we use three voices. **V2** is not used. **V1** consists of a single dotted half-note. **V3** has two quarter-notes played after a quarter-note rest (S)

```
V3QS44
```

That's a quarter-note rest followed by two symbol **4** quarter-notes.

V4 is a dotted half-note, symbol **-1**

This example was correct notation-wise in that it was not necessary to calculate any missing rests.

Listen again to this piece, especially **Measure 4**

The two notes in **V3** have no **Articulation** making them sound like one continuous note Change **V3** in **Measure 4** to:

```
V3QS4'4
```

We've created a small space between the two notes to make them more distinct

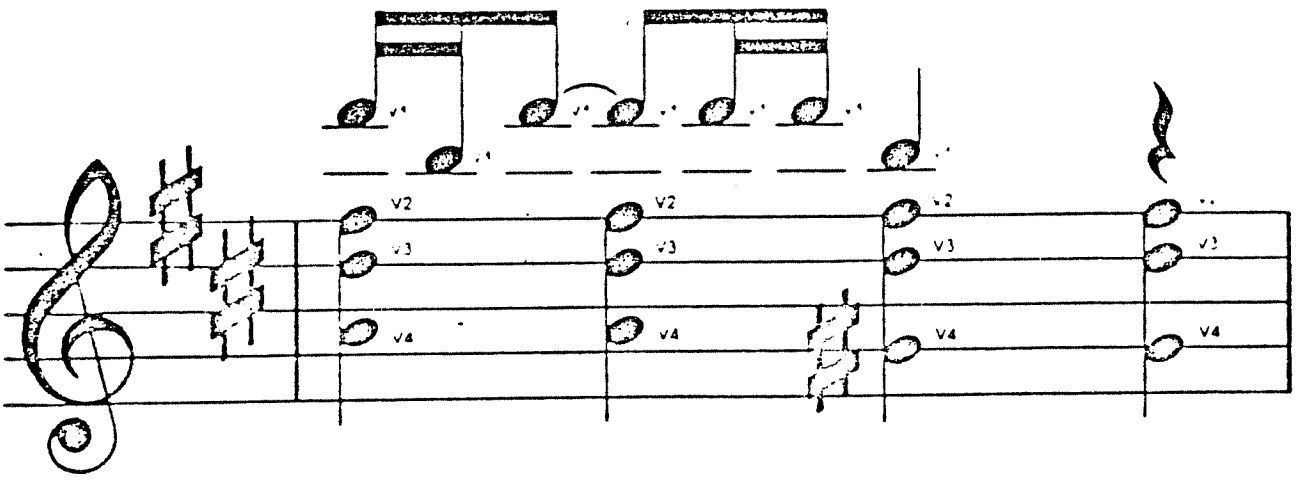
We can also produce an effect called **Staccato** by adding a semi-colon, **;**, behind each note in **V1, Measures 1** and **2**

```
M01 *Q9:7:4:
M02 *QA:6:4:
```

SCORE and **PLAY** this new version The musical notation for staccato is a dot directly above or below the note

Experiment with articulation and staccato to produce different effects in your music

MUSIC SAMPLE 3



This measure, in 4 4 time, looks difficult at first, but is actually very simple. The example is extremely important, however. Enter the file exactly as shown:

```
K2#
NQ=80
V1YA V2YB V3YC V4YD
P01
M01 *SF'D'IFF'SF'F'QDS
    V2QAAAA
    V3Q8888
    V4Q554#4
```

SCORE and **PLAY** the piece. It may not be apparent to you at first, but the melody notes (**V1**) are so high that they are barely audible. Let's make a change that will bring the melody down to a comfortable frequency.

We can do this in two ways. If we want to lower only **V1**, we can add a voice modifier after the part number.

```
P01 V1U-7
```

V1U-7 instructs the system to play the notes in **V1** one octave lower than written. Make this change. **SCORE** and **PLAY**. The melody is more audible, however, the other voices are still too high. The solution is to lower all the voices.

Remove **V1U-7** from the line and replace with **<9**

```
K2#
NQ=80
V1YA V2YB V3YC V4YD
P01<9
M01 *SF'D'IFF'SF'F'QDS
    V2QAAAA
    V3Q8888
    V4Q554#4
```

SCORE and **PLAY**, the entire piece has been lowered over an octave. Try replacing the **9** with an **A**, **C** or any hex digit up to **F**. The tone quality of the voices improves and distortion of the high notes is reduced.

V1 starts out with two sixteenth-notes, **F** and **D**, each followed by a **'**, (articulation to make these notes more distinct).

The third and fourth notes are eighth-note **F**. The curved line between the two notes indicates that the notes are **Tied**, meaning no articulation on the first **F**. The two notes must sound without interuption. The second **F** has articulation to separate it from the two sixteenth-notes that follow.

