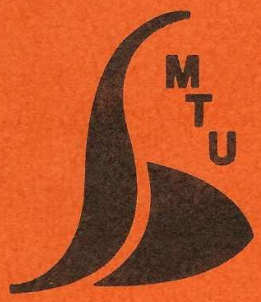

Micro Technology Unlimited

K-1002-1L
8 BIT DIGITAL MUSIC SOFTWARE

SIMPLIFIED INTERPRETER
NOTRAN INTERPRETER
NOTRAN COMPILER

JANUARY 1, 1979

Micro Technology Unlimited
PO Box 4596
841 Galaxy Way
Manchester, NH 03108

RECEIVING, UNPACKING AND CONNECTING THE SYSTEM

If the K-1002 DAC board and/or speaker was ordered along with this software package, the DAC and speaker should be connected to the KIM and the short test routine included in the DAC board documentation run. If no oscilloscope is available to observe the DAC output waveform, listen for a clear robust buzz when the test routine is running. If the waveform is wrong or the buzz seems to have a particularly prominent harmonic, check the connections to the application connector to insure that none of the 8 data bits are scrambled. DO NOT LET THE CASSETTE COME CLOSE TO THE SPEAKER MAGNET OR OTHER MAGNETIC SOURCES.

It is advisable that the program cassette be copied as soon as possible after receipt. Besides providing a backup in case the program cassette becomes jammed or worn, the copy will better match the head alignment in the user's cassette recorder. The copying should be accomplished by loading a file into the KIM and writing it back out onto the copy cassette. If loading problems are experienced, try varying the volume control (too loud is worse than not enough volume) and cassette position in the recorder. In severe cases a friend's recorder could be used to load the programs and make a backup copy. Checksum errors are usually isolated so if it is impossible to read a file, scan through memory comparing with the program listings and correct any errors spotted. If the cassette absolutely refuses to load at all, return it for a replacement.

GENERAL COMMENTS

The simplified 4-voice music program is very similar to the one written up in Byte Magazine September, 1977. The major difference is the inclusion of a "musical subroutine" feature that allows groups of notes to be set aside and played as a group with only a short call code in the song table required for each repetition. The subroutines may be nested allowing complex music to be built from simpler segments. The program is supplied with a song table for Exodus already coded. The user can of course supply his own song table. This program will run in a basic KIM-1 without additional memory, however the size of song tables will be restricted and all 4 voices must have the same waveform.

The Fourier Series program allows the computation of waveforms from harmonic amplitude and phase information. Hand drawn and entered waveforms seldom sound good due to high-order harmonics characteristic of most geometric waveshapes that violate the Nyquist criterion. The Fourier Series program allows precise control over the harmonic content of the waveforms. This program can compute waveforms for use with either the simplified 4 voice program or the NOTRAN interpreter. The Fourier Series program will run on the basic KIM-1 with a full page of memory left over for storage of the computed waveform.

The NOTRAN compiler and interpreter make up a high-level music language system. Besides providing greater ease of use and flexibility in the music played, notes are more compactly stored in memory thus allowing longer songs. Additional memory is required to effectively utilize the compiler and the interpreter.

RUNNING THE DEMONSTRATION SONGS

EXODUS ON THE SIMPLIFIED 4-VOICE PROGRAM

1. Load file 01 into the KIM. This fills the main KIM RAM with the simplified interpreter, a waveform table, and part of the EXODUS song table.
2. Load file 02 into the KIM. This fills the auxilary RAM in the 6530 chips with the reaminder of Exodus.
3. Start execution at 0100. When the song is complete, control is returned to the KIM monitor.
4. Try varying the tempo by changing the byte at TEMPO, 0016.
5. Try writing your own songs following the instructions given in the simplifed 4 voice program listing.

FOURIER SERIES PROGRAM

1. Load file 03 into the KIM. This fills part of the main KIM RAM with the Fourier Series program and a table for a 16 harmonic approximation to a sawtooth wave. Start execution at 0039.
2. The program requires about 10 seconds to compute and scale the waveform specified by the spectrum table after which it enters a loop playing out the waveform through the DAC at a very low frequency. Hit the ST or RESET button to halt the playout and return to the KIM monitor.
3. If an oscilloscope is available, examine the waveform from the DAC. It should approximate a sawtooth with wavelike ripples superimposed. A sync pulse will be present at application connector pin 15 to facilitate scope sync. A 4.7K resistor from this pin to the +5 supply is necessary.
4. If an oscilloscope is not available, examine memory from 0300 to 03FF. Rough plotting of the values read on graph paper with the coordinates labelled in hex should show a wavy sawtooth wave.
5. Try some other spectrum table values. Try changing the phase of the fundamental only and observe the effect on the waveform. Save some of the waveforms on cassette tape by writing out from 0300 to 03FF after a waveform computation. If additional memory is available, the waveforms may be saved in RAM by altering the wave table address at 0013 (low) and 0014 (high).

