Symphony 90

Composer

MUSIC COMPOSER SYSTEM

Model III and Model 4/4P/4D

by David Goben

Published and Distributed by

Computer News 80

PO Box 680 Casper, WY 82602

SYMPHONY-90
MUSIC COMPOSER SYSTEM

Published and Distributed by
Computer News 80
PO Box 680
Casper, WY 82602

Printed and produced in the United States of America.

10 9 8 7 6 5 4 3 2 1

From time to time the author may choose to enhance or improve this product, either through documentation or through the program. The author reserves the right to apply these enhancements and/or improvements without notice.

TABLE OF CONTENTS

INTRODUCTION

INTRODUCTION TO THE COMPOSER

Welcome to the exciting world of synthesized music composition with the ***SYMPHONY-90 MUSIC COMPOSER SYSTEM.*** The ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** is the only truly serious music composition package yet produced for both the Model III and Model 4 computers. With this software package installed in your Model III or Model 4/4p/4d, and the ***SYMPHONY-90*** hardware interface (or Orchestra-90 interface) connected between the computer's 50-pin external I/O card edge connector and your stereo, you can load, edit, save, play, and compose music selections with up to 5 instruments (including percussion if you wish) in captivating stereo sound. Also, other ***SYMPHONY-90*** users can enjoy your compositions by playing them using their own ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** or their ***SYMPHONY-90 MUSIC PLAYER SYSTEM*** (available from *Computer News 80*).

Due to a slower clock speed, Model III users can enjoy only up to 4 voices, unless they are operating on a Model 4 in the III-mode, or have a 4MHz speedup board installed, whereas they can also enjoy 5 voices just like the Model 4 users. If you do not have the music hardware interface, then you cannot play music from this package until it is installed. Contact *Computer News 80* for ***SYMPHONY-90*** hardware information.

DISTRIBUTION DISK FORMATS

The ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** came to you configured for the Model III or the Model 4. Both packages support the same features and commands, so moving from one to the other is like stepping back into familiar territory

IMPORTANT NOTE: Both versions are supplied on DATA diskettes. What this means is that they *do not* contain a *Disk Operating System* (DOS). Therefore you cannot expect to simply insert the disk into Drive :0, boot it up, and run your program. The reason for this is that a DOS is a copyrighted program, and if it were included you would also be required to pay the additional cost for the DOS so that the distributor could in turn pay the owner of the DOS their due royalties. This of course greatly increases the cost of the disk. Since most people already have a DOS, this can be considered an unnecessary expense.

The Model 4 version is supplied to you on a single sided double density 40 track TRSDOS 6-compatible data disk; the standard format for the Model 4. The Model III version is supplied on a single sided double density 40 track LDOS format data disk. If you are using the TRSDOS 1.3 Disk Operating System, you can boot up the ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** disk in Drive :0 by inserting it and pressing the **RESET** switch, and allow the distribution disk to copy its files to a *single sided* double density disk previously formatted by your DOS, by following the simple instructions provided on the screen by the copier system on the disk when you boot it up. Use that copy as your work master.

Before doing anything else, you should make at least 1, but preferably 2 backup copies of the master disk (or the data disk created per instructions above for TRSDOS 1.3). This is for your own safety. This way you will always have a perfectly good copy of the master disk in case something goes wrong with your work copy. No matter how much you may think you know, NEVER use the original disk as a work disk! Mistakes are possible, even for the experts. Remember Murphy's Law.

DISK CONTENTS

PROGRAM FILENAME	*PURPOSE*
SYMPHONY/CMD	SYMPHONY-90 Composer program.
ASC2SYM/CMD	Utility file to convert ASCII files using the Music Programming Language to SYMPHONY-90 format.
SYM2ASC/CMD	Convert Symphony-90 files to ASCII format

SAMPLE MUSIC FILENAME	*WHAT IT REPRESENTS*
AMADEUS/SYM	Rock Me, Amadeus
ANILUVER/SYM	And I Love Her
ANTHEM/SYM	Star Spangled Banner
BRIAN/SYM	Brian's Song
BRNDBR/SYM	Brandeburg Concerto # 3
CLSORC/SYM	Classical Orch
COUNTRY/SYM	Take Me Home, Country Road
CTWNRACE/SYM	Camptown Race
DAME/SYM	There is Nothing Like a Dame
DNCDRK/SYM	Dancing in the Dark
ENTRY/SYM	Entry of the Gladiators
FLSHDC/SYM	Flashdance (What a Feeling)
GHOST/SYM	Ghostbusters
GJWTHF/SYM	Girls Just Want to Have Fun
GLORIA/SYM	Gloria
GRACE/SYM	Amazing Grace
HELLO/SYM	Hello
HERO/SYM	Holding Out for a Hero
HOTM/SYM	Heaven On Their Minds
LUFT99/SYM	99 Luftballons

NOTE: If there have been any updates, corrections, or any documentation which had been later added, but not included in the manual, then there will be additional files called README/CMD and README/TXT. You can view the updates by entering **README** from the DOS ready line. If these files are not on your disk, then there have not been any updates added.

CHAPTER 1

STARTING UP THE SYMPHONY-90 COMPOSER

This section explains system power-up, software configuration, and program initialization.

The software provided on the disk accompanying this manual are designed to work with either the Model III (mode) or Model 4. It will not work on both. If you desire such an arrangement, then you must purchase another package with the software configured for the other system.

MODEL 4 STARTUP

The Model 4 version requires no special configuration, and you can immediately begin using the SYMPHONY/CMD program. Therefore, with a copy of SYMPHONY/CMD on a mounted disk, you can simply enter the command **SYMPHONY** from the DOS ready line. When you do this, the program will load, and once loaded, it will display its main screen, with a temporary copyright notice in the lower screen area. Go on to the section titled ***USING THE COMPOSER PROGRAM.***

MODEL III STARTUP

The Model III version requires a configuration phase before you can use the composer program (SYMPHONY/CMD). This configuration consists of selecting a clock speed and the number of voices you wish for ***SYMPHONY-90*** to support on your computer. If you are operating on a Model 4 in the Model III mode, you will not be prompted for a clock speed as the program will automatically take advantage of the 4MHz clock speed capability in your computer. Remember, the faster clock speed allows you to take full advantage of up to 5 voices (instruments).

To configure the Model III version, you must have your DOS booted up and a copy of SYMPHONY/CMD on a mounted disk. From the DOS Ready prompt, enter **SYMPHONY.** The program will load, displaying copyright information, and then a set of pre-programmed prompts will be displayed.

If you are running the program on a Model 4 operating in the III-mode, then the program will automatically activate and deactivate the 4MHz clock as required for proper operation. If this is your case, you can skip the following section on clock speed prompts, as they will not be presented to you. You can go on to the section titled ***NUMBER OF VOICES.***

CLOCK SPEED SELECTION

If you are running on a stock Model III, the first question you will be asked is:

```
Use your fast clock (Y/N)?
```

If your computer is modified for 4MHz operation, then respond by pressing **Y** and pressing the **ENTER** key. Otherwise respond with **N.**

If you selected Y, you will be next prompted with:

```
Select clock by software control (Y/N)?
```

There are two ways of activating the clock speed modifications, depending upon the version installed in your system -- through hardware or software. If the speed-up is activated by hardware, this is usually done via a manual switch, which can be set into one of its pole positions to activate speed-up. If you have a speed-up kit installed which is activated in this manner, then answer the prompt with "N".

If your speed-up kit is activated by software control, then answer the prompt with "Y". When you select "Y", you will next be prompted with:

```
Enter the ENABLE code:
```

To make efficient use of a software controlled speed-up device, the program needs to know the codes used to select the 4MHz clock speed. Enter the required instructions in the form of up to 8 hexadecimal machine code instruction bytes. If you have the DOS configured to use the fast clock during normal operations, and in fact have the system already running at the fast speed before going into the composer program, then you can respond to this prompt with ***00*** (a machine code NOP -- No OPeration). Otherwise, you must tell the program how this is done.

For example, suppose that your clock speed-up required the instructions:

OUT(254),1 (from BASIC)
or
LD A,1
OUT (254),A (from machine language)
or
3E01D3FE (from machine code -- in hexadecimal).

The last example above performs the same functions as the BASIC and machine language examples. ***SYMPHONY-90*** requires that you use the coding format of the last example. Therefore you should answer the prompt with 3E01D3FE. If your hardware expects some other coding, then enter the hexadecimal machine code for it in the format given in the last example.

You will next be prompted with:

```
Enter the DISABLE code:
```

It is in most cases important that the fast clock be disabled when the system reads from or writes to a disk. If you tell the program how, the composer will switch to standard clock speed before doing any disk input/output. If you have your DOS configured to use the fast clock during normal operations, then you can respond to this prompt with ***00*** (NOP). Otherwise you must give the program the codes required to slow the clock back down to its normal speed. Examples of such instructions are:

OUT(254),0 (from BASIC)
or
LD A,0
OUT (254),A (from machine language)
or
3E00D3FE (from machine code -- in hexadecimal).

The last example performs the same instructions as the other two, but is in the format required by the composer program. If your slow-down command requires you to send a value of zero to port 254, then answer the prompt with 3E00DEFE. If your hardware expects some other coding, then enter the hexadecimal machine code for it in the format given in the last example.

NUMBER OF VOICES

The next prompt you will be presented with is:

```
Use how many voices (4/5):
```

If your system is capable of 4MHz operation, then answer the prompt with the number of voices you wish to use. You are not required to select 5 voices. If you do not have 4MHz capability, then please select 4. Though 5 voices will work; it will cause an effect called "aliasing", which causes the sound quality on slower-clock systems to slow down during playing, and causing unacceptable distortion, causing the sound quality to be very poor.

SAVING THE CONFIGURATION

The next prompt asks you:

```
Save this configuration (Y/N)?
```

If you do not wish to save these selections permanently, answer with "N". If you answer with "Y", then you can save the selections permanently in a file which will automatically make these selections when you start the program. In this case you will be given the following prompt:

```
Enter the program name to use:
```

Type in a program name (less extension) to save a pre-configured copy of the composer program to. This can be as simple as **S**, so that you only have to type the letter "S" to activate the program. If you use SYMPHONY, remember that your current copy of SYMPHONY/CMD will be over-written. If you try this with your master disk, then you should afterward voluntarily admit yourself to a mental hospital for psychiatric evaluation.

USING THE COMPOSER PROGRAM

The ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** screen is divided into 2 parts: the Command Line and the Programming Buffer. The Command Line is situated between the 2 horizontal bars at the top of the screen. This is where you give the program specific instructions for performing special tasks. You can load, save, and play music files from here. The Programming Buffer is located in the large space below the lower bar. This is where you compose and edit music files. All commands will be covered in detail in this manual.

A SAMPLE CONCERT

To play music, the ***SYMPHONY-90*** hardware interface (or Orchestra-90 interface) must be connected to the computer's 50-pin card edge connector, and standard RCA cables must connect the interface to your stereo system. Your stereo should be on, and its peripheral selector should be set to the device used to connect the interface. Always be sure that everything is connected correctly. Most "bug" reports end up being an connection error on the part of the user. The stereo should also be set at a low volume.

If the cursor is not blinking on the Command Line, then press the BREAK key to put it there. Next type:

GET AMADEUS

and press the ENTER key. The file AMADEUS/SYM (included on the distribution disk) will load. The composer will then compile the file into its internal notation format, and begin playing the music through your stereo. If you do not hear any music after several seconds, you should check your connections until music is produced. You can also use GET to load other /SYM files on the distribution disk. There are 20 sample files in all, giving you a good start on your own personal collection. Hundreds of other files can be obtained from the SYMPHONY-90 LIBRARY, available from *Computer News 80.*

CREATING YOUR FIRST MUSIC FILE

This portion of the manual will help you to become familiarized with the ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** by showing you what it can do. The first step is to create your own music file. The next step is to save it onto disk for later use.

With the program loaded and the cursor blinking on the Command Line, type NEW and press the ENTER key in case there is anything already in the Programming Buffer. Next type EDIT and press the ENTER key. This will cause the cursor to move to the Programming Buffer. Notice that when the blinking cursor is in the Programming Buffer that it is displayed as a graphic block.

As is almost a tradition with music composition programs, your first simple music file will be the familiar "Twinkle Twinkle, Little Star". This version uses the simplest features of the Music Programming Language used by the ***SYMPHONY-90 MUSIC COMPOSER SYSTEM.*** It will contain no embellishments, and will consist of only a single voice. Chords, percussion, and multi-voice pieces should be saved for later when you become more familiar with the program.

With the cursor blinking in the Programming Buffer, type the following, terminating each line by pressing the ENTER key (all the "odd" syntax will be explained in the next section of this chapter):

```
NQ=60
M01 Q3,3,7,7,
M02 Q8,8,7,$
M03 Q6&,6&;5,5,
M04 Q4,4,3,$
M05 Q7,7,6&,6&,
M06 Q5,5,4,$
M07 Q7,7,6&,6&,
M08 Q5,5,4,$
M09 Q3,3,7,7,
M10 Q8,8,7,$
M11 6&,6&,5,5,
M12 4,4,3,$
```

If you make an entry error, use the arrow keys to point to your mistake, and overtype the error with the correct information.