The two sixteenth-notes are **F** and have articulation because they are not tied.

A quarter-note **D** follows without articulation.

The last beat in **V1** is a rest (**S**).

V2 and **V3** consist of four quarter-notes each. **V2** has 4 **A** notes. **V3** has 4 **8** notes.

V4 also has four quarter-notes. The first two notes are **5**, the third note has an accidental applied to it. In musical terms, the **#**, a **sharp**, preceding the note means that the note and all following notes of the same symbol are to be played 1/2 step higher than normal. Our system carries the accidental automatically so the first note is entered as **4#** and the following note is entered just like it is written, **4**. The second **4** will be played as a **4#** no matter which voice plays it.

After playing the original transcription, change **V2**, **V3**, and **V4**:

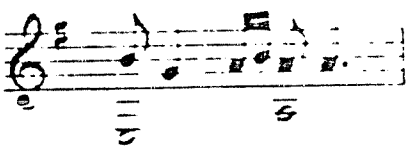
```
V2QA:A:A:A:
V3Q8:8:8:8:
V4Q5:5:4#:4:
```

A ; has been added to every note. **SCORE** and **PLAY** this new version and you will hear a completely different musical effect. The chords have a shorter duration which tends to accent the four beats in the measure.

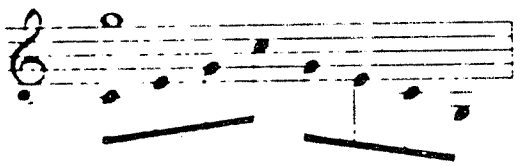
Articulation, used in this piece and the following samples, is included to demonstrate a special musical effect. Articulation is optional; in fact, most notes don't need articulation unless they are followed by a note of the same symbol.

MORE SAMPLE MUSIC

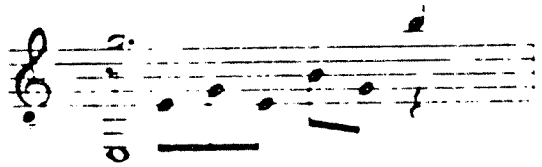
Note: Remember to check the Key signatures; all samples are in 4/4 time.



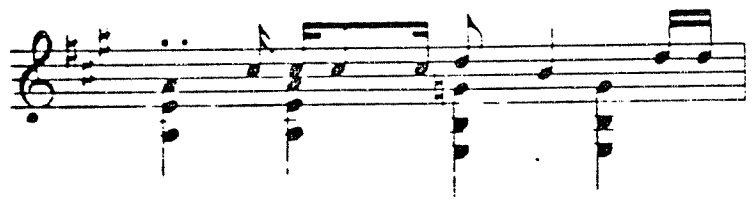
```
*I6Q4S5'6'I5'Q.5
@V2H52
```



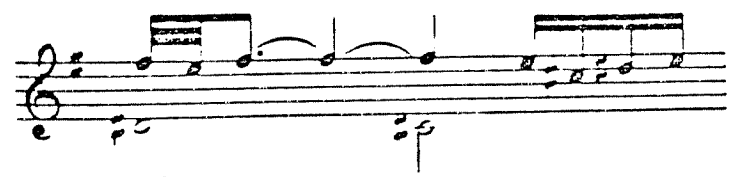
```
*WB
V2I0247420-3
```



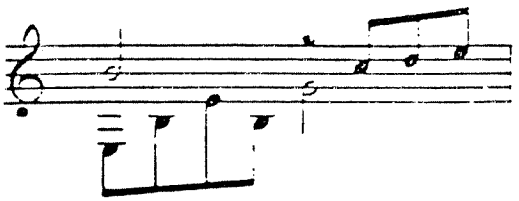
```
*H.BQD
V2IS24264QS
V3W-5
```



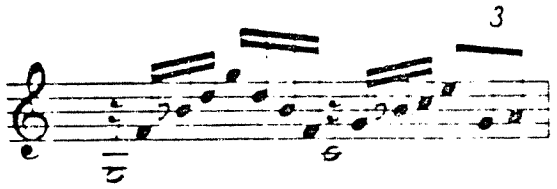
```
*I.SS7'7'I7'S7'I8Q6S8'8'
V2Q5'54%4
V3Q2'2-1'-1
@V4Q2'25'5
```



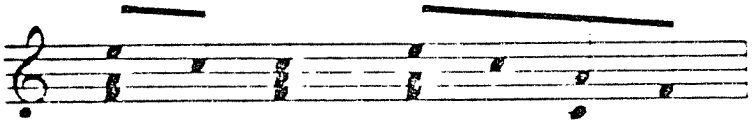
```
*TA'9'I.AQAQAS9'7#'8#'9'
V2H1#0#
```