Try the following harmonic spectrums:

			#1	#2	#3	#4	#5	#6
NHARM	0016		04	05	08	10	10	08
FSRAM	0017	DC	0000	0000	0000	0000	0000	0000
	0019	FUND	FF00	FF00	FF00	FF40	FFC0	2000
	001B	2nd	8000	0000	FF00	0000	0000	3800
	001D	3rd	80C0	C000	0000	5540	5540	5000
	001F	4th	6080	0000	FF00	0000	0000	7000
	0021	5th		8040	0000	3340	3340	FF00
	0023	6th			0000	0000	0000	6000
	0025	7th			0000	2440	2440	3000
	0027	8th			FF00	0000	0000	1000
	0029	9th				1C40	1C40	
	002B	10th				0000	0000	
	002D	11th				1740	1740	
	002F	12th				0000	0000	
	0031	13th				1340	1340	
	0033	14th				0000	0000	
	0035	15th				1140	1140	
	0037	16th				0000	0000	

FOURIER SERIES PROGRAM-con't

6. Play the waveforms saved using the simplified 4-voice music program and Exodus or other song table. What audible effect did changing the fundamental's phase have? If the waves were saved in RAM, try having different voices play different waves. This is accomplished by storing the page number of the desired waveform table at 0002, 0005, 0008 and 000B for voices 1-4 respectively. Notice that high notes and high harmonics can give a very harsh sound. The rule when setting up a spectrum for a voice is that the highest harmonic of the highest note to be played by the voice must not exceed 5kHz. For example, if the highest note played by voice 3 is C5 (522Hz) then the highest harmonic that should be present in its waveform is the nineth. The NOTRAN system allows voice waveforms to be changed in the middle of a song so it is possible to have harmonic content "track" the pitch range of the voice to some extent. Alternatively, voices used primarily for the bass parts may be given a rich harmonic spectrum while those carying the higher pitched melody should be given a more restricted spectrum.

DEMONSTRATION SCORE USING THE NOTRAN INTERPRETER
(requires additional memory located anywhere, 2K minimum)

1. Load file 04 into the KIM. This fills the main KIM memory with the NOTRAN interpreter program.
2. Load file 05 into your memory expansion. This is the NOTRAN object code that was compiled from the demonstration score. If the expanded memory does not include addresses 262F-28A2, choose a convenient load address and follow the procedure in the KIM manual for loading a file at an address different from where it was written.
3. Load file 06 into your memory expansion. This is a set of 4 waveforms used by the demonstration score. Again, this file may be loaded anywhere that is convenient.
4. Using the KIM monitor, store the address of the object code (loaded in step 2) at 0000 (low) and 0001 (high); and the address of the waveform tables (loaded in step 3) at 0002 and 0003.
5. Start execution at 0200. What you will hear will certainly not win any awards but should illustrate the capabilities of the NOTRAN system and some more common music coding techniques. A non-destructive error was left in the score (compiled NOTRAN demonstration listing in the source code) to illustrate the error message format. Because of the infinite loop programmed into the score, the playing must be stopped with the ST or R key.

RUNNING THE NOTRAN MUSIC COMPILER

The NOTRAN music compiler accepts an ASCII string as input (NOTRAN source statements) and produces an ASCII string as output (object hex with source (optionally) interleaved), and object code in memory. The compiler is supplied with I/O routines for use with an ASR-33 teletype (has paper tape reader and punch). The teletype reader should respond to X-ON (DC2 hex 12) and X-OFF (DC1 hex 11) control codes. Using such a TTY, the NOTRAN source would be punched into paper tape using an editor program or even just punched by hand in local mode since the input routine ignores rubouts and honors the backspace control code or backarrow character. The listing would be printed on the TTY. Optionally, the user can supply his own I/O routines for cassette I/O or other methods. See the program listing for instructions regarding user I/O routines.

1. Load file 7 into the KIM. This fills the basic KIM memory with base page variables and default I/O routines. Note that page 3 will be used for I/O buffers and symbol table storage but these areas may be moved elsewhere by modifying the appropriate base page values (see source listing).

2. Load file 8 into the KIM. This is the compiler itself and will load into locations 2000-262E.

3. Check the base page parameters between 0000 and 0016 (see listing) to be certain that the default values assembled into the program are appropriate for your system. They should be fine for systems with 4K or more of expansion memory starting at 2000.

4. If using a teletype, put the NOTRAN source tape into the reader and ready the KIM for teletype I/O. This would already have been done if your are using the KIM monitor in teletype mode.

5. Start the compiler at 2000. An X-ON control will be sent out starting the tape to read. At the end of each line a X-OFF on the tape will stop the reader and the compiled listing of that line will be typed depending on the settings of LIST and ECHO. If user I/O routines are used, input lines will be read and alternated with output lines.

6. When an END statement is compiled a jump to the KIM monitor is taken. The object code may be left in memory and the interpreter loaded to hear the results or the object code may be dumped to tape for later use.

7. If input/output difficulties are experienced, the status of things can be pretty well determined by examining the contents of the input and output line buffers, the buffer pointers, and the current object code location counter.

8. If a listing is not needed the compiler itself is fast enough to keep up with 110 baud continuous input with two stop bits.

9. If the user has sufficient memory and long, complex scores, the defualt length of the symbol table and object code area may be changed. Note that there is an absolute limit of 255 symbols in the symbol table.

PRINCIPLES OF OPERATION

SIMPLIFIED INTERPRETER

(See page of source listing)

The music playing program consists of two major routines: MUSIC and PLAY.

PARAMETER	ADDRESS
MUSIC	0100/0178 (addresses are in hex unless noted)

MUSIC steps through the list of notes in the song table and sets up the duration parameter and the note parameters to be played for each voice for the PLAY routine (see SONG TABLE paragraph for futher DUR description).

DUR	0017
V1IN	000C-000D
V2IN	000E-000F
V3IN	0010-0011
V4IN	0012-0013

The PLAY routine simultaneously plays the four notes specified by V1IN thru V4IN for the time period specified by DUR. Another variable, TEMPO, controls the overall tempo of the music independently of the durations specified in the song table. TEMPO cannot be varied during a song with a simplified interpreter version song table.

PLAY	0179/01CD
TEMPO	0016

The simplified version in this package (different from the BYTE SEPT 77 article) allows 4 different waveform tables to be played simultaneously. These waveform tables require 256 bytes (one memory page) each and are located at WAV1TB thru WAV4TB. The actual waveform samples stored in the table have already been scaled so that when four of them are added up there is no possibility of overflow. The page addresses of the waveform tables are located at the following addresses.