COMPOSING THE MUSIC FILE

The next step is to check your music file for syntax errors, and at the same time compile the file into a format usable by the music playing portion of the composer. You must issue this command from the Command Line. So press the BREAK key to return the cursor to the Command Line, and then enter:

```
COMPOSE
```

to compose the file. When you issue this command, notice that the cursor disappears. Once the music file is composed, the cursor will reappear, ready for the next command. If an error message is displayed on the command line, notice that the line in question is displayed at the top of the Programming Buffer. Press any key to turn the cursor back on and return to the Command Line level. You must edit the line in question and compose it again before you can begin playing it.

PLAYING YOUR MUSIC FILE

The next step is to play the file. You do this by issuing the command:

PLAY

You will then hear your music played through the stereo. Once it is completed, control will return to the Command Line, where you can then type in a new instruction.

SAVING YOUR MUSIC FILE

You can save your new file to disk using the SAVE command. Let's call our file TTLS. With the cursor in the Edit Line, enter the command:

SAVE TTLS

A file called TTLS/SYM will then be created on disk. You can include a drive extension, such as TTLS:1, to save the file onto a particular disk drive.

MULTI-VOICED EXAMPLE

To begin this session, be sure that the Programming Buffer is empty by issuing NEW from the Command Line. Next enter EDIT to go into the now blank Programming Buffer.

Converting traditional music notation to a format recognizable by the composer program is called transcribing. Due to the diversity of traditional notation symbols used on a standard music score, translating these symbols to keyboard entry format may seem quite an undertaking at first glance. But once you become familiar with the keyboard symbols used, which are in most cases keyed off of the names or appearances of the symbols, you will find them easy to understand and remember.

To properly transcribe sheet music it is a great help to understand its music notation. To assist you in this, the appendix sections of this manual contain numerous tables and charts. Several examples are also included, to graphically demonstrate how you might empalement those features in your transcription to the music language substitutes which the composer program uses.

For example, take the following sheet music line:

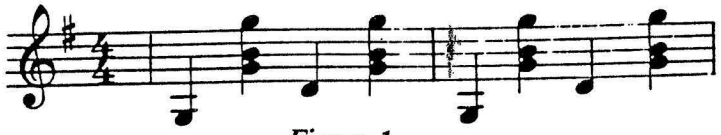


Figure 1.

This line of music contains various symbols which will allow you to transcribe it into a format easily understood by the composer program. Below is a breakdown of those symbols

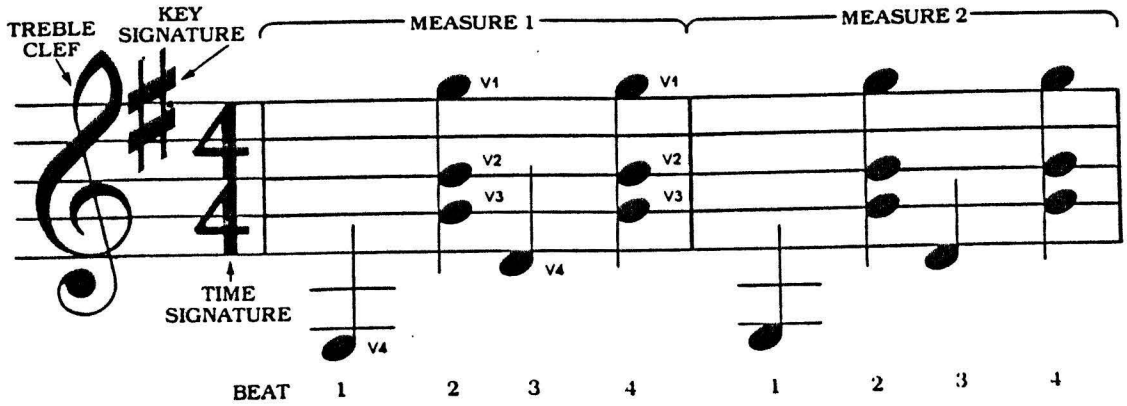


Figure 2.

Each of the 5 horizontal lines are known as a *Ledger Line.* Combined they are called the *Staff.* Most of the information you will be using to transcribe musical data will be found on these 5 lines, or on additional lines found above or below the standard 5-line Staff, which allows for higher or lower notes.

Music is read from left to right, and so let us begin examining Figure 2 in this fashion. On the leftmost side we see a swirling figure called the *Treble Clef.* This is also called the "G" Clef, because, as you can see in the figure, it is centered on the 2nd line of the staff, indicating the G note above middle C. This symbol tells us that all notes on this staff will be in the treble clef. You can examine all the notes available for the treble clef and bass clef by examining Tables 1 and 2 in Chapter 2.

Just to the right of the treble clef is a solitary crosshatch symbol "#" which is called a *Sharp* in music notation. This single item gives us a very important clue about this piece of music -- what key it is to be played in; its *Key Signature.* For those familiar with music notation, a single sharp indicates the key of G.

To tell the composer to play in the key of G, type:

```
K1#
```

This basically tells the composer that the "KEY" has 1 sharp.

The 4/4 on the staff following the key signature is called the *Time Signature.* In sheet music, this is always represented as one value over the top of another, like a fraction. The top value represents the number of beats to a measure. Each measure is marked off by two vertical lines in the Staff, extending from the top Ledger Line to the bottom one, called Bar Lines, on either side of the measure (notice the Bar Line to the immediate right of our 4/4 Key Signature). Our example shows us that the music uses 4 beats to a measure. When you usually tap your foot or clap your hands in time with music, you are normally counting off the beats.

Below that value is another value of 4. This bottom value indicates that each beat will indicate a Quarter Note. From this we now know that there will be 4 beats in each measure, and each beat will be equal to 1 Quarter Note. All the notes in our example (the black ovals with vertical lines extending from them) are Quarter Notes (see Appendix F for examples of other notes).

In our Music Programming Language, an instruction line must be added to describe the music's Tempo; how fast the music should be played. This is a bit more complex than previous explanations of sheet music notation. This command indicates both the Time Signature and our Tempo. Enter:

```
NQ=8Ø
```

This line actually consists of 2 separate instructions, and could have in fact be placed on separate lines if you had wished to. The "NQ" portion denotes our Time Signature, telling us that our Note type (N) is set for 1 Quarter Note (Q) for each beat. The "=8Ø" sets the tempo. 8Ø is actually a hexadecimal value. This value must always be represented by 2 hex digits. You may like to start out all your new compositions with this tempo value. Later you can speed up or slow down the tempo of your music by editing this value (see Appendix B for details on setting various tempo values).

So far, our music file consists of the lines:

```
K1#
NQ=8Ø
```

As previously stated, you can have up to 5 voices represented in your music. Our current example uses only 4. SYMPHONY-9Ø can be programmed to make each voice an individual sound. There are 5 standard sounds available annotated as A, B, C, D, and E. These 5 sounds, as set up when you start SYMPHONY-9Ø, are as follows:

```
A = Trumpet
B = Oboe
C = Clarinet
D = Organ
E = Violin
```

If you are interested in defining sounds other than the defaults, refer to Appendix D for examples for other instruments, or Appendix E for sample percussion instruments.

To assign sounds A, B, C, and D to Voices 1, 2, 3, and 4 respectively, you would type:

```
V1YA V2YB V3YC V4YD
```

This command line tells the composer that Voice 1 is to be assigned (Y) Register A -- to make Voice 1 sound like a trumpet. Voice 2 is assigned (Y) Register B -- to make Voice 2 sound like an Oboe. Voice 3 is assigned (Y) Register C -- a Clarinet. Finally, Voice 4 is assigned Register D -- an Organ.

The music can then begin with a Part Number. Marking at least 1 Measure with a Part Number allows you the capability to repeat that measure more than once in your music, either consecutively, or in different portions of the music. Part Numbers are essential since Measures cannot be repeated individually. Part numbers can be represented by 2 individual hexadecimal digits. The numbers you assign to parts should be unique from names of other parts you may assign in the music score. This is especially important if you are marking them for repeated playing. If more than one part has the same number, then any reference to that number will cause only the first encountered part with that number to be used. This can cause you intense hair pulling until you find the error in transcription.

To mark the current part we are creating as part Ø1 (there must always be 2 digits), you would enter:

```
PØ1
```

You next begin entering your measures. A measure is marked by beginning it with the letter "M", and then followed by a 2-digit Measure Number. Measure numbers are not actually used by the composer, but are allowed for your own personal reference. By using a pencil and assigning each measure in your sheet music a number, you can easily relate positions on the sheet music to positions in the Programming Buffer. Therefore, next type (without pressing the ENTER key):

```
MØ1
```

The reason we did not press the ENTER key is because we must follow this marker with a Measure symbol. So press the SPACEBAR, and then type the asterisk character "*". The asterisk tells the composer that all the notes following are in the treble Clef.

We must next tell the composer which voice we are going to enter note data for. Therefore type the letter "V" and the digit 1 after the asterisk. Our line now looks like this:

```
MØ1  *V1
```

Please note that BY DEFAULT, the first line on a measure is assumed to be on the Treble Clef, and to be assigned to Voice 1. Therefore we could have gotten away with not typing "*V1" on our line, but by adding it, the demarcation of voices and clefs for the measure becomes visually clearer.

By referring back to Figure 2, we notice that Voice 1 (V1) has notes only on beats 2 and 4. Beats 1 and 3 on V1 are silent. Since the composer cannot assume such information, you must enter the silent notes as Rests. The Music Programming Language's code for a Rest is $. We therefore type Q$ after "*V1" to add 2 new pieces of information. The "Q" indicates that all notes or rests which follow on the line (unless altered by a subsequent note type insertion) will be Quarter Notes. The $ is the symbol for a rest. This tells us that the first Quarter Note for V1 is silent. In the case of this sample music line, the Rest follows a Quarter Note time value symbol (Q), and so this first beat becomes a quarter note rest.

Now the actual notes for Voice 1 can be entered. By referring to Tables 1 and 2 in Chapter 2, you can see that the first "vocal" note for Voice 1 is an F note, which is represented by the letter B on the scale. Therefore we would type the letter B to represent beat 2 for Voice 1. Our line now looks like this:

```
MØ1  *V1Q$B
```

Voice 1 does not have a note for the third beat, and so we add another rest to the line:

```
M01  *V1Q$B$
```

Finally, the note translation for beat 4 is also a B, so add that to the line, making:

```
M01  *V1Q$B$B
```

You can now press the ENTER key to move on to the next line.

Voice 2 (V2) can be defined like Voice 1. Please note that we should NOT try to define Measure information for V2 since we are still in the current measure. At the same time, we could also make reading our information much clearer by indenting the data for subsequent voices on the line. We can do this by pressing the SPACEBAR a number of times to indent the next line, so that V2 will line up under V1 on the first measure line. Notice that the rests reside in the same positions between V1 and V2, but Voice 2 harmonizes on an "A" note, indicated by the table position 6. Thus, we can type:

```
*V2Q$6$6
```

and press the ENTER key to add the data for Voice 2. To clarify everything so far, and to assure yourself that you have not made a mistake, our measure currently consists of the 2 lines:

```
M01  *V1Q$B$B
     *V2Q$6$6
```

Voice 3 has the same timing as V1 and V2, but it harmonizes on a note translated to the number 4. To define V3, you would enter:

```
*V3Q$4$4
```

We have now defined 3 voices. When these 3 voices are played, they will play in unison, making a chord.

Unlike the other voices, Voice 4 has notes on beat 1 and beat 3. It has rests on beat 2 and beat 4. Its first note is translated to be -3 (negative 3). The note on beat 3 is translated to be a 1. We would therefore enter Voice 4 as:

```
*V4Q-3$1$
```

In all, our music file now consists of the following lines

```
K1#
NQ=90
M01  *V1Q$B$B
     *V2Q$6$6
     *V3Q$4$4
     *V4Q-3$1$
```

Before going on, let's stop and listen to what we have created so far. To do this, press the BREAK key to return control to the Command Line, then enter COMPILE to compile the file, and then enter PLAY to play the music.

After you have listened to your masterpiece, return to the Programming Buffer for continued editing by entering EDIT. Move the cursor below the last line entered by using the down arrow key.

Examining our sample sheet music in Figure 2, we can see that Measure 2 is exactly the same as Measure 1.

To type this second measure in, we could go to all the trouble of retyping measure 1 in (but naming it MØ2), or we can take advantage of the REPEAT command. To repeat Measure 1, type:

```
RØ1
```

This command tells the composer to repeat Part Ø1 until it encounters another part number, or encounters a repeat command. Our example will now look like this:

```
K1#
NQ=9Ø
MØ1  *V1Q$B$B
     *V2Q$6$6
     *V3Q$4$4
     *V4Q—3$1$
RØ1
```

Before you start beating your head against a table because you had trouble discerning rest positions in our example, rest assured that most sheet music clearly defines rests. However, the purpose in using that example was to make you much more keenly aware of timing, which is crucial to correctly transcribing music into ***SYMPHONY-90*** format.