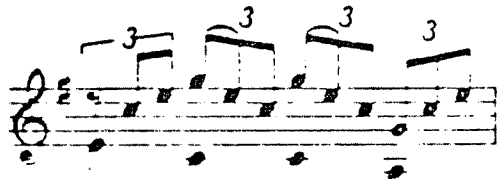
*H6IS789
V2I-5-12-1H4



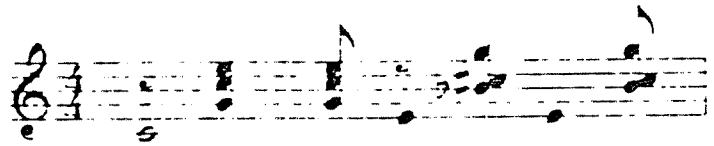
*SS3'6&'8'B'8'6'3'S4'6&'7'I:945
V2H-30



*I97'Q7I9753
V2Q5'5'5'S
V3Q3'3'30

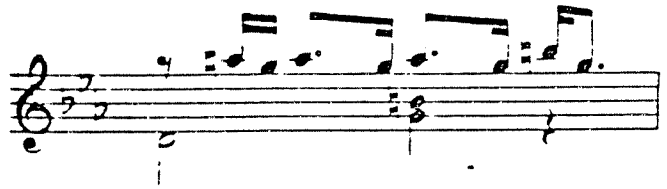


*I:S7'9'B9'7'B9'7'4'7'9'
V2Q20'0-2



*ISQ9'I9SQB'IB
V2ISQ7'I7SQ7#'I7
V3ISQ4'I4SQ6&'I6
V4H0Q2'2

*QS765
V2H0Q23



*ISSC%'B'I.CSB'I.CSB'D%'I.B
V2HSQ6%S
V3H1S

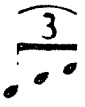
DOTTED NOTES A dot after a note adds one-half the value of the preceding note

Examples in 4-4 time

Dotted Quarter Note = 1+1+2 beats

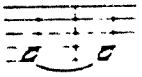
Dotted Half Note = 3 beats

TRIPLET NOTES Three notes of equal time value played in the time of two of the same kinds of notes



HOLD Notes sound longer than their actual values

TIED NOTES The first of two tied notes must have no articulation. The two notes sound as one continuous note



When the last note in a measure is tied to the first note in the next measure an accidental on the first note is passed to the note in the next measure

STACCATO Expression modifier used to make a note sound separate or detached from the other notes. Use a comma or a semicolon after the note symbol to produce this effect

ARTICULATION Expression modifier used to make a note sound clear and distinct especially within a series of similar notes. Use a hyphen or quotation mark after the note symbol

Repeat Sign

First and second endings

𝄎 Repeat preceding measure

𝄴 Common time $\frac{4}{4}$

𝄵 Cut time $\frac{2}{2}$

rit. Gradually slower

Fine Finish

D.C. al Fine Repeat from the beginning to the end of the measure marked *Fine*

D.S. al Fine Repeat from the sign 𝄋 to the end of the measure marked *Fine*

D.C. al 𝄌 *Coda* Repeat from the beginning to the *Coda* sign 𝄌 and then skip to the *Coda*

D.S. al 𝄌 *Coda* Repeat from the sign 𝄋 to the *Coda* sign 𝄌 and then skip to the *Coda*

𝄋 Repeat from this sign

𝄌 *Coda* mark when playing second time after *D.C.* skip from this sign to the *Coda*