WAVE1TB	0002
WAVE2TB	0005
WAVE3TB	0008
WAVE4TB	000B

The song table can be stored anywhere in memory with the begining address of the table stored at:

SONGA	0014 (low byte) - 0015 (high byte)

The table has an entry for each musical "event" in the piece. A musical event is any song table change of duration or note frequency. By suitable choice of the TEMPO parameter in page 0, "round numbers" (in the binary sense) may be used for duration parameters of common note durations. Each note ID points to a location in the note frequency table which in turn contains a 2 byte frequency parameter for that note which is placed in the V1IN thru V4IN registers. A song table entry requires 5 bytes (reduced to as little as 1 byte for the advanced version) the first of which is the duration parameter.

SIMPLIFIED INTERPRETER - con't

The first byte of a song table event is the duration of the event in units according to the value of TEMPO. For a typical tempo setting of 109 (decimal) a duration of 48 would correspond to a quarter note at approximately 100 beats per minute. If the first byte is 0-4, then a control function is performed as the following:

0 End of song, control is returned to the KIM monitor.
1 End of song table segment, next 2 bytes contain the address of the beginning of the next segment.
2 "REFRAIN DESIGNATOR" (jump to subroutine). The current location in the song table is saved on the stack and the next 2 bytes contain the address of the refrain.
3 End of refrain (return from subroutine). Playing resumes at the lacation in the song table saved by the last refrain designator.
Note: Musical subroutines may be stacked as deep as desired subject only to the amount of stack area available.

The PLAY routine is optimized for speed, because its loop time determines the sample rate. Essentially, the routine maintains four pointers:

V1PT	0000-0001
V2PT	0003-0004
V3PT	0006-0007
V4PT	0009-000A

to the four waveform tables. Each pointer consists of three bytes in order of increasing significance. The first byte is the "fractional part" of the pointer, and the second byte is the integer part which is also the lower half of an address in the waveform table. The third byte is the upper address (WAVXTB, X = 1 to 4) which normally remains constant. Waveform table lookup is considerably simplified by using the indirect addressing mode of the 6502 with these pointers. Note that the fractional part of the pointer is ignored when the table lookup takes place, since interpolation is much too slow for a real time routine.

During each sample, waveform table entries for each voice are fetched, added up, and sent to the digital to analog converter output port. Then the increment (VXIN) is added (double precision) to each pointer (VXPT). Wraparound from the end of a waveform table to the beginning is automatically taken care of due to the fact that the table occupies a full memory page. Finally, the tempo counter is decremented and checked. If the tempo counter is zero, it is restored and the duration counter is decremented and checked. If it is also zero the note is finished and PLAY returns. The net result is that TxD samples are computed and sent out for the event, where T is the tempo parameter and D is the duration parameter.