ANOTHER SAMPLE SESSION

Our next music session contains more "interesting" music, and is longer, consisting of 4 measures in 3/4 time. Figure 3 shows our next example, with voice notation added:

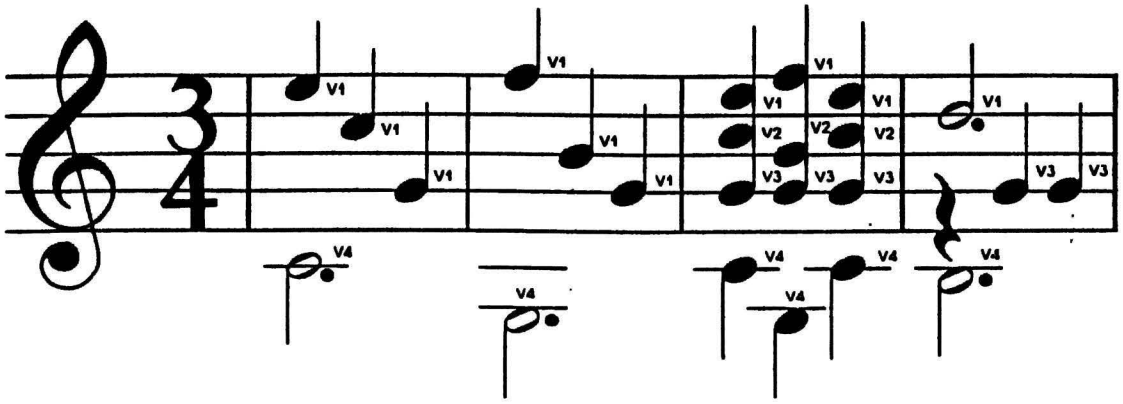


Figure 3.

This sample session describes the music shown in Figure 3. Pay careful attention to the sample sheet music and the SYMPHONY-90 transcription below. Follow each line of the transcription and do not proceed to the next until you fully understand how the information was interpreted. Notice the "dotted" Half Notes accompanying the Quarter Notes. Refer to Appendix F to understand Whole and dotted notes.

```
KØ#
NQ=8Ø
V1YA V2YB V3YC V4YD
PØ1
MØ1  *V1Q974
     *V4H.Ø
MØ2  *V1QA64
     *V4H.—3
MØ3  *V1Q9A9
     *V2Q767
     *V3Q444
     *V4QØ—3Ø
MØ4  *V1H.8
     *V3Q$44
     *V4H.—1
```

Notice that there are no sharps or flats in the Key Signature. Although the KØ# could have been left out, by using it, it reminds us (and others) that this is how we want it. Also notice that the time signature tells us that there are 3 quarter-note beats to a measure.

Although our music example uses 4 voices, Measure 1 only requires 2 -- V1 for the melody and V4 for the bass. The melody notes are normally the highest notes, and bass notes are usually the lowest (but this is by no means a rule). Voice 2 and Voice 3 are not required until the third measure.

The first measure contains a dotted half note, H., followed by the note symbol for middle C (Ø). A Half Note lasts twice as long has a Quarter Note. A dotted note has its value increased by 1/2. So a dotted Half Note lasts as long as 3 Quarter Note beats. In the indicated 3/4 time, three beats fills the measure.

Measure 2 has the same timing as Measure 1, except that it uses different note symbols.

Measure 3 uses all 4 voices in 4-part harmony, all using Quarter Notes.

Measure 4 uses only 3 voices, not using Voice 2. Also notice that Voice 3 begins with a Rest. Also notice that Voice 3's two Quarter Notes are the same note, and since we do not have any articulation between them, they sound like one continuous note; a Half Note in this case. In fact, we could have re-written is line to become: *V3Q$H4, and it would have sounded exactly the same. However, in most cases when humans play the music on instruments, they, by design of intent or not, insert some articulation between these notes, to allow one to discern that they are in fact two separate notes. Thus we could change the line to: *V3Q$4'4.

The ' is used as an articulation mark, which will make the note it precedes sound more distinct. Go ahead and make this change. Also, to produce an effect called Staccato, which is a very short pause, as opposed to the longer articulation pause, add a semicolon ";" behind each note in Voice 1, Measures 1 and 2:

```
MØ1  *V1Q9;7;4;

MØ2  *V1QA;6;4;
```

Once completed, return to the Command Line and COMPOSE and PLAY this selection. After you listen to your music, it might be a good idea to experiment with articulation and staccato, noting the different effects they produce. Refer to Appendix F for variations on these two effects.

A FINAL EXAMPLE

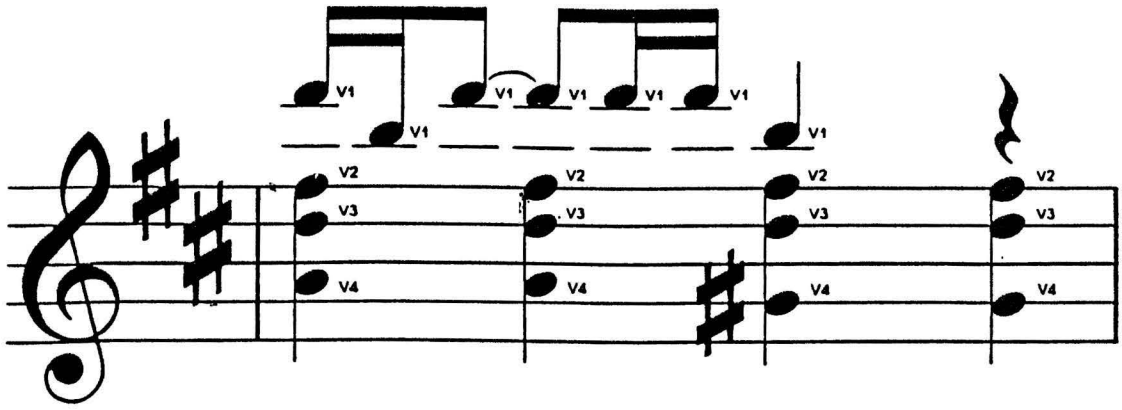


Figure 4.

When music scores do not show a Time Signature, it is considered to be in Common Time, or 4/4 Time. This example is a very important one, and has been saved for last in order that you should understand all other aspects of transcription before having to work with transpositions. Also be sure to refer to Appendix F to note the codes for some of the notes not yet covered.

Type in, COMPOSE, and PLAY the sample. When you are done typing it in, it should look like the following example:

```
K2#
NQ=8Ø
V1YA V2YB V3YC V4YD
PØ1
MØ1  *V1SF'D'IFF;SF;F;QD$
     *V2QAAAA
     *V3Q8888
     *V4Q554#4
```

When you played it, you may have noticed that the melody played by Voice 1 (V1) was so high that it was barely audible. A simple change to the file can be performed which will bring the score's pitch down to a much more comfortable level. This can be done in more than one way.

The first method is to bring a voice down individually. If we chose to lower only Voice 1, we could add the command V1U-7. This tells the composer to transpose Voice 1's note down (down due to the negative sign) 7 whole steps. Try it. Change the line "PØ1" to:

```
PØ1 V1U-7
```

Now COMPOSE and PLAY the file again. You will find Voice 1 much more audible, but the other voices are still too high. A way around this is to use a different method. If we changed the modified line to:

```
PØ1 <9
```

what we will have done is instructed the compose to transpose ALL voices in the current and subsequent measures down. In this case we are transposing the notes down 9 semitones (half steps). Alter the line and

then COMPOSE and PLAY the file again. The file's notes have been lowered by 9 semitones, or 1 half-step over an octave (an octave covers a range of 8 semitones). If you wish to, experiment with different semitone levels. Remember that the value is a single hexadecimal digit. Hexadecimal digits range from Ø-9, and A-F, where A represents 1Ø and F represents 15. As you use higher values, such as A, C or E, the tonal quality of each voice improves, reducing distortion.

Notice the 2 Sixteenth Notes (S) in V1, each articulated using an apostrophe. Also notice the Eighth Notes (I). Notice that from observing the arched line between them, that these two notes should be tied, thus expressing them as a single notes with a duration of a Quarter Note (2 Eight Notes). Finally notice the use of a staccato in V1 to make the melody more distinct.

Notice the sharp used within the measure for Voice 4. A sharp applied to a note means that the note and all subsequent notes of the same symbol will be played 1 semitone (1/2 step) higher than normal for the duration of the measure. By default, the composer will carry a sharp (or flat) through to the end of the measure (see the O modifier for variations from this result in Appendix F). Due to the default nature of the composer, we entered the third note for Voice 4 as 4#, and the fourth was written as strictly 4, which will play as a 4# no matter which voice plays it.

Now alter V2, V3, and V4 to the following:

```
*V2QA;A;A;A;
*V3QB;8;8;8;
*V4Q5;5;4#;4;
```

We have added a staccato (long staccato, to get technical) to each of their notes. COMPOSE and PLAY this altered version and you may be surprised to hear an entirely different musical effect. Due to shortening the duration of each chord in the measure due to the addition of the staccato, the result is a much sharper accent on each of the four beats.

CHAPTER TWO

USING THE COMPOSER

This chapter will describe both the command functions of the ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** as well as the Music Programming Language understood by ***SYMPHONY-90***. Familiarity with both is *essential* to the composition and modification of your music files. Each command will be described, and usually accompanied by an example of usage. You are strongly encouraged to experiment with them, and adapt them to your own requirements. Doing so will give you the confidence you need to work comfortably and quickly with the composer system.

COMPOSER COMMANDS

Commands given to the composer are issued on the Command Line. You can always ensure that you are at the Command Line by pressing the BREAK key. Each command word is keyed off of its first character, and you are required to issue at least the first character of a command for the composer to understand what you want it to do. The fact is, the command words described are only exapanded for your personal reference, as the composer will ignore all characters following the first letter. Thus the command **NEW** can be abbreviated to **N**, the command **BOTTOM** can be abbreviated to **B**, and so forth. A brief summary will be described first, followed by a detailed description:

COMMAND	ABBREVIATION	MEANING
APPEND	A	*Add a file to the contents of the buffer*
BOTTOM	B	*Go to the bottom line of the file*
COMPOSE	C	*Compose the current music file*
DIR	D	*Display a disk directory of music files*
EDIT	E	*Edit the contents of the Programming Buffer*
GET	G	*Load, Compose, and Play the specified file(s)*
HARDCOPY	H	*Send a hardcopy of the buffer to the printer*
KILL	K	*Kill (remove) a file from the directory*
LOAD	L	*Load a file into the Programming Buffer*
NEW	N	*Clear the Programming Buffer*
PLAY	P	*Play the file in the buffer*
QUIT	Q	*Quit from the composer and return to DOS*
ROUND	R	*Play a continuous round (multiple GET) of files*
SAVE	S	*Save the buffer to the disk filename assigned*
TOP	T	*Go to the top line of the file*
UNNEW	U	*Recover the Programming Buffer after a NEW*
Z	Z	*Refresh the Programming Buffer after a DIR*
>	/	*Forward search for a string*
<	-	*Reverse search for a string*
?	?	*Display the voicing registers*
!	!	*Play from the current point in the buffer*
@	@	*Pinpoint a wrong section of the score*

APPEND

This command allows you to append a file to the *beginning* of the current contents of the Programming Buffer. For example, using this command you could add a file called **ALLEGRO** to the current contents of the buffer by issuing the command:

```
APPEND ALLEGRO
```

This command is handy if you wanted to create a medley of files. Always ensure that if part numbers are included in the files, that their number do not conflict with each other.

BOTTOM

This command moves the buffer pointer to below the last line in the buffer. This command is the same as using SHIFT-DOWN ARROW from within the the Edit Mode in the Programming Buffer.

COMPOSE

Before you can hear your musical masterpiece, it must first be compiled into a format understood by the playing part of the program. The code generated by the COMPOSE command cannot be saved to disk. Besides, it is usually *much* larger than the source code (the typed data). After completing a musical score, or after finishing only a portion of it, you can COMPILE and then PLAY it to check the file for errors. A music file does not have to be complete in order to compile it.

If while compiling the composer finds that it cannot hold *both* the typed score *and* the compiled code, it will ask you:

```
Overwrite buffer?
```

If you answer with the letter "Y", composition will continue, but the data in the Programming Buffer will be erased. Please ***do not*** answer with "Y" if you have not saved the file to disk, as the data will be lost. In this case answer "N", save the file, and then COMPOSE again, this time answering with "Y". Rest assured that this will rarely occur, if at all, but only with extremely large files.

DIR

You can obtain a disk directory of all files containing the extensions /SYM (Symphony-90 files) and /ORC (Orchestra-90 files) by using the DIR command. The DIR command defaults to drive :0, and so if you wish to view the music file directory of another drive, you must specify that drive. For example, if you wanted to view the music file directory of drive :1, you would issue the command:

```
DIR :1
```

or

```
DIR 1
```

The directory is displayed in 4-across (5-across on the Model 4) format. Once the listing is completed, the program will return to the command mode. If you have a file in the Programming Buffer, it will no longer be visible, however it will still be fully intact and resident in memory. You can refresh the screen, thus erasing the directory display, by issuing the "Z" command, or EDIT, TOP, or BOTTOM commands.