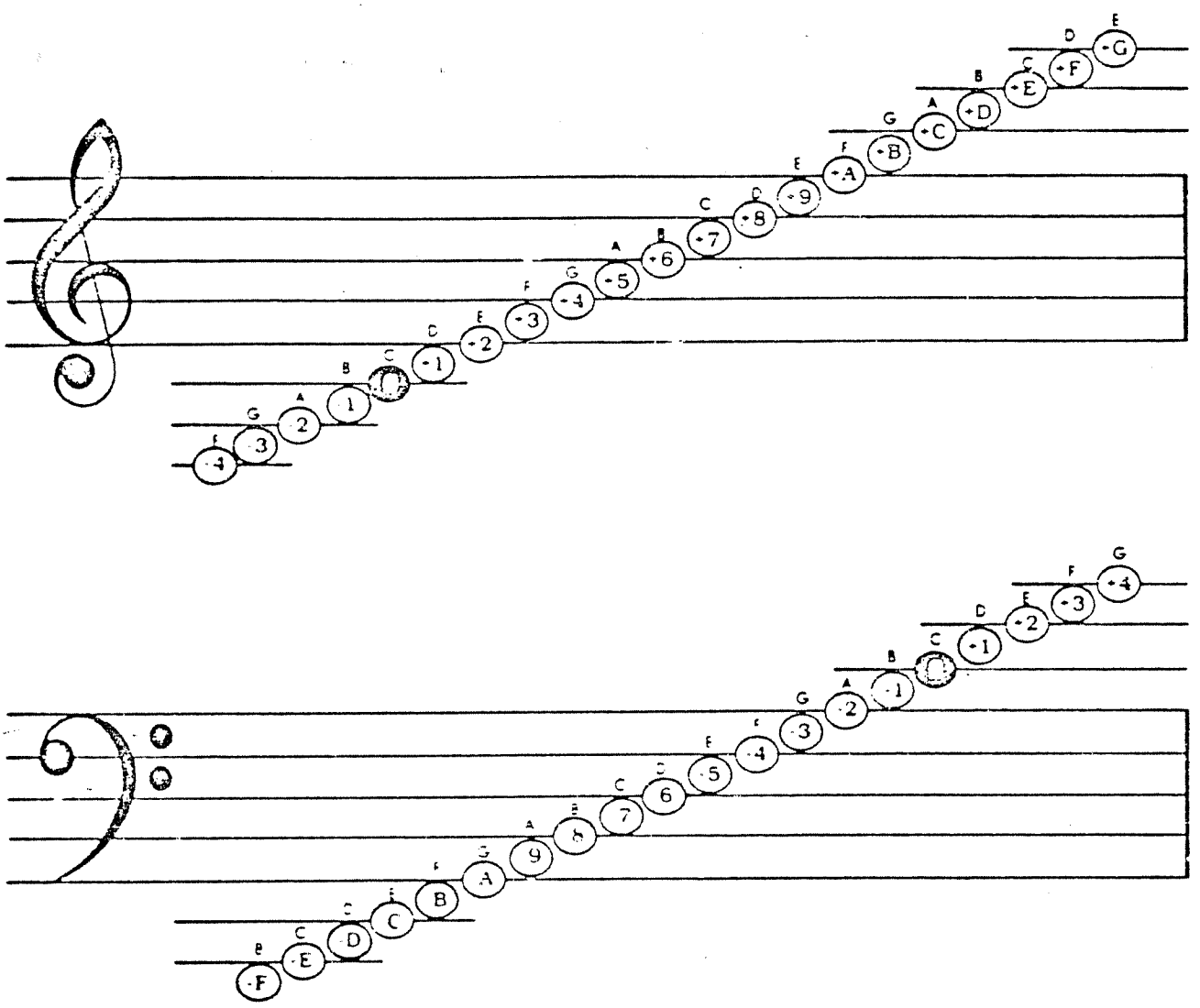
NOTE MODIFIER	MUSICAL EXAMPLE	NAME
#	♯	ACCIDENTAL SHARP
&	♭	ACCIDENTAL FLAT
%	♮	ACCIDENTAL NATURAL
##	𝄪	DOUBLE SHARP
&&	𝄫	DOUBLE FLAT
%#	♮♯	NATURAL SHARP
%&	♮♭	NATURAL FLAT

EXPRESSION MODIFIER	EXAMPLE	NAME
, (comma)		SHORT STACCATO
; (semicolon)		LONG STACCATO
' (apostrophe)	none	SHORT ARTICULATION
" (quotation mark)	none	LONG ARTICULATION

TIME VALUE SYMBOL	MUSICAL EQUIVALENT	NAME OF NOTE	REST ($)
W	𝅝	WHOLE note	𝄻
H	𝅗𝅥	HALF note	𝄼
Q	♩	QUARTER note	𝄽
I	♪	EIGHT note	𝄾
S	𝅘𝅥𝅯	SIXTEENTH note	𝄿
T	𝅘𝅥𝅰	THIRTY SECOND note	𝅀
X	𝅘𝅥𝅱	SIXTY FOURTH note	𝅁

TIME VALUE MODIFIER	MUSICAL EQUIVALENT	NAME
. (period)	♩.	DOTTED note
: (colon)	3	TRIPLET

40



Note. The symbols inside the notes represent the orchestra-80 scale. The small letters above the notes represent the musical scale and are for reference only.

COPYRIGHT

TRADEMARK

The names "Orchestra-80" and "Software Affair" are trademarks of Software Affair, Ltd.

DISCLAIMER

Software Affair, Ltd., assumes no responsibility with respect to any liability, loss or damage caused by programs or computer equipment sold by Software Affair, Ltd. Software Affair, Ltd., makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

COLOPHON

Designed by Thomas A. Blakeley, Santa Clara, California.
Photorypeset by Blakeley Graphics, Santa Clara, California. The types are ITC Bookman and Avant Garde.