The system requires an 8 bit Digital to Analog Converter to create the waveform as a varying voltage. The port address on the KIM-1 for this DAC to be connected to is 1700, with the data direction register (DDR at 1701) set to output by storing hex FF.

```
                        .PAGE  'DOCUMENTATION'
                ;       COPYRIGHT 1977 BY MICRO TECHNOLOGY UNLIMITED, BOX 4596,
                ;       MANCHESTER, NEW HAMPSHIRE 03108
                ;       PROGRAM WRITTEN BY HAL CHAMBERLIN

                ;       THIS PROGRAM PLAYS MUSIC IN 4-PART HARMONY ON THE KIM-1 OR
                ;       OTHER 6502 BASED SYSTEM USING AN 8-BIT UNSIGNED
                ;       DIGITAL-TO-ANALOG CONVERTER CONNECTED TO AN OUTPUT PORT.  TUNED
                ;       FOR SYSTEMS WITH A 1 MHZ CRYSTAL CLOCK.  DOES NOT USE THE ROR
                ;       INSTRUCTION.

                ;       SONG TABLE IS AT "SONG"  (HEX 0200)
                ;       ENTRY POINT IS AT "PLAY" (HEX 0100)

                ;       IN THE SONG TABLE EACH MUSICAL EVENT CONSISTS OF 5 BYTES.  THE
                ;       FIRST BYTE IS THE DURATION OF THE EVENT IN UNITS ACCORDING TO
                ;       THE VALUE OF "TEMPO".  FOR A TYPICAL TEMPO SETTING OF 109
                ;       (DECIMAL) A DURATION OF 48 WOULD CORRESPOND TO A QUARTER NOTE
                ;       AT APPROXIMATELY 100 BEATS PER MINUTE.
                ;       IF THE FIRST BYTE IS 0 - 4, THEN A CONTROL FUNCTION IS
                ;       IS PERFORMED AS FOLLOWS:
                ;       0       END OF SONG, CONTROL IS RETURNED TO THE KIM MONITOR.
                ;       1       END OF SONG TABLE SEGMENT, NEXT 2 BYTES CONTAIN ADDRESS
                ;               OF BEGINNING OF NEXT SEGMENT.
                ;       2       "REFRAIN DESIGNATOR" (JUMP TO SUBROUTINE) THE CURRENT
                ;               LOCATION IN THE SONG TABLE IS SAVED ON THE STACK AND THE
                ;               NEXT 2 BYTES CONTAIN THE ADDRESS OF THE REFRAIN.
                ;       3       END OF REFRAIN (RETURN FROM SUBROUTINE) PLAYING RESUMES
                ;               AT THE LOCATION IN THE SONG TABLE SAVED BY THE LAST
                ;               REFRAIN DESIGNATOR.
                ;       NOTE:   MUSICAL SUBROUTINES MAY BE STACKED AS DEEP AS DESIRED
                ;               SUBJECT ONLY TO THE AMOUNT OF STACK AREA AVAILABLE.

                ;       THE NEXT 4 BYTES ARE NOTE ID NUMBERS FOR THE 4 SIMULTANEOUS
                ;       VOICES WITH VOICE 1 FIRST.  SEE THE NOTE FREQUENCY TABLE FOR
                ;       CORRESPONDANCE BETWEEN NUMBERS AND ACTUAL NOTES.

                ;       THE WAVEFORM TABLE CONTAINS A TABULATION OF THE WAVEFORM TO BE
                ;       PLAYED BY A VOICE.  WITH A BASIC KIM-1 (1.1K OF MEMORY) ALL
                ;       VOICES MUST USE THE SAME WAVEFORM.  FOR LARGER MEMORY
                ;       CONFIGURATIONS EACH VOICE MAY USE A DIFFERENT WAVEFORM TABLE.
                ;       WHEN USING A DIFFERENT WAVEFORM TABLE FOR VOICE 1 AS AN
                ;       EXAMPLE, THE PAGE NUMBER OF THE TABLE MUST BE ENTERED AT
                ;       V1PT+2; LIKEWISE AT V2PT+2 FOR VOICE 2, ETC.  THE AMPLITUDE
                ;       OF THE WAVEFORM TABULATED MUST BE SUCH THAT WHEN THE 4 VOICES
                ;       ARE ADDED UP, THERE IS NO POSSIBILITY OF OVERFLOW.  ALSO, THE
                ;       WAVEFORM SAMPLE VALUES ARE UNSIGNED, POSITIVE NUMBERS.  THE NO
                ;       OVERFLOW PROVISION IS SATISFIED IF THE LARGEST SAMPLE VALUE IN
                ;       A WAVEFORM TABLE IS 3F (HEX) OR LESS.  HOWEVER SOME VOICES MAY
                ;       BE MADE LOUDER THAN OTHERS BY ADJUSTING THE WAVEFORM
                ;       AMPLITUDES.  NOTE THAT MAXIMUM PERMISSABLE WAVEFORM AMPLITUDE
                ;       SHOULD ALWAYS BE USED TO MAXIMIZE SIGNAL-TO-NOISE RATIO.
```



```
                        .PAGE  'EQUATES AND BASE PAGE STORAGE'

0000                    .=     0            ; ORG AT PAGE 0 LOCATION 0

1700            DAC     =      X'1700       ; OUTPUT PORT ADDRESS WITH DAC
1701            DACDIR  =      X'1701       ; DATA DRIECTION REGISTER FOR DAC PORT
1780            AUXRAM  =      X'1780       ; ADDRESS OF EXTRA 128 BYTES OF RAM IN 65
1C22            KIMMON  =      X'1C22       ; ENTRY POINT TO KIM KEYBOARD MONITOR

0000 00         V1PT:   .BYTE  0            ; VOICE 1 WAVE POINTER, FRACTIONAL PART
0001 00                 .BYTE  0            ; INTEGER PART
0002 03                 .BYTE  WAV1TB/256   ; WAVEFORM TABLE PAGE ADDRESS FOR VOICE 1
0003 00         V2PT:   .BYTE  0            ; SAME AS ABOVE FOR VOICE 2
0004 00                 .BYTE  0
0005 03                 .BYTE  WAV2TB/256
0006 00         V3PT:   .BYTE  0            ; SAME AS ABOVE FOR VOICE 3
0007 00                 .BYTE  0
0008 03                 .BYTE  WAV3TB/256
0009 00         V4PT:   .BYTE  0            ; SAME AS ABOVE FOR VOICE 4
000A 00                 .BYTE  0
000B 03                 .BYTE  WAV4TB/256

000C 0000       V1IN:   .WORD  0            ; VOICE 1 INCREMENT (FREQUENCY PARAMETER)
000E 0000       V2IN:   .WORD  0            ; VOICE 2
0010 0000       V3IN:   .WORD  0            ; VOICE 3
0012 0000       V4IN:   .WORD  0            ; VOICE 4

0014 0002       SONGA:  .WORD  SONG         ; ADDRESS OF SONG
0016 B6         TEMPO:  .BYTE  182          ; TEMPO CONTROL VALUE, TYPICAL VALUE FOR
                                            ; 4:4 TIME, 60 BEATS PER MINUTE, DURATION
                                            ; BYTE = 48 (10) DESIGNATES A QUARTER NOT

0017 00         DUR:    .BYTE  0            ; DURATION COUNTER
0018 0000       NOTES:  .WORD  0            ; NOTES POINTER
001A 0000       INCPT:  .WORD  0            ; POINTER FOR LOADING UP V1NT - V4NT
001C 0C00       INCA:   .WORD  V1IN         ; INITIAL VALUE OF INCPT
```