Be aware that if you are using a DOS other than those recommended by this manual, that the DIR command may crash the system. The Model III version is designed to follow the directory dump format specified for both LDOS 5.x and TRSDOS 1.3. Most other DOSes fail to follow this requirement. The Model 4 version follows the similar format for TRSDOS 6.x and LS-DOS 6.x. This has not been tested with other DOSes.

EDIT

This command moves cursor control into the Programming Buffer, where the cursor is displayed as a block graphic. You may notice that the Programming Buffer pointer will return you to the position you were the last time you were editing the buffer, unless you have loaded a new file, or had issued the TOP or BOTTOM commands. Edit mode commands will be discussed after the the program commands.

GET

This is a multifunction command which will LOAD, COMPOSE, and PLAY a file or a list of files. This command allows you to play a file, or a series of files, one after the other. Be aware that if the program finds that composing the file will require overwriting the Programming Buffer in order to have enough room for compilation, that this will be done automatically without notice. You can specify a drive name after a filename by separating them with a colon ":". The program will by default, if you do not specify a file extension, look for a file by the specified name with a /SYM extension. If it is not found, it will look for a file with a /ORC extension. If the specified file is not found, this will be reported on the Command Line and the GET command will be aborted. Press any key to erase the error message and return control to the Command Line.

For example, to consecutively play the files FLSHDC/SYM, AMADEUS/SYM, and LUFT99/SYM, you would issue the command:

```
GET FLSHDC/SYM AMADEUS/SYM LUFT99/SYM
```

or

```
GET FLSHDC AMADEUS LUFT99
```

HARDCOPY

This command will send a copy of the file to the printer. This hardcopy will be formatted exactly as it is displayed on the screen. Pressing BREAK during printout will stop this command.

KILL

This command will remove the specified file from the disk directory. You can include a drive specification if you wish. If you do not include an extension, then KILL will assume a file with a /SYM extension. Examples:

```
KILL HOTM
```

will remove a file named HOTM/SYM from the first drive encountered which contains that file.

```
KILL OLD/ORC:1
```

will remove a file named OLD/ORC from the disk in Drive :1.

LOAD

This command will load a specified file into the Programming Buffer, erasing the previous contents of the buffer (see APPEND if you wish to add a file to the buffer). You can include a drive extension. If you do not include a file extension, then the program will try to load a file with a /SYM extension. If such a file is not found, then it will try to load a file with a /ORC extension. Examples:

```
LOAD GLORIA
```

Will try to load a file named GLORIA/SYM from the first drive it is encountered on. If it is not found, then the program will try to load a file named GLORIA/ORC.

LOAD HERO /SYM: 1

This command will look to Drive :1 for a file named HERO/SYM.

NEW

This command will totally erase the contents of the Programming Buffer. If this command was performed by accident, refer to the UNNEW command for data recovery.

PLAY

The PLAY command tells the program to play the most recently composed music data. If the file has not been recently composed, then that function will also be performed. Remember that if you had used the COMPOSE command previously on a file, and then did some subsequent editing, that the data from the previous compilation will still be in memory, and so the old data will be played again. In cases such as this you must first manually COMPOSE the file to set it to the file's current state.

You can also play a file by starting at a specified PART number. For example, suppose you wished to play the current file starting at Part 22. For this you would issue the command:

PLAY 22

This is especially handy during the development phase where you have all previous parts correctly playing, and wish to begin testing in the newly edited portions.

STOPPING IN MID-PLAY: Please note that while the program is playing, you can stop the play mode and immediately return control to the Command Line by pressing the zero key "Ø" either on the top keyboard row or on the numeric keypad.

QUIT

This command leaves the program and returns control to your DOS Command Line (DOS Ready).

ROUND

This command works exactly like the GET command, except that after the last file in the list is played, it starts all over again with the first file in the list.

SAVE

When you have created a new music composition, or at any time during its development, you can save the contents of the Programming Buffer to a disk file. You save a file to disk by specifying a filename after the SAVE verb. If you do not include an extension, then a default extension of /SYM will be added by default.

SAVE MYMUSIC: 1

will save a file called MYMUSIC/SYM onto Drive :1. If the file already exists, then it will be overwritten by the current contents of the Programming BUffer.

SAVE MEDLEY /ORC

will save a file called MEDLEY/ORC on the first drive available for writing.

TOP

This command moves the buffer pointer to the first, top line in the buffer. This command is the same as using SHIFT-UP ARROW from within the Edit Mode in the Programming Buffer.

UNNEW

This is a handy instruction if you accidently issue the NEW command, or if you had executed NEW and then changed your mind. The UNNEW command will recover the contents of the buffer, except for the *very last* character in the Programming Buffer. This is a small price to pay. This command is also handy if you had left the program without first saving the buffer to a disk file. If you execute the UNNEW command as soon as you re-enter the program, the buffer will be recovered.

Z

This command refreshes the display of the Programming Buffer on the screen. This is handy after a DIR listing. Although the contents of the Programming Buffer are fully intact, it will be rendered invisible until you otherwise issue a TOP, BOTTOM, or EDIT command. Using the "Z" command simply updates the display of the Programming Buffer to its current status.

>

The right angle brace ">" allows you to search forward through the Programming Buffer and find the next occurrence of a specified string of characters, and place the location of the edit cursor on that line.

For example, to search for the next occurrence of the string M44, you would issue the command:

```
>M44
```

If the text is found, the line it is contained in will be moved to the top of the Programming Buffer, and control will return to the Command Line. If you were to issue the EDIT command afterward, the cursor position would be placed on that line.

<

The left angle brace "<" allows you to search backward through the Programming Buffer and find the next occurrence of a specified string of characters. This command works exactly like the previous command, except that it searches from the current line and toward the top of the file.

SPECIAL COMMAND FUNCTIONS

There are 3 additional command functions available than work differently that the instructions previously covered.

?

The question mark "?" will display the current voicing in effect (voice register assignments and transpositions) at the current position. The current position is determined by the position of the cursor the last time you were in the Programming Buffer (unless you had issued the TOP or BOTTOM commands). To use it, position the cursor anywhere within the current file, press the BREAK key to return control to the Command Line, and then issue the command "?". When you press the ENTER key, the current line in the Programming Buffer will be moved to the top of the Programming Buffer display, and the voicing registers will be displayed within the Command Line. Press any key to return control to the Command Line. This command is handy if you are curious about what voice registers are assigned to what instruments, and to check to ensure they are set as you intent for them to be.

This command also performs a partial COMPOSE function, so that you will have to re-COMPOSE the file if you wish to listen to it play in its entirety.

!

The exclamation mark "!" allows you to play for the current position in the Programming Buffer to the end of the file. This command will pass up all notes previous to the current line. This command is handy if you wish to check new data added to a music file, choosing to bypass previous data which may have already been checked or corrected.

@

This command allows you to move the Programming Buffer pointer to the point where you stopped the file's playing using the zero "0" key (see the PLAY command). After you have stopped play, which will return control to the Command Line, issue the "@" command. The line which was currently being played will be shifted to the top line of the Programming Buffer. Were you to use the EDIT command afterward, the cursor pointer would be placed on that line. This command is useful in pinpointing a bad section of a music file, so that you can immediately go to it and edit the offensive data.

This command also performs a partial COMPOSE function, so that you will have to re-COMPOSE the file if you wish to listen to it play in its entirety.

SPECIAL REMINDER ON COMMANDS

Always be sure that control is on the Command Line before issuing any of the commands listed so far in this chapter. These commands only work from the Command Line. By the same token, Edit Mode commands (to be discussed shortly) cannot be used while control is in the Command Line.

EDIT MODE COMMANDS

The Edit Mode is activated when you issue the EDIT directive from the Command Line. You know that the Edit Mode is active when the cursor is displayed as a graphic block and is blinking within the Programming Buffer area.

CONTROL (CTL) KEY

Many of the Edit Mode commands are issued using a Control key (CTL). Depending on the version of the program you are using, the CTL key will differ:

CLEAR Used as the CTL key on the Model III version. The CTL key is used by holding it down and pressing an additional key in conjunction with it. For example, CTL-D would be issued by holding the CLEAR key down and tapping the "D" key.

CTRL Used as the CTL key on the Model 4 version. The CTL key is used by holding it down and pressing an additional key in conjunction with it. For example, CTL-D would be issued by holding the CTRL key down and tapping the "D" key.

CURSOR CONTROL

These commands allow you to move the cursor while in the Edit Mode:

Left Arrow Move the cursor one position left.

Shift-Left Arrow Move the cursor to the start of the current line.

Right Arrow Move the cursor one position right.

Shift-Right Arrow Move the cursor to the end of the current line.

Up Arrow Move the cursor one row upward.

Shift-Up Arrow Move the cursor to the top of the file.

Down Arrow Move the cursor one row downward.

Shift-Down Arrow Move the cursor to the bottom of the file.

CTL-Up Arrow Move the cursor to the top of the displayed buffer.

CTL-Down Arrow Move the cursor to the bottom of the display buffer.

CTL-F Scroll the Programming Buffer Window forward one screen.

CTL-R Scroll the Programming Buffer Window in reverse one screen.

CTL-T Tab the cursor 4 spaces over to the right. Good for indenting.

EDITING COMMANDS

These command allow you to exercise greater control of your data.

BREAK Pressing the BREAK key will quit the Edit Mode and return control to the Command Line.

ENTER Pressing the ENTER key forces the editor to accept the current line and move the cursor down to the start of the next line.

CTL-CLEAR Delete all text from at and to the right of the cursor.

CTL-A Append the current line to the end of the line above it.

CTL-D Delete a character at the cursor position. Any character on the line to the right of the cursor is shifted left one character.

F2 Model 4 only. Delete a character. Same as CTL-D.

CTL-I Insert a space. A space is inserted at the cursor position, and data which was at and to the right of the cursor is shifted rightward. If any data shifts past the end of the displayed line, then it is lost.

F1 Mode 4 only. Insert a space. Same as CTL-I.

CTL-N New line insert. Inserts a blank line at the current cursor line. The line at and those below will be shifted down by one line.

CTL-S Split the line at the current cursor position. This command moves the characters at and to the right of the cursor down one line.

CTL-U Unedit the current line. Handy when you change you mind about current editing on a line. Be aware that once you leave that line, that it cannot be unedited.

CTL-Y Yank the current line. This command deletes the line the cursor is on and moves all lines below it up one line.

BLOCKING COMMANDS

These commands allow you to move or copy whole sections of text from one place in the file to another, whether it be a single line or a whole series of lines. Be aware that blocks are considered to encompass whole lines. Thus if you have the cursor located at the end of a line and select a block marking command (CTL-B or CTL-E) that it is considered to include the *entire* line.

CTL-B Begin block. This command allows you to specify the beginning position of a block. Notice that a "B" will be displayed in the Command Line, indicating that the beginning of a block has been defined. Be aware that you can select a different beginning without cancelling, or you can specify the end of the block before you specify the beginning.

CTL-E End a block. This command allows you to specify the ending position of a block. Notice that an "E" will be displayed in the Command Line, indicating that the end of a block has been defined. Be aware that you can select a different ending without cancelling.

CTL-X Cancel blocking. This command allows you to cancel block marking after the beginning or ending or both have been selected.

CTL-C Inserts a copy of the marked block of data to the line where the cursor is currently resting. The current line will be moved down as many lines as needed to insert a copy of the new data. The original text will be left intact.

CTL-M Moves the marked block of data to the line where the cursor is currently resting. The current line will be moved down as many lines as needed to insert the marked block of data. The original text will be removed from its original position.

NOTE CONVERSION

Tables 1 and 2 will probably be the most referred to charts in the manual. As such you may wish to tab these pages with a piece of tape. Make special note of the *Middle C* positions in both of the tables. Also notice that the circled notes (called a *Note Symbol* by the Music Programming Language) are those which are used by the Music Programming Language, and the smaller letters above each circle represent the actual note as used on sheet music (called a *musical note name).* Also be aware that the composer uses the letters A-G to represent 10 through 16.

NOTE SYMBOL	MUSICAL NOTE NAME	
+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	
Ø	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	
$	REST	

Table 1.