KIM4V SIMP KIM-1 4 VOICE
NOTE FREQUENCY TABLE

```
                        .PAGE  'NOTE FREQUENCY TABLE'
 91              ;      NOTE FREQUENCY TABLE FOR 8.772 KHZ SAMPLE RATE
 92              ;      RANGE FROM C2 (65.41 HZ) TO C6 (1046.5 HZ)
 93              ;                                  ID  NOTE  FREQ.    INCR.
 94 001E 0000    FRQTAB: .BYTE  0,0          ;   0  SILENCE
 95 0020 01E9            .BYTE  1,233        ;   2  C2    65.405  1.9089
 96 0022 0206            .BYTE  2,6          ;   4  C2#   69.295  2.0224
 97 0024 0225            .BYTE  2,37         ;   6  D2    73.415  2.1427
 98 0026 0245            .BYTE  2,69         ;   8  D2#   77.783  2.2701
 99 0028 0268            .BYTE  2,104        ;   A  E2    82.408  2.4051
100 002A 028C            .BYTE  2,140        ;   C  F2    87.308  2.5481
101 002C 02B3            .BYTE  2,179        ;   E  F2#   92.498  2.6996
102 002E 02DC            .BYTE  2,220        ;  10  G2    97.998  2.8601
103 0030 0308            .BYTE  3,8          ;  12  G2#  103.83   3.0302
104 0032 0336            .BYTE  3,54         ;  14  A2    110.00  3.2104
105 0034 0367            .BYTE  3,103        ;  16  A2#   116.54  3.4013
106 0036 039A            .BYTE  3,154        ;  18  B2    123.47  3.6035
107 0038 03D1            .BYTE  3,209        ;  1A  C3    130.81  3.8178
108 003A 040B            .BYTE  4,11         ;  1C  C3#   138.59  4.0448
109 003C 0449            .BYTE  4,73         ;  1E  D3    146.83  4.2854
110 003E 048A            .BYTE  4,138        ;  20  D3#   155.57  4.5402
111 0040 04CF            .BYTE  4,207        ;  22  E3    164.82  4.8102
112 0042 0519            .BYTE  5,25         ;  24  F3    174.62  5.0962
113 0044 0566            .BYTE  5,102        ;  26  F3#   185.00  5.3992
114 0046 05B8            .BYTE  5,184        ;  28  G3    196.00  5.7203
115 0048 060F            .BYTE  6,15         ;  2A  G3#   207.65  6.0604
116 004A 066C            .BYTE  6,108        ;  2C  A3    220.00  6.4208
117 004C 06CD            .BYTE  6,205        ;  2E  A3#   233.08  6.8026
118 004E 0735            .BYTE  7,53         ;  30  B3    246.94  7.2071
119 0050 07A3            .BYTE  7,163        ;  32  C4    261.62  7.6356
120 0052 0817            .BYTE  8,23         ;  34  C4#   277.18  8.0897
121 0054 0892            .BYTE  8,146        ;  36  D4    293.66  8.5707
122 0056 0915            .BYTE  9,21         ;  38  D4#   311.13  9.0804
123 0058 099F            .BYTE  9,159        ;  3A  E4    329.63  9.6203
124 005A 0A31            .BYTE  10,49        ;  3C  F4    349.23 10.1924
125 005C 0ACC            .BYTE  10,204       ;  3E  F4#   369.99 10.7984
126 005E 0B71            .BYTE  11,113       ;  40  G4    391.99 11.4405
127 0060 0C1F            .BYTE  12,31        ;  42  G4#   415.30 12.1208
128 0062 0CD7            .BYTE  12,215       ;  44  A4    440.00 12.8416
129 0064 0D9B            .BYTE  13,155       ;  46  A4#   466.16 13.6052
130 0066 0E6A            .BYTE  14,106       ;  48  B4    493.88 14.4142
131 0068 0F45            .BYTE  15,69        ;  4A  C5    523.24 15.2713
132 006A 102E            .BYTE  16,46        ;  4C  C5#   554.36 16.1794
133 006C 1124            .BYTE  17,36        ;  4E  D5    587.32 17.1414
134 006E 1229            .BYTE  18,41        ;  50  D5#   622.26 18.1607
135 0070 133E            .BYTE  19,62        ;  52  E5    659.26 19.2406
136 0072 1462            .BYTE  20,98        ;  54  F5    698.46 20.3847
137 0074 1599            .BYTE  21,153       ;  56  F5#   739.98 21.5969
138 0076 16E2            .BYTE  22,226       ;  58  G5    783.98 22.8811
139 0078 183E            .BYTE  24,62        ;  5A  G5#   830.60 24.2417
140 007A 19AF            .BYTE  25,175       ;  5C  A5    880.00 25.6831
141 007C 1B36            .BYTE  27,54        ;  5E  A5#   932.32 27.2103
142 007E 1CD4            .BYTE  28,212       ;  60  B5    987.76 28.8283
143 0080 1E8B            .BYTE  30,139       ;  62  C6   1046.5  30.5426
144
145 0082        POEND   =      .            ; DEFINE BEGINNING ADDRESS FOR SECOND PART
146                                         ; OF SONG TABLE
```

KIM4V SIMP KIM-1 4 VOICE
SONG TABLE INTERPRETER ROUTINE

```
                        .PAGE  'SONG TABLE INTERPRETER ROUTINE'
148 0082                .=     X'100        ; START PROGRAM CODE AT LOCATION 0100
149
150                     ;      MAIN MUSIC PLAYING PROGRAM
151
152 0100 A9FF   MUSIC:  LDA    #X'FF        ; SET PERIPHERAL A DATA DIRECTION
153 0102 8D0117         STA    DACDIR       ; REGISTER TO OUTPUT
154 0105 D8             CLD                 ; INSURE BINARY ARITHMETIC
155 0106 A514           LDA    SONGA        ; INITIALIZE NOTES POINTER
156 0108 8518           STA    NOTES        ; TO BEGINNING OF SONG
157 010A A515           LDA    SONGA+1
158 010C 8519           STA    NOTES+1
159 010E A000   MUSIC1: LDY    #0           ; SET UP TO TRANSLATE 4 NOTE ID NUMBERS
160 0110 A51C           LDA    INCA         ; INTO FREQUENCY DETERMINING WAVEFORM TABLE
161 0112 851A           STA    INCPT        ; INCREMENTS AND STORE IN V1IN - V4IN
162 0114 B118           LDA    (NOTES),Y    ; GET DURATION FIRST
163 0116 F02D           BEQ    ENDSNG       ; BRANCH IF END OF SONG
164 0118 C902           CMP    #2           ; TEST FOR OTHER CONTROL FUNCTIONS
165 011A F03B           BEQ    RFNCAL       ; BRANCH IF REFRAIN CALL
166 011C 902A           BCC    NXTSEG       ; BRANCH IF END OF TABLE SEGMENT
167 011E C903           CMP    #3
168 0120 F03E           BEQ    RFNRET       ; BRANCH IF RETURN FROM REFRAIN
169 0122 8517           STA    DUR          ; OTHERWISE SAVE DURATION IN DUR
170 0124 207201 MUSIC2: JSR    DINNOT       ; DOUBLE INCREMENT NOTES TO POINT TO THE
171                                         ; NOTE ID OF THE FIRST VOICE
172 0127 B118   MUSIC3: LDA    (NOTES),Y    ; GET A NOTE ID NUMBER
173 0129 AA             TAX                 ; INTO INDEX X
174 012A B51F           LDA    FRQTAB+1,X   ; GET LOW BYTE OF CORRESPONDING FREQUENCY
175 012C 911A           STA    (INCPT),Y    ; STORE INTO LOW BYTE OF VOICE INCREMENT
176 012E E61A           INC    INCPT        ; INDEX TO HIGH BYTE
177 0130 B51E           LDA    FRQTAB,X     ; GET HIGH BYTE OF FREQUENCY
178 0132 911A           STA    (INCPT),Y    ; STORE INTO HIGH BYTE OF VOICE INCREMENT
179 0134 207201         JSR    DINNOT       ; DOUBLE INCREMENT NOTES TO POINT TO THE
180                                         ; NOTE ID OF THE NEXT VOICE
181 0137 E61A           INC    INCPT        ; INDEX TO NEXT VOICE INCREMENT
182 0139 A51A           LDA    INCPT        ; TEST IF 4 VOICE INCREMENTS DONE
183 013B C914           CMP    #V4IN+2
184 013D D0E8           BNE    MUSIC3       ; LOOP IF NOT
185 013F 207901         JSR    PLAY         ; PLAY THIS GROUP OF NOTES
186 0142 4C0E01         JMP    MUSIC1       ; GO LOAD UP NEXT SET OF NOTES
187
188 0145 4C221C ENDSNG: JMP    KIMMON       ; END OF SONG, RETURN TO MONITOR
189
190 0148 C8     NXTSEG: INY                 ; END OF SEGMENT, NEXT TWO BYTES POINT TO
191 0149 B118           LDA    (NOTES),Y    ; BEGINNING OF THE NEXT SEGMENT
192 014B 48             PHA
193 014C C8             INY                 ; GET BOTH SEGMENT ADDRESS BYTES
194 014D B118           LDA    (NOTES),Y
195 014F 8519           STA    NOTES+1      ; THEN STORE IN NOTES POINTER
196 0151 68             PLA
197 0152 8518           STA    NOTES
198 0154 4C0E01         JMP    MUSIC1       ; GO START INTERPRETING NEW SEGMENT
199
200 0157 A518   RFNCAL: LDA    NOTES        ; REFRAIN CALL, SAVE CURRENT VALUE OF NOTES
201 0159 48             PHA                 ; ON THE STACK
```

```
KIM4V  SIMP KIM-1 4 VOICE
SONG TABLE INTERPRETER ROUTINE

  202 015A A519              LDA    NOTES+1
  203 015C 48                PHA
  204 015D 4C4801            JMP    NXTSEG        ; GO INTERPRET NEXT TWO ADDRESS BYTES
  205
  206 0160 68        RFNRET: PLA                  ; REFRAIN RETURN, RESTORE SAVED VALUE OF
  207 0161 8519              STA    NOTES+1       ; NOTES
  208 0163 68                PLA
  209 0164 8518              STA    NOTES
  210 0166 207201            JSR    DINNOT        ; BUMP UP NOTES BY THREE SINCE IT WAS SAVED
  211 0169 207201            JSR    DINNOT        ; UNINCREMENTED
  212 016C 207201            JSR    DINNOT
  213 016F 4C0E01            JMP    MUSIC1        ; GO INTERPRET NEXT EVENT
  214
  215 0172 E618      DINNOT: INC    NOTES         ; DOUBLE INCREMENT NOTES POINTER SUBROUTINE
  216 0174 D002              BNE    DINN1
  217 0176 E619              INC    NOTES+1
  218 0178 60        DINN1:  RTS
  219
```



```
KIM4V  SIMP KIM-1 4 VOICE
SOUND GENERATION SUBROUTINE

                           .PAGE  'SOUND GENERATION SUBROUTINE'
  220
  221                      ;      4 VOICE PLAY SUBROUTINE
  222                      ;      ENTER WITH VARIOUS TABLE POINTERS ALREADY SET UP
  223                      ;      LOOPS TEMPO*DUR TIMES
  224
  225 0179 A000      PLAY:  LDY    #0            ; SET Y TO ZERO FOR STRAIGHT INDIRECT
  226 017B A616             LDX    TEMPO         ; SET X TO TEMPO COUNT
  227                                            ; COMPUTE AND OUTPUT A COMPOSITE SAMPLE
  228 017D 18        PLAY1: CLC                   ; CLEAR CARRY
  229 017E B101             LDA    (V1PT+1),Y    ; ADD UP 4 VOICE SAMPLES
  230 0180 7104             ADC    (V2PT+1),Y    ; USING INDIRECT ADDRESSING THROUGH VOICE
  231 0182 7107             ADC    (V3PT+1),Y    ; POINTERS INTO WAVEFORM TABLES
  232 0184 710A             ADC    (V4PT+1),Y    ; STRAIGHT INDIRECT WHEN Y INDEX = 0
  233 0186 8D0017           STA    X'1700        ; SEND SUM TO DIGITAL-TO-ANALOG CONVERTER
  234 0189 A500             LDA    V1PT          ; ADD INCREMENTS TO POINTERS FOR
  235 018B 650C             ADC    V1IN          ; THE 4 VOICES
  236 018D 8500             STA    V1PT          ; FIRST FRACTIONAL PART
  237 018F A501             LDA    V1PT+1
  238 0191 650D             ADC    V1IN+1
  239 0193 8501             STA    V1PT+1        ; THEN INTEGER PART
  240 0195 A503             LDA    V2PT          ; VOICE 2
  241 0197 650E             ADC    V2IN
  242 0199 8503             STA    V2PT
  243 019B A504             LDA    V2PT+1