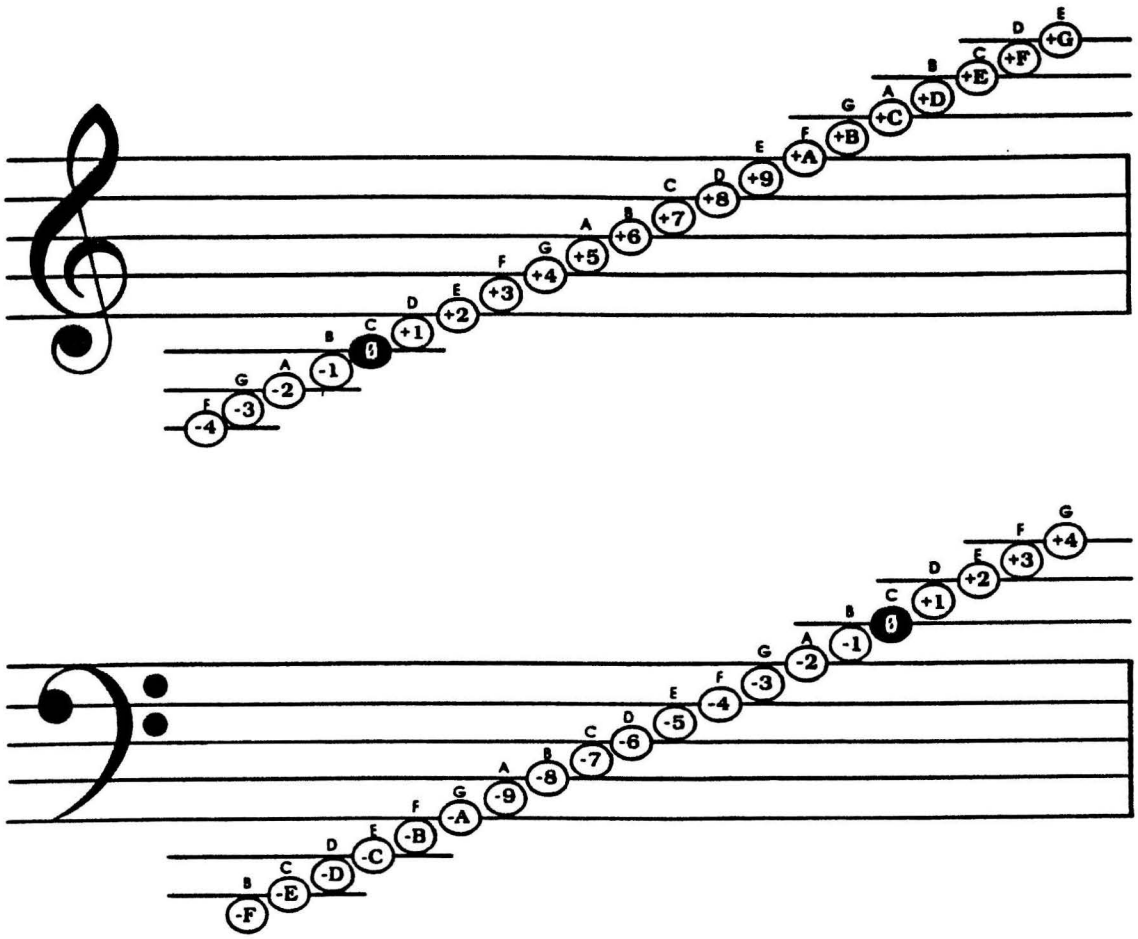


Table 2.

NOTES ON SIGN NOTATION

A note's staff position is specified by its association to a fixed position on the staff. Middle C is centered at location **Ø**, and all notes above it are defined by a positive displacement. Notes below it are defined by a negative displacement.

When notes are entered on the Treble Clef (*), positive values do not need to be typed with the plus (+) prefix. Thus *+4* would be entered as simply *4*. However, negative value notes on the Treble Clef *must* have the minus (-) prefix added.

When Notes are entered on the Bass Clef (@), negative values do not need to be typed with the minus (-) prefix. Thus *-2* would be entered as simply *2*. However, positive value notes on the Bass Clef *must* have the plus (+) prefix added.

MUSIC PROGRAMMING LANGUAGE FUNCTIONS

Aside of specifying note symbols, you must also specify durations of notes, and modifiers to change the presentation of your music to the way in which you choose to have it arranged and performed. These modifications include note types, articulations, flats, sharps, and accidentals.

NOTES

Note Value	Note name	Description
	*Whole note*	This note is held for the duration of 4 quarter notes in 4/4 time. Its Music Programming Language name is ***W.***
	Half Note	This note is held for 1/2 the length of a Whole Note. Its Music Programming Language name is ***H.***
	Quarter Note	This note is held for 1/4 the length of a Whole Note. Its Music Programming Language name is ***Q.***
	*Eight Note*	This note is held for 1/8 the length of a Whole Note. Its Music Programming Language name is ***I.***
	*Sixteenth Note*	This note is held for 1/16 the length of a Whole Note. Its Music Programming Language name is ***S.***
	*Thirty-second Note*	This note is held for 1/32 the length of a Whole Note. Its Music Programming Language name is ***T.***
	*Sixty-fourth Note*	This note is held for 1/64 the length of a Whole Note. Its Music Programming Language name is ***X.***
	Dotted Note	Placing a period (.) after the note will increase its length value by 1/2. Thus ***HØ*** is a dotted half note played at Middle C (its duration is equal to three Quarter Notes).
	*Triplet*	Placing a colon (:) before a note makes the note a triplet. Thus ***:Q*** indicates that any note symbols following this will be a quarter-note triplet (a triplet is played for the duration of 2 notes in normal time).
	*Rests*	Rests are indicated by using a dollar sign ($). Placing a $ after a note value will indicate a rest for the duration of a note of that value. Thus ***Q$ØØØ*** tells us that there will be a rest for 1 quarter note beat, and followed by 3 Middle C Quarter Notes.

ARTICULATION

SYMPHONY-90 plays consecutively placed notes as Legato (no space between each note). This ties the notes of equal value together into a much longer one. Often this is not the desired effect, and so you may wish to insert some space to allow for articulation of each successive note. Articulation of notes can be altered by using expression modifiers. Articulation is used to interject a small pause or rest at the end of a note, thus separating it from the next note. Articulation is accomplished by choosing from a selection of 2 portato pauses, or 2 staccato pauses.

'	*Short Portato*	Placing an apostrophe (') after a note shortens the duration of the note slightly. This articulation symbol allows the smallest pause between consecutive notes. A Short Portato shortens a note by an amount equal to 1/3 of a 1/128 note, and adds a pause of equal value.
"	*Long Portato*	Placing a quotation mark (") after a note shortens the duration of the note slightly longer that a Short Portato. The notes are articulated more than with a Short Portato. A Long Portato shortens a note equal to 2/3 of a 1/28 note, and adds a pause of equal value.
;	*Staccato*	Placing a semicolon (;) after a note creates a Staccato effect by shortening the note more than with a Long Portato. This causes the note to be much more distinctly separated between consecutive notes. A Staccato shortens a note by 1/4 of its value, and adds a rest of equal value.
,	*Intense Staccato*	Placing a comma (,) after a note create an Intense Staccato effect by replacing 1/2 the note's duration with a pause or rest. This produces the strongest articulation available. This also makes consecutive notes their most distinct.

Expression modifiers affect *only* the note they follow. If accidentals are also applied to the note, they must *precede* the modifier.

SHARPS, FLATS, AND ACCIDENTALS

Accidentals immediately follow the note they will affect. Accidentals remain in effect until then end of the measure, or until another accidental is defined within the measure. All voices within the measure are affected. See the "Z" command described later for altering this default.

#	*Sharp*	Placing a # after a note indicates an accidental sharp.
b	*Flat*	Placing a & after a note indicates an accidental flat.
♮	*Natural*	Placing a % after a note indicates an accidental natural.
##	*Double Sharp*	Placing a ## after a note indicates a double sharp.
bb	*Double Flat*	Placing a && after a note indicates a double flat.
♮#	*Natural Sharp*	Placing a %# after a note indicates a natural sharp.
♮b	*Natural Flat*	Placing a %& after a note indicates a natural flat.

PERIPHERAL FUNCTIONS

There are many things to consider when designing a music score using the Music Programming Language. These special features will be covered in this section.

( )*hex* — A music programming command which is enclosed within parenthesis and followed by a single hexadecimal (hex) digit tells the program that the enclosed grouping of music commands is to be repeated for the hex digit's count. Thus ***(S3;T52')2*** indicates that the enclosed sequence should be repeated 2 *ADDITIONAL* times (played 3 times total).

* — Placing an asterisk (*) in a line indicates that all notes which follow it on the current and subsequent lines are considered to be placed in the Treble Clef, until another clef signature is indicated.

\- — Placing a dash, or minus (-) sign in front of a note indicates that it should be played in the Bass Clef. This will have not effect on any previously defined Treble Clef (*) command.

/ — A slash in a line indicates that all text to the right of it on the line are to be considered as comments. This is handy for placing titles and special notation in your files.

<*hex* — A "<" followed by a single hexadecimal digit tells the composer that all notes on the current and following measures shall be transposed down the number of semi-tones indicated by the hex digit.

>*hex* — A ">" followed by a single hexadecimal digit tells the composer that all notes on the current and following measures shall be transposed up the number of semi-tones indicated by the hex digit.

@ — Placing an at-sign (@) in a line indicates that all notes which follow it on the current and subsequent lines are considered to be placed in the Bass Clef, until another clef signature is indicated.

\+ — Place a plus sign (+) before a note to play only that note in the Treble Clef. This will not affect any previous Bass command.

_ — Placing an underscore (_), created by typing SHIFT-@, indicates that all notes on the current and following measures are considered to be placed on the Percussion Clef, until another clef signature is indicated.

J — The letter "J" starts a Register Definition. "J" is followed by either the letter R *(Random)* or S *(Sinesoidal;* harmonic*)* to indicate the wave form type, eight hex digits to indicate the wave form description, and a ninth hex digit to indicate the volume of the register. See Appendix D and E for a discussion on register definitions.

K — The letter "K" followed by a single digit from Ø through 7, and either "#" or "&" defines the Key Signature. The digit indicates the number of sharps (#) or flats (&) in the key signature. Thus, **K1&** indicates that the key signature contains one flat (key of F).

M — The letter "M" followed by a number defines a Measure. The number is optional, but is handy for reference purposes. For example, **M26** indicates Measure 26. You can even use letters to define a measure, as the composer only looks for the letter "M", and scans for a single space. The Measure definition *must* be followed by *at least* one space. Thus **M12, MEASURE, M7654X,** and **M** all signify a Measure definition.

N — The letter "N" followed by a single note symbol (H, Q, I, S, or T) will define the Time Signature. This tells the composer what type of symbol gets a single beat. For example, NI indicates that an Eighth Note gets one beat.

=*hex* — The equal sign (=) followed by 2 hex digits sets the TEMPO of the music, or how "fast" it will play. See Appendix B for examples for tempo settings. Note that you can also combine this command with the "N" command. Thus NQ=8Ø sets the Time Signature to one Quarter Note to a beat, and sets the TEMPO to 8Ø hex.

O*digit* — The letter "O" followed by the digits "Ø", "1", or "2" specifies the manner in which accidentals shall be handled within a measure. By default, an accidental will affect *all* voices in a measure. By using O1 anywhere within a measure, the accidental will affect *only* the voice in which it is defined.

Using O2 will carry any accidentals through the current measure and into the next. This feature is handy when a measure must be split into two or more measures to comply with the 32 notes per voice per measure limit of the composer. Option O2 must be set in each and every measure in which accidentals are to be carried forward.

Option OØ will cancel both options O1 and O2.

P*hex* — The letter "P" followed by a 2-digit hex pair will define the beginning of a part, which will be named as the 2-digit pair. P1Ø, for example, begins Part number 1Ø. A part can be terminated by defining another part, the "R" repeat option (described below), or by using PØØ, which is a convenient part terminator (Part ØØ is by default the beginning of the file, and so subsequent PØØ definitions will not be referenced or interfere with the playing of the piece).

R*hex* — The letter "R" followed by a 2-digit hex pair causes the notes in the specified part number to be played again (repeated) using the currently defined registers and tempo, then proceed from the current position in the file. Since the repeat command only picks up the NOTES from the specified Part, you may wish to ensure that voicing and tempo are correctly set if they had been subsequently altered after the part had been defined. This is a handy feature if you wish to change tempo or voices between repeats.

Any Measure which follows a Repeat command must start with a new part number. If a specific Part number for the subsequent data is not required, you can use the PØØ definition.

U — The letter "U" followed by a plus (+) or minus (-) and a single hex digit forces all subsequent notes in the specified voice to be transposed up or down by the number of whole steps specified by the hex digit. Thus V1+7 would raise Voice 1 by seven steps, or one octave.

This command is most useful when you with to address a Clef out of the normal range of the Treble Clef (*) or Bass Clef (@), such as the soprano, alto, tenor, and bass. To take advantage of this ability, you can define voices by applying a number which is the displacement from Middle C in the clef being defined to Middle C in the Treble Clef. For example, to define Voices 1-4 as soprano, alto, tenor, and bass respectively, you could use the command:

V1 U–2 V2 U–6 V3 U–8 V4 U–C

V — The letter "V" followed by the digits 1, 2, 3, 4, or 5 tells the composer which voice is being programmed. For example, V2 indicates Voice 2.

Y — The letter "Y" follows a Voice specification, and then is itself followed by the letter "A", "B", "C", "D", or "E". This allows you to assign an instrument to a particular voice. For example,

V3YD tells Voice 3 to emulate an organ (See Appendix D and E for changing the instrument registers A-E to sound like different instruments).

Zdigit The letter "Z" followed by a single digit of Ø-9 allows you to define which voices will play from what side of your stereo (left or right). See the next section on Stereo Mapping.

STEREO MAPPING

Stereo mapping allows you to define through which speakers what voice will be played. Channel Left indicates the left speaker, and Channel B indicates the Right Speaker. Model III users who have only 4 voices defined should note that voice 5 will be ignored if it is not included in your definition.

Stereo mapping is defined by using the "Z" command, followed by a single digit from Ø through 9.

Command	Left Channel		Right Channel			Minimum Voices needed
ZØ	V1	V2	V3	V4	V5	4,5
Z1	V1	V3	V2	V4	V5	4,5
Z2	V2	V3	V1	V4	V5	4,5
Z3	V1	V4	V2	V3	V5	4.5
Z4	V2	V4	V1	V3	V5	4,5
Z5	V3	V4	V1	V2	V5	4,5
Z6	V1	V5	V2	V3	V4	5
Z7	V2	V5	V1	V3	V4	5
Z8	V3	V5	V1	V2	V4	5
Z9	V4	V5	V1	V2	V3	5

Please take note that if mapping symbols (Zx) appear somewhere other than at the beginning of a part, a voice can become disconnected from its register. Therefore you should avoid remapping voices between measures unless the measures are separated by parts, or if all voices being used are playing using the same register.

NOTE LIMITATIONS

Special notice should be given to the fact that the structuring of ***SYMPHONY-90*** limits you to 32 notes per voice per measure. This limitation extends to the inclusion of rests and articulations (an articulation produces both a note and a rest). If you are writing music which will move outside of this limit, you may choose to split each measure into two separate measures, adapting the Key Signature accordingly. This may be crucial where you will be using a lot of articulation on short notes, or are using a lot of 1/64 notes (such as with bagpipe music).

APPENDIX A: ERROR MESSAGES

When ***SYMPHONY-90*** detects an error, a message describing the error will be displayed on the Command Line. In most cases the line in question will be displayed at the top of the programming Buffer. Keyboard input will be suspended until you press the ENTER key to acknowledge the error.

ERROR MESSAGE	MEANING OF MESSAGE
BAD PART #	This may be due to the Part number defined by the PART (P) command, or referenced by the REPEAT (R) command being in error, the specified Part cannot be found in the file, or the Part number is *already* defined (P00 is considered to be an unnumbered Part, and so can be used any number of times). This can also be due to a Part not being found due to the file needing to first be compiled using the COMPOSE command.
MEASURE O/F	Measure overflow. The measure being compiled has been found to have more than 32 notes per voice. A remedy is to split the measure and carry forward any accidentals. Also, staccato and portato articulations should be counted as *two* notes because they produce a note and a rest.
MEMORY O/F	Memory overflow. There is not enough memory available to COMPOSE the file, even after overwriting the Programming Buffer, or there is not enough memory to LOAD or GET a file. This is a rare occurrence, and in most cases will never happen to anyone except those composing extremely huge pieces (and ***SYMPHONY-90*** can easily handle some pretty big pieces). A suggestion is to break the file up into two or more smaller pieces. Use the GET command to play them in sequence.
NOTE O/F	7 dotted triplets, dotted whole notes, and other unusual composition mixtures cannot be properly compiled.
PARAMETER ERROR	This can be caused by a number of things: 1) The note specified in the Time Signature is not a H, Q, I, S, or T. 2) The number of flats or sharps specified in the Key Signature is too large, or is not followed by a Flat (&) or Sharp (#) symbol. 3) The specified Voice is not 1, 2, 3, 4, or 5. 4) The specified Register is not A, B, C, D, or E.
SYMBOL OUT OF CONTEXT	This error can be caused by the Tempo symbol (=) not being followed by 2 hex digits, a command operand containing an invalid hex digit, or the compiler was expecting a valid hex digit or note symbol and not finding it.
SYS I/O ERROR	An error occurred while communicating with the DOS system.
UNKNOWN COMMAND	The user attempted to issue an invalid command to ***SYMPHONY-90***.

Any other errors reported are DOS errors, and you should refer to the error section in your DOS manual to interpret them if you do not understand them.

APPENDIX B: TEMPO CONVERSION

While a piece of music is playing, you can press any conbination of the 1-7 number keys to change the tempo. The more keys pressed down, the slower the tempo. Releasing the keys causes the play to resume at the tempo it is programmed at in the file. This is a convenient way to experiment with tempo settings and find a tempo best suited to your music file.

By experimenting thus, you can refer to the chart below to see what hexadecimal value you should use for that key combination in your file, if you decide that you want to use your keyed tempo as the new programmed tempo for the file.

Suppose, for example, you like the tempo played when you hold the 1, 3, 6, and 7 keys down simutaneously. By referring to the chart, you can see that this key combination corresponds to the hex value *CA*. You could then go to the tempo specification in your file and set the tempo to the new value of "CA". An example would be *NQ=CA*.

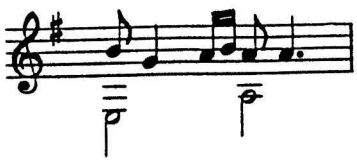
Tempo setting below 8Ø hex may produce shifts in frequency which may be undesirable. Also, you may have to modify the Time Signature parameter to obtain a tempo between 8Ø and FE. By increasing the note value in the Time Signature, you can double the value of the tempo. Thus NQ=CØ is preferred over NI=6Ø.

Finally, the zero key (Ø) is used to instantly stop playing the music and return control to the Command Line.

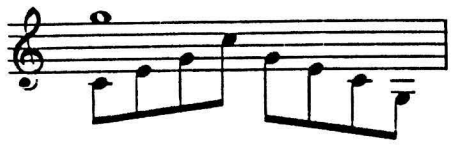
DIGITS	HEX	DIGITS	HEX	DIGITS	HEX	:	DIGITS	HEX
1234567	FE	123_67	CE	1234_7	9E		123_56_	6E
_234567	FC	_23_67	CC	_234_7	9C		_23_56_	6C
1_34567	FA	1_3_67	CA	1_34_7	9A		1_3_56_	6A
__34567	F8	__3_67	C8	__34_7	98		__3_56_	68
12_4567	F6	12__67	C6	12_4_7	96		12__56_	66
_2_4567	F4	_2__67	C4	_2_4_7	94		_2__56_	64
1__4567	F2	1___67	C2	1__4_7	92		1___56_	62
___4567	FØ	____67	CØ	___4_7	9Ø		____56_	6Ø
123_567	EE	12345_7	BE	123___7	8E		1234_6_	5E
_23_567	EC	_2345_7	BC	_23___7	8C		_234_6_	5C
1_3_567	EA	1_345_7	BA	1_3___7	8A		1_34_6_	5A
__3_567	E8	__345_7	B8	__3___7	88		__34_6_	58
12__567	E6	12_45_7	B6	12____7	86		12_4_6_	56
_2__567	E4	_2_45_7	B4	_2____7	84		_2_4_6_	54
1___567	E2	1__45_7	B2	1_____7	82		1__4_6_	52
____567	EØ	___45_7	BØ	______7	8Ø		___4_6_	5Ø
123__67	DE	123_5_7	AE	123456_	7E		123__6_	4E
_23__67	DC	_23_5_7	AC	_23456_	7C		_23__6_	4C
1_3__67	DA	1_3_5_7	AA	1_3456_	7A		1_3__6_	4A
__3__67	D8	__3_5_7	A8	__3456_	78		__3__6_	48
12___67	D6	12__5_7	A6	12_456_	76		12___6_	46
_2___67	D4	_2__5_7	A4	_2_456_	74		_2___6_	44
1____67	D2	1___5_7	A2	1__456_	72		1____6_	42
_____67	DØ	____5_7	AØ	___456_	7Ø		_____6_	4Ø

APPENDIX C: PROGRAMMING EXAMPLES

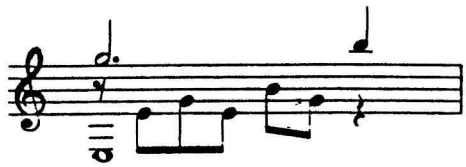
Numerous sample measures of music are displayed in this appendix to show you how to transcribe a variety of diverse musical scores to a format acceptable by ***SYMPHONY-90***. All samples are in 4/4 time. No Key Signatures are included.



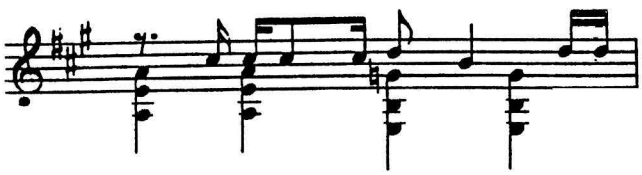
```
*V1I6Q4S5'6'I5'Q.5
@V2H52
```



```
*V1WB
*V2IØ24742Ø—3
```

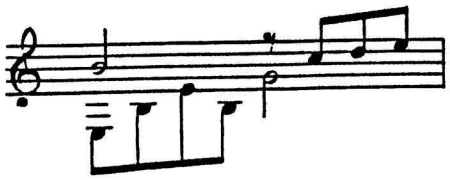


```
*V1H.BQD
*V2I$2426 4Q$
*V3W—5
```

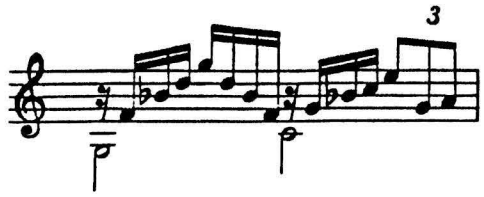


```
*V1I.$S7'7'I7'S7'I8Q6S8'8'
*V2Q5'54%4
*V3Q2'2—1'—1
@V4Q2'25'5
```

```
*V1TA'9I.AQAAS9'7#'8#'9'
*V2H1#Ø#
```

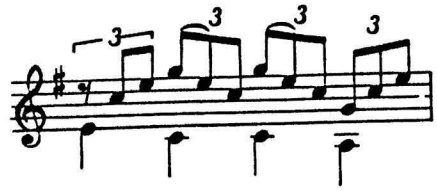


```
*V1H6I$789
*V2I–5–12–1H4
```

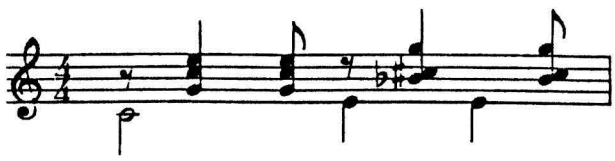



```
*S$3'6&'8'B'8'6'3'$4'6&'7'I:945
V2H–3Ø
```

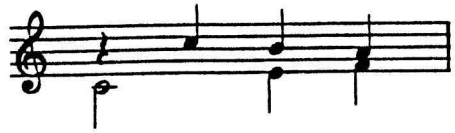
```
*I97'Q7I9753
V2Q5'54%'4
V3Q3'3'3'Ø
```




```
*I:$7'9'B9'7'B9'7'4'7'9'
V2Q2Ø'Ø–2
```



```
*I$Q9'I9$QB'IB
V2I$Q7'I7$Q7#'I7
V3I$Q4'I4$Q6&'I6
V4HØQ2'2
```



```
*Q$765
V2HØQ23
```

APPENDIX D: SPECIAL INSTRUMENTATION

A lot of the ***SYMPHONY-90*** music would be pretty boring if it were capable of playing only one type of instrument. The ***SYMPHONY-90 MUSIC COMPOSER SYSTEM*** allows you to create a great symphonic atmosphere by providing you with the ability to arrange your music with up to five instruments and percussion (see Appendix E for percussion examples). This Appendix shows you both the default instrumentation, and how to change the instrumentation.

The default instruments was chosen to provide you with a wide range of musical sounds. However, sometimes these five default instruments are not enough. As such they have been designed to be easily altered, so that you can change them to imitate other instruments, or create special effect instrumentation of your own. With ***SYMPHONY-90*** you have the capability to create sounds using harmonic or percussive sounds, or combine them by utilizing creative voicing techniques.

An instrument is assigned to a Tone Color Register, commonly called a Register in this manual. A register definition consists of the following data:

The letter "J".
The Register name it's assigned to (A, B, C, D, or E).
The wave form type: R for Random, or S for Sinesoidal.
A list of 8 hex digits comprising the weights of partials 1-8.
A single hex digit representing volume.

The default Registers are all harmonic (waveform type S -- Sinesoidal). Percussion sounds may be produced using either waveform type Random or Sinesoidal.

The harmonic type registers are defined by eight individual sine waves, called partials. A partial is an integer multiple of the fundamental frequency. The first partial represents the fundamental frequency. The second partial is 2 times the fundamental, the third partial is 3 times the fundamental, and so on through the eighth partial. These eight partials define the weight or the relative strength of each partial in the Register. It is quite easy to modify a Register to create your own musical instrument.