  244 019D 650F             ADC    V2IN+1
  245 019F 8504             STA    V2PT+1
  246 01A1 A506             LDA    V3PT          ; VOICE 3
  247 01A3 6510             ADC    V3IN
  248 01A5 8506             STA    V3PT
  249 01A7 A507             LDA    V3PT+1
  250 01A9 6511             ADC    V3IN+1
  251 01AB 8507             STA    V3PT+1
  252 01AD A509             LDA    V4PT          ; VOICE 4
  253 01AF 6512             ADC    V4IN
  254 01B1 8509             STA    V4PT
  255 01B3 A50A             LDA    V4PT+1
  256 01B5 6513             ADC    V4IN+1
  257 01B7 850A             STA    V4PT+1
  258 01B9 CA               DEX                  ; DECREMENT & CHECK TEMPO COUNT
  259 01BA D008             BNE    TIMWAS        ; BRANCH TO TIME WASTE IF NOT RUN OUT
  260 01BC C617             DEC    DUR           ; DECREMENT & CHECK DURATION COUNTER
  261 01BE F00C             BEQ    ENDNOT        ; JUMP OUT IF END OF NOTE
  262 01C0 A616             LDX    TEMPO         ; RESTORE TEMPO COUNT
  263 01C2 D0B9             BNE    PLAY1         ; CONTINUE PLAYING
  264 01C4 D000      TIMWAS: BNE    TMWS1         ; 3  WASTE 12 STATES
  265 01C6 D000      TMWS1:  BNE    TMWS2         ; 3
  266 01C8 D000      TMWS2:  BNE    TMWS3         ; 3
  267 01CA D0B1      TMWS3:  BNE    PLAY1         ; 3  CONTINUE PLAYING
  268 01CC 60        ENDNOT: RTS                  ; RETURN
  269                                            ; TOTAL LOOP TIME = 114 STATES = 8770 HZ
  270
  271 01CD           P1END   =      .             ; DEFINE BEGINNING ADDRESS FOR THIRD PART
  272                                            ; OF SONG TABLE
  273
```

```
                                .PAGE  'SONG TABLE'
  274                   ;       SONG TABLE
  275
  276 01CD                      .=     X'200        ; START SONG AT 0200
  277
  278                   ;       SONG TABLE FOR "EXODUS"
  279                   ;       DURATION COUNT = 48 FOR QUARTER NOTE
  280
  281                                                ; DUR  V1   V2   V3   V4
  282 0200 02           SONG:   .BYTE  2             ; CALL THE MUSICAL SUBROUTINE
  283 0201 8200                 .WORD  SUB1
  284 0203 0C22323A             .BYTE  X'0C,X'22,X'32,X'3A,X'3C  ; 1/16 E3   C4   E4   F4
  285 0208 1828323A             .BYTE  X'18,X'28,X'32,X'3A,X'40  ; 1/8  G3   C4   E4   G4
  286 020D 1828323A             .BYTE  X'18,X'28,X'32,X'3A,X'44  ; 1/8  G3   C4   E4   A4
  287 0212 241E2C36             .BYTE  X'24,X'1E,X'2C,X'36,X'3C  ; 1/8. D3   A3   D4   F4
  288 0217 0C1E2C00             .BYTE  X'0C,X'1E,X'2C,X'00,X'36  ; 1/16 D3   A3        D4
  289 021C 30222C34             .BYTE  X'30,X'22,X'2C,X'34,X'3A  ; 1/4  E3   A3   C4#  E4
  290 0221 90142C34             .BYTE  X'90,X'14,X'2C,X'34,X'3A  ; 1/2. A2   A3   C4#  E4
  291 0226 02                   .BYTE  2             ; CALL THE SUBROUTINE AGAIN
  292 0227 8200                 .WORD  SUB1
  293 0229 0C22323A             .BYTE  X'0C,X'22,X'32,X'3A,X'40  ; 1/16 E3   C4   E4   G4
  294 022E 18242E36             .BYTE  X'18,X'24,X'2E,X'36,X'3C  ; 1/8  F3   B30  D4   F4
  295 0233 18242E36             .BYTE  X'18,X'24,X'2E,X'36,X'40  ; 1/8  F3   B30  D4   G4
  296 0238 241A2832             .BYTE  X'24,X'1A,X'28,X'32,X'3A  ; 1/8. C3   G3   C4   E4
  297 023D 0C000000             .BYTE  X'0C,X'00,X'00,X'00,X'4A  ; 1/16                  C5
  298 0242 18063644             .BYTE  X'18,X'06,X'36,X'44,X'4E  ; 1/8  D2   D4   A4   D5
  299 0247 18143644             .BYTE  X'18,X'14,X'36,X'44,X'4E  ; 1/8  A2   D4   A4   D5
  300 024C 181E3644             .BYTE  X'18,X'1E,X'36,X'44,X'4E  ; 1/8  D3   D4   A4   D5
  301 0251 18223644             .BYTE  X'18,X'22,X'36,X'44,X'4E  ; 1/8  E3   D4   A4   D5
  302 0256 30243644             .BYTE  X'30,X'24,X'36,X'44,X'4E  ; 1/4  F3   D4   A4   D5
  303 025B 30243644             .BYTE  X'30,X'24,X'36,X'44,X'4E  ; 1/4  F3   D4   A4   D5
  304 0260 0C323A44             .BYTE  X'0C,X'32,X'3A,X'44,X'4A  ; 1/16 C4   E4   A4   C5
  305 0265 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'4E  ; 1/16 C4   E4        D5
  306 026A 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'4A  ; 1/16 C4   E4        C5
  307 026F 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'44  ; 1/16 C4   E4        A4
  308 0274 30283240             .BYTE  X'30,X'28,X'32,X'40,X'4A  ; 1/4  G3   C4   G4   C5
  309 0279 30143240             .BYTE  X'30,X'14,X'32,X'40,X'4A  ; 1/4  A2   C4   G4   C5
  310 027E 182C3240             .BYTE  X'18,X'2C,X'32,X'40,X'4A  ; 1/8  A3   C4   G4   C5
  311 0283 182C3240             .BYTE  X'18,X'2C,X'32,X'40,X'44  ; 1/8  A3   C4   G4   A4
  312 0288 18263644             .BYTE  X'18,X'26,X'36,X'44,X'4E  ; 1/8  F3#  D4   A4   D5
  313 028D 18263A00             .BYTE  X'18,X'26,X'3A,X'00,X'52  ; 1/8  F3#  E4        E5
  314 0292 4824404E             .BYTE  X'48,X'24,X'40,X'4E,X'58  ; 3/8  F3   G4   D5   G5
  315 0297 18243C00             .BYTE  X'18,X'24,X'3C,X'00,X'54  ; 1/8  F3   F4        F5
  316 029C 18243A00             .BYTE  X'18,X'24,X'3A,X'00,X'52  ; 1/8  F3   E4        E5
  317 02A1 18243600             .BYTE  X'18,X'24,X'36,X'00,X'4E  ; 1/8  F3   D4        D5
  318 02A6 0C323A44             .BYTE  X'0C,X'32,X'3A,X'44,X'4A  ; 1/16 C4   E4   A4   C5
  319 02AB 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'4E  ; 1/16 C4   E4        D5
  320 02B0 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'4A  ; 1/16 C4   E4        C5
  321 02B5 0C323A00             .BYTE  X'0C,X'32,X'3A,X'00,X'44  ; 1/16 C4   E4        A4
  322 02BA 30283240             .BYTE  X'30,X'28,X'32,X'40,X'4A  ; 1/4  G3   C4   G4   C5
  323 02BF 30143240             .BYTE  X'30,X'14,X'32,X'40,X'4A  ; 1/4  A2   C4   G4   C5
  324 02C4 182C3240             .BYTE  X'18,X'2C,X'32,X'40,X'4A  ; 1/8  A3   C4   G4   C5
  325 02C9 182C3240             .BYTE  X'18,X'2C,X'32,X'40,X'44  ; 1/8  A3   C4   G4   A4
  326 02CE 182C3E44             .BYTE  X'18,X'2C,X'3E,X'44,X'4E  ; 1/8  A3   F4#  A4   D5
  327 02D3 182C3E44             .BYTE  X'18,X'2C,X'3E,X'44,X'52  ; 1/8  A3   F4#  A4   E5
```