By default, the initial Registers for 2Mhz (Model III) are set at:

JASEFA5ØØØØE	Defines Register A; *trumpet*
JBS48F8ØØØØF	Defines Register B; *oboe*
JCSEØ5ØØØØØA	Defines Register C; *clarinet*
JDSF4Ø8ØØØØB	Defines Register D; *organ*
JES4F28ØØØØD	Defines Register E; *violin*

The defaults for 4MHz operation (Model 4, fast Model III) are set at:

JASEFA54EØØE	Defines Register A; *trumpet*
JBS48F8F2ØØF	Defines Register B; *oboe*
JCSEØ5ØF8ØØA	Defines Register C; *clarinet*
JDSF4Ø8ØØØØB	Defines Register D; *organ*
JES4F2814ØØD	Defines Register E; *violin*

The TRUMPET, defined as default Register A, is programmed as follows:

Symbol	Meaning
J	*begin Register definition*
A	*define as Register A*
S	*Sinesoidal waveform (harmonic)*
E	*Partial # 1; fundamental, weight E (near maximum)*
F	*Partial # 2; first harmonic, weight F (maximum)*
A	*Partial # 3; second harmonic, weight A*
5	*Partial # 4; third harmonic, weight 5*
Ø	*Partial # 5; fourth harmonic, weight Ø*
Ø	*Partial # 6; fifth harmonic, weight Ø*
Ø	*Partial # 7; sixth harmonic, weight Ø*
Ø	*Partial # 8; seventh harmonic, weight Ø*
E	*Volume E (near maximum)*

Every register definition must contain weight values for all eight partials. Partials having a weight value of zero (Ø) do not affect the waveform.

Due to the fact that it takes a sizeable amount of time to generate a harmonic wave, the compiler will generate them only when the register definitions change. You may notice that the first time you compile a file, it may take a while longer because at that time *all five* registers are having their waveforms generated. Further, all Registers are defined to default values unless they are redefined by your music file.

Redefining a Register is as easy as entering a complete Register definition. For example:

JASEFA5ØØØØF

This redefinition simply changed the volume of Register A from a volume of "E" to a volume of "F". It is also just as easy to alter the weights of any of the partials to cause it to generate waveforms for a different instrument. If you set all partials to Ø, then the Register will be silent. Please note that if you are operating on a 2MHz Model III that you should leave partials 5 through 8 as Ø! If you are operating at 2MHz, then ***SYMPHONY-90*** will automatically zero out partials 5 through 8 of the default Register settings. However, it is your responsibility to ensure that they are Ø during your own definitions, as they will not be likewise automatically nulled. Users with 4MHz capability have the ability to use partials 5 through 8 effectively.

SAMPLE WAVEFORMS

Following are a list of sample waveforms for various instruments. All you have to do is replace the "x" with the letter A, B, C, D, or E, do define that Register to the desired instrument. Also note that these register definitions are set up for 4MHz usage. If you are confined to 2MHz operation (Model III without a fast clock), then be sure to place a Ø in partials 5 through 8. Finally, you may wish to change the volume setting for the instrument to one of your preference (all set at "F" in these examples).

JxSA284ØØØØF	*/CHIMNEY FLUTE*
JxSØØØ2ØØØ4F	*/CONCERT PICOLO*
JxSE2Ø2ØØØØF	*/FOREST FLUTE*
JxSE844ØØØØF	*/GERMAN FLUTE*
JxSE242ØØØØF	*/MAGIC FLUTE*
JxSCEEAØ4Ø2F	*/MELOPHONE*
JxS666ØØØØØF	*/MELODIA*
JxS8E48Ø4Ø2F	*/SOLO OPEN FLUTE*
JxS468Ø22ØØF	*/VIENNA FLUTE*
JxS48EA64Ø2F	*/GRAND VIOLIN*
JxSØ8C646Ø2F	*/ORCHESTRAL VIOLIN*
JxS68C6Ø2Ø2F	*/CELLO*
JxS48A2Ø2Ø6F	*/VIOLA D' AMORE CELESTE*
JxSACE2Ø4Ø2F	*/VIOLIN*
JxS82C442Ø6F	*/VIOLINCELLO*
JxSØØ2242ØAF	*/APFELREGAL*
JxSA6884Ø2F	*/BALLAD HORN*
JxS44A624Ø6F	*/BAROQUE OBOE*
JxS8AAAAAO8F	*/BAROQUE TRUMPET*
JxSCØA222Ø2F	*/BASSOON*
JxSA2E266Ø8F	*/BELL CLARINET*
JxSACEF86Ø4F	*/BUGLE*
JxSØAØAØAØ8F	*/CLARION HARMONIQUE*
JxSAØCØØ6Ø2F	*/CLARINET*
JxSØ424Ø2ØØF	*/CONTRA OBOE*
JxS2Ø286Ø2F	*/DULZIAN*
JxS262A26ØCF	*/EGYPTIAN BAZU*
JxS84E824Ø8F	*/ENGLISH HORN*
JxSAFEF86Ø4F	*/FLUGELHORN*
JxSEABBØØØØF	*/FRENCH HORN*
JxSCAA2Ø2ØØF	*/HUNTING HORN*
JxSEFFCFFØFF	*/MILITARY FANFARE*
JxS2ØAFØ8ØØF	*/MUSETTE MIRABILIS*
JxSAAFF44Ø6F	*/MINOR TRUMPET*
JxSØ2A2Ø8Ø6F	*/ORIENTAL REED*
JxS8EAEEEØEF	*/POST HORN*
JxS42Ø48AØ6F	*/ROHRSCHALMEI*
JxSA622ØØØØF	*/SACKBUT*
JxSFC6ØØ4Ø2F	*/SAXAPHONE*
JxSAE8CØ8ØEF	*/SILVER TRUMPET*
JxSFF8FFFØFF	*/SOLO TUBA*
JxSCFCE6ØØØF	*/STENTOR HORN*
JxSEFFEA6Ø2F	*/TROMBONE*
JxSCEE222ØØF	*/WALDHORN*

```
JxS26ACCC04F     /BANJO
JxS40060004F     /BELLS
JxS26020202F     /CELESTA HARP
JxS2C020002F     /CHIMES
JxS0200000002F   /GLOCKENSPIEL
JxS48C88402F     /GUITAR
JxSA4022200F     /HARP
JxS600C0000F     /MARIMBA
JxSC4020600F     /MARIMBA HARP
JxS2400400008F   /ORCHESTRAL BELLS
JxS6F220002F     /VIBRAHARP
JxS0A0A0E0EF     /ZINK
JxS2002220202F   /VOX HUMANA
JxS46A24404F     /GAMBA
JxSC8422200F     /CLARABELLA
JxS26ACCC0CF     /TOY TRUMPET
JxS26E06402F     /SARRUSOPHONE
JxSFFFFFF0FF     /OPHECLEIDE
JxSFCA00000F     /SOLO SAXOPHONE
JxSC0C0A202F     /MELODY CLARINET
JxS0E080400F     /CORNO OCTAVE
JxS2004000008F   /MINIATURE BELLS
JxS28222202F     /ETHEREAL VIOLINS
JxSA8642202F     /PIANO
JxR0505779FF     /CASTINETS # 1
JxSF00FA00AF     /CASTINETS # 2
```

APPENDIX E: PERCUSSION

Percussion instrumentation and special effect instrumentation can be created by defining a Register as Random (R), rather than the usual Sinesoidal (S). What this waveform type does is that instead of using harmonic sine waves, it uses a random number generator. In this case, the eight partials are divided up into two groups of four digits each. The first four digits are used to generate a random "seed", and a the last four digits are used by a "randomizing" function. You can look at these like the RND(Ø) and RANDOM functions in BASIC, respectively, except that in this case you can choose what the initial values will be. You are strongly encouraged to experiment with these seed values, as there are literally *billions* of possible combinations. Because of this vast number, few generalizations can be made about the kinds of percussive effects available.

Be it as it may, here are a couple of samples to get you started:

```
JBR1Ø1ØØØØ17      /Register B; squeaky sound
JCRØØØ3ØØØ57      /Register C; scratchy sound
```

Notice how these definitions differ from the Sinesoidal definitions of Appendix D. We use the letter "J" just like with other instrument definitions, as well as Register assignments (B and C, respectively, in this sample case). We also use 8 data digits. And finally the last digit is used to set the volume (at 7, in both cases). The major differences is that we use "R" instead of "S" to define the waveform, and the 8-digit partial field is now used as two groupings of seeds. In the first example, the hexadecimal grouping 1Ø1Ø is used as the random seed, and ØØØ1 is used by the randomizing function. The second example uses ØØØ3 as the random seed, and ØØØ5 is used by the randomizing function.

Sinesoidal waveforms can also be used for percussive instrumentation. For example:

```
JDS8ØØ11ØØ1D
```

In this case we use the "S" type waveform parameter in our definition of Register "D". We are also using the 8 data digits as 8 partials again. 2MHz Model III computer users can go ahead and utilize partials 5-8, as they are warned against using during harmonic (S-type) instrument definitions. Finally, we have set the volume to "D". This sort of "percussive" instrument produces more of a "special effect" type of sound, rather than something which might be produced on a normal percussion instrument.

To take advantage of percussive instruments, we must assign a voice to the register, and define it on a Percussion Clef, which is defined by the underscore "_" character, produced by pressing SHIFT-@. The Percussion Clef has 16 notes assigned to it, using the hex digits Ø-9, and A-F (*NO* minus "-" values are allowed).

The Percussion Clef also *inverts* the function of Articulation annotation in that rather than playing a note with a short inserted pause, it instead inserts a long leading pause and a short note (reversing "roles" of the notes and pauses). The quotation mark (") and apostrophe (') are best suited to this Reverse Articulation. This reversal allows you to generate the short percussive sounds of a drum, and allows you to enter drum rhythms using the same note values found in drum tabulation. Consider the following example:

```
JCS8ØØ111ØØD      /Define Register C, Sinesoidal waveform, volume D.
JER8ØØ111ØØB      /Define Register E, Random waveform, volume B.
                  /other registers use defaults.
                  /Articulation played on all notes.
PØØ <7            /Unnumbered part, played one octave down.
                  /Play standard scale using Register default D.
MØ1 V1YD *IØ'1'2'3'4'5'6'7'8'9'A'B'C'D'E'F'Q$
                  /now play same scale using random waveform without Percussion Clef.
MØ2 V1YE *IØ'1'2'3'4'5'6'7'8'9'A'B'C'D'E'F'Q$
                  /now play this scale using random waveform with Percussion Clef.
MØ3 V1YE _IØ'1'2'3'4'5'6'7'8'9'A'B'C'D'E'F'Q$
                  /now play this scale using sinesoidal waveform and Percussion Clef.
MØ4 V1YC _IØ'1'2'3'4'5'6'7'8'9'A'B'C'D'E'F'Q$
```

In review of these examples, you will note that Measure Ø1 (MØ1) sounded normal, using the default register D (organ). However, Measure Ø2 (MØ2) sounded unusual because the notes are not shortened using Reverse Articulation, due to using the Treble Clef rather than the Percussion Clef. Measures Ø3 and Ø4 (MØ3 and MØ4) are example of proper percussive register definitions, reverse articulation, and use of the Percussion Clef. Once a percussion register has been defined, you may wish to play a sampling by using the demonstrated scales in MØ3 and MØ4 to select notes which sound best. Numerous notes in any percussion register may produce no sound at all, or may not be very well suited for a particular piece of music.

Always remember that the underscore character "_" defines Percussion Clef with Reverse Articulation, and that all notes which follow it will be interpreted as percussion notes until a Treble Clef "*" or Bass Clef "@" symbol are encountered. Always remember when you set up your voicing to ensure that the proper clef is set. Thus after defining data for a Percussion Clef, you must afterward use "*" or "@" to set normal music voices, otherwise they would be interpreted as percussion voices. By the same token, be sure to set the Percussion Clef symbol "_" each time you wish to use a voice in this Clef.

PERCUSSION EXAMPLES

The following examples typify only a small number of the percussion effects possible when using ***SYMPHONY-90***. When setting up your music file you could enter one for each voice you want to use for a percussion instrument. The Register definition line should be entered at the beginning of the file. The following examples are all defined for Voice 4, and do not comprise a single piece of music, but rather typify single measure-length samplings possible. Remember, you are not "locked" into Voice 4 or Register E, but can use any voice or register available -- the particular voice and register selection is ultimately up to you.

You are strongly encouraged to experiment by altering these pieces, or creating your own original percussion lines after you have gained some experience with these.

```
JES8ØØ11ØØ1E      /define Register E, sine wave, volume E
V4YE               /assign Register E to Voice 4
```

4/4 TIME SIGNATURE SAMPLES

```
_V4IE"SF"C"ID"E"SF"C"IF"D"SF"C"
_V4QD"E"8"IE"E"
_V4SA"B"C"D"ID"8"QD"IB"A"
_V4QC"9"IF"D"E"9"
_V4Q8"I8"8"9"9"9"9"
_V4QC"9"D"IA"E"
_V4Q8"I9"D"8"D"A"A"
_V4QB"IA"8"E"8"6"C"
_V4Q8"SA"A"I9"F"SF"8"I8"E"
_V4QE"I9"F"Q7"I8"9"
_V4QC"F"I8"9"F"8"
```

3/4 TIME SIGNATURE SAMPLES

```
_V4S6"7"8"9"IE"8"E"8"
_V4IE"SE"B"I8"B"QD"
_V4I8"8"B"S:C"D"E"ID"E"
_V4(ID"SE"E"ID")1
_V4I8"8"QC"ID"E"
_V4S:(E"F"F")3IF"F"
```

APPENDIX F: COMMAND SUMMARY

EDITING COMMANDS

KEY	MEANING
BREAK	Return to Command Line
CLEAR	CTL key for Model III
CTRL	CTL key for Model 4
ENTER	Accept current line
CTL-CLEAR	Erase to end of line
CTL-A	Join lines
CTL-B	Mark BEGINNING of a block
CTL-C	Copy marked block
CTL-D	Delete character
CTL-E	Mark END of a block
CTL-I	Insert space
CTL-M	Move marked block
CTL-N	Insert new line
CTL-S	Split line at cursor
CTL-U	Unedit line
CTL-Y	Yank (delete) current line
F1	Model 4 alternate space insert
F2	Model 4 alternate character delete
Left Arrow	Move cursor left
Right Arrow	Move cursor right
Up Arrow	Move cursor up
Down Arrow	Move cursor down
Shift Left Arrow	Move cursor to start of line
Shift right Arrow	Move cursor to end of line
Shift Up Arrow	Move cursor to top of file
Shift Down Arrow	Move cursor to end of file
CTL Up Arrow	Move to top of window
CTL Down Arrow	Move to bottom of window
CTL-F	Scroll screen forward
CTL-R	Scroll screen in reverse
CTL-T	Tab cursor right 4 spaces

COMMAND LINE COMMANDS

COMMAND	OPERAND	FUNCTION
APPEND	*filename*	Append the named file into memory at the beginning of the current file.
BOTTOM		Move the Programming Buffer pointer to the end of the current file.
COMPOSE		Compile the current file.
DIR	*d or :d*	List all **/SYM** and **/ORC** files on the disk number (d) indicated in the operand. Default = **Ø**.
EDIT		Enter **Edit Mode.**
GET	*filename...*	**Load, Compose,** and **Play** the file(s) named in the operand.
HARDCOPY		List the current file to the printer.
LOAD	*filename*	Load the file specified into the buffer.
NEW		Erase the current file from Programming Buffer.
PLAY	*Part # or blank*	Play the current file starting with the specified Part number, or from the beginning of the file if the operand is blank.
QUIT		Return control to **DOS Ready.**
ROUND	*filename...*	Perpetual **GET.**
SAVE	*filename*	Save the current buffer contents to the file specified.
TOP		Move the Programming Buffer pointer to the start of the current file.
Z		Refresh the display of the Programming Buffer after a **DIR** command.
>	*<any string>*	Search forward for the next occurrence of the specified string.
<	*<any string>*	Search backward for the next occurrence of the specified string.
?		Display the voicing assignments from the current position in the Programming Buffer.
!		Start playing from the current position in the Programming Buffer.
@		Pinpoint position in Programming Buffer of last measure played when aborted using the **Ø** key.

MUSIC SCORE NOTATION AND INTERPRETATION

SYMBOL	MODIFIER	DEFINITION
(		Begin REITERATION
)	*hex digit*	REITERATE the number of times specified by the modifier.
*		Unless otherwise indicated, all notes following are considered to be TREBLE Clef or +.
/		Comment. The remainder of the line will be ignored.
<	*hex digit*	All notes in the current and following measures will be transposed DOWN the specified number of semi-tones.
=	*2-digit hex*	Set the TEMPO to the value of the modifier.
>	*hex digit*	All notes in the current and following measures will be transposed UP the specified number of semi-tones.
@		Unless otherwise indicated, all notes following are considered to be BASS Clef or -.
_		(Underscore character) All the unsigned notes following are considered to be PERCUSSION Clef.
J	*register.name* *R or S,* *9 hex digits*	Define the named REGISTER with the specified waveform, description, and volume.
K	*digits Ø-7, and* *# or &*	Define the KEY SIGNATURE by number of type specified by the modifier.
M	*name + space*	Define the start of a MEASURE. This also ends any current Measure. Any active Accidentals are dropped. The Key Signature is restored. A trailing SPACE is *required.*
N	*H,Q,I,S,or T*	Define the TIME SIGNATURE. Set note type in the modifier to one beat.
O	*Ø,1,or 2*	Set the OPTIONS to the value specified.
P	*2-digit hex*	Define the beginning of a PART named by the modifier. Any current Measure is ended. Any current Part is ended.
R	*2-digit hex*	Repeat the Part number specified by the modifier. The specified Part must have been previously defined. This symbol group may be followed by a Tempo group and/or Register group. A Measure following a Repeat must be defined by a new Part number.
U	*signed hex*	Transpose all notes following which belong the the current voice up or down the specified number of whole steps.
V	*1,2,3,4,or 5*	All the following notes are to be addressed by the specified Voice number.
Y	*A,B,C,D, or E*	Set the current Voice in the current Part to the Register specified by the modifier.
Z	*hex digit*	Map the stereo channels as specified by the modifier.

Time Value Symbol	Musical Equivalent	Name Of Note	Rest($)
W		WHOLE note	
H		HALF note	
Q		QUARTER note	
I		EIGHTH note 	
S		SIXTEENTH note 	
T		THIRTY-SECOND note 	
X		SIXTY-FOURTH note 	

Time Value Modifier	Musical Equivalent	Name
. (period)		DOTTED note
: (colon)		TRIPLET

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

NOTE SIGNS

DOTTED NOTES — A dot after a note adds 1/2 the value of the preceding note. Examples:

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 kind of notes.

HOLD — Notes sound longer than their actual value.

TIED NOTES — The first of two tied notes must have *no* articulation. The two notes sound like one continuous note. If tied notes cross measure boundaries, and an accidental is applied to the first note, then you must ensure that the accidental is carried through to the next note.

STACCATO — Expression modifiers used to force a note sound separate from another note following it. Use a comma or a semicolon to produce this effect.

ARTICULATION — An expression modifier used to make a note sound clear and distinct, essential when notes are a series of equal values. Use an apostrophe or a quotation mark after the note symbol.

PERFORMANCE SIGNS

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*

Computer News 80
PO Box 680 Casper, WY 82602
(307) 265-6483

INSTALLATION INSTRUCTIONS
FOR THE
SYMPHONY 90 INTERFACE

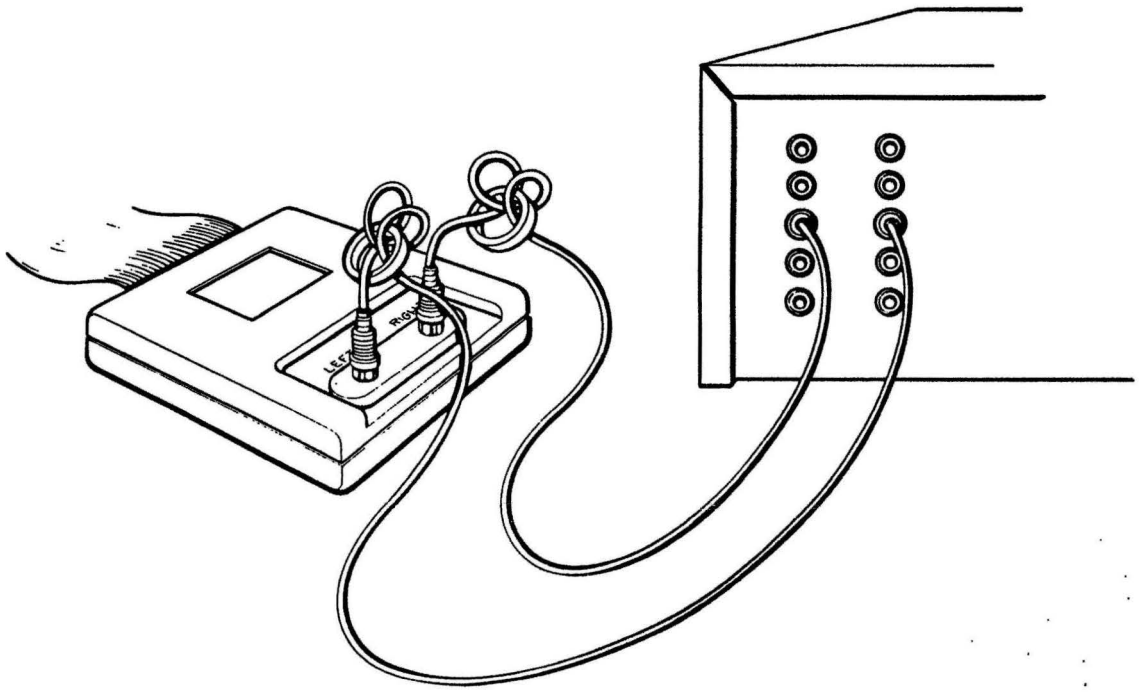
The installation and use of the Symphony 90 Interface is very simple. Just plug the interface into the 50 pin I/O connector that is on the bottom of the computer which is labeled for use with hard drives.

Plug two audio patch cords into the interface and your stereo Aux or input jacks and turn your stereo selector switch to select the input (Tape Recorder or Aux or Input Jacks) to the amplifier.

You must use the amplifier of the stereo to send the sound to the stereo speakers, as there is no amplifier contained in the interface or your computer.

Should you have a hard drive, you can have your Symphony 90 Interface and hard drive connected at the same time. Simply add a 50 pin male edgecard connector to your hard drive cable, or order the adapter cable from CN80. Be sure that your hard drive is connected to the first connector and the interface to the last connector in line.

HAPPY LISTENING



Computer News 80
PO Box 680 Casper, WY 82602
(307) 265-6483

October 2, 1991

Thank you for your order for the SYMPHONY-90.

Do to having to switch to a different manufacturer for the interface unit, we will not be able to ship your order complete at this time.

We expect to be able to fill all orders for the SYMPHONY-90 interface by the last week of October. Your SYMPHONY 90 order will be shipped complete at that time.

If this does not meet with your approval please let us know.

Your patience is appreciated.

Computer News 80
PO Box 680 Casper, WY 82602
(307) 265-6483

October 12, 1991

David Thomas
C/-87 Snell Grove
Oak Park 3046
Australia

Dear David:

Thank you for your letter concerning the Symphony 90 interface. There was no intent on our part to present any misrepresentation, or mislead anyone. Here's exactly what happened (and is happening).

When we first launched the Symphony 90 project, we already knew that we had the right person who would create new and better software to bring the ORCH-90 into the realm of the model 4.

We were contacted in May of 1990 by an individual (when we first solicited for a producer of the interface required to connect the audio systems to the Model 4) who claimed that he could produce the needed interface and we agreed to let him do so.

We also purchased all the left over parts from Tandy that comprised the interface and assembled them to take care of any immediate requirements that users had to play the Orch-90 music collection from the File Cabinet. These were, as you noted, the boards manufactured by Tandy/Radio Shack with the Jon Bokelman design. We offered these units as a separate item in our classified section for some time.

As time went on, and while the software package was being developed for Symphony 90, we received promises of delivery and early reports from the developer of the interface. Naturally we took them at their word, when they said the product would incorporated the newest in chip technology, and even (based upon their claimed prototype construction) twice the sound quality of the early Radio Shack model, or so they said. along with repeated delivery promises for the finished product. This information we happily passed on to our readers through our update articles concerning the Symphony 90 project.

A year and three months later, the Symphony 90 software programs and the Symphony 90 Library Collection was finished and growing, we still did not have a new interface. We got weekly promises of delivery, and of course advise on what the unit would cost so we could include it in our advertising. (We advertised it at $50.00. to cover the cost of manufacture only, as we did not intend to make a profit on it.

What to do! Orders were coming in, and no interface! We were filling early orders with the Tandy original assembled parts. Time and time again we contacted the individual who was supposed to be making the interface, and received promises of only a few more days to have the finished product.

Finally, the first of September he admitted that he had never created anything and had been lying to us all along and could not produce the product as promised, nor would he ever.

Computer News 80
PO Box 680 Casper, WY 82602
(307) 265-6483

We quickly proceeded to work with another manufacturer, and the new (or at least our version of the interface) should be available by the end of October. The problem is that the board will be no better, and no worse than the original board from Tandy, and is costing us now about $70. Remember we advertised it for $50 and have accepted orders for it at that price.

Talk about having egg on your face! But the egg on our faces, turned out to be a slightly odoriferous material having a distinctive brown color, thrown at us by the individual who had strung us along for fifteen months.

The interface that was shipped to you was the last one of the Tandy originals that we had, and we made the choice to send it to you being an overseas shipment, and make the others who had place orders for the system wait. Their being from the States meant that their delivery time would be much less than yours. If we waited till the end of October to ship your order you wouldn't have it for weeks after that.

Should you still feel that we mislead you, we can send you the new board, but the one you received will give you the pleasure of using the Symphony 90 without delay. Or if you already owned an original ORC-90 board, you can return the one we sent you for full credit.

Please remember that with the Composer Package only you can write, edit and play the music you compose, or any of the prewritten music files in the Symphony 90 Library by loading the music files as explained in the manual, but there is no provision to use the music menus that are on the Library Menu disks. To utilize the menus on the Library Menu disks you need the Player System part of the package.

As for the ferrite rings, see enclosed sheet. They are used on your audio patch cord to cut out any distortion signals that the cables may give out, it really is a method of giving the patch cords better shielding.

Thank you again for your support, if you have any questions please let us know.

Sincerely yours,

Stan Slater,
Publisher