KIM4V SIMP KIM-1 4 VOICE
SONG TABLE

```
  328 02D8 483C4452             .BYTE  X'48,X'3C,X'44,X'52,X'5C  ; 1/4. F4   A4   E5   A5
  329 02DD 18404400             .BYTE  X'18,X'40,X'44,X'00,X'58  ; 1/8  G4   A4        G5
  330 02E2 243A4400             .BYTE  X'24,X'3A,X'44,X'00,X'52  ; 1/8. E4   A4        E5
  331 02E7 0C364400             .BYTE  X'0C,X'36,X'44,X'00,X'4E  ; 1/16 D4   A4        D5
  332 02EC 302C444C             .BYTE  X'30,X'2C,X'44,X'4C,X'52  ; 1/4  A3   A4   C5#  E5
  333 02F1 6014444C             .BYTE  X'60,X'14,X'44,X'4C,X'52  ; 1/2. A2   A4   C5#  E5
  334 02F6 00                   .BYTE  0             ; END OF PIECE
  335                   ;       THIS IS A "MUSICAL SUBROUTINE" WHICH CONTAINS A PASSAGE THAT IS
  336                   ;       USED TWICE IN THE MAINLINE "EXODUS"
  337
  338 02F7                      .=     POEND         ; START SUBROUTINE IN
  339                                                ; REMAINDER OF PAGE 0
  340
  341                                                ; DUR  V1   V2   V3   V4
  342 0082 1C002C00     SUB1:   .BYTE  X'1C,X'00,X'2C,X'00,X'00  ; 1/8       A3
  343 0087 4814242C             .BYTE  X'48,X'14,X'24,X'2C,X'36  ; 1/4. A2   F3   A3   D4
  344 008C 18000000             .BYTE  X'18,X'00,X'00,X'00,X'44  ; 1/8                  A4
  345 0091 481E3036             .BYTE  X'48,X'1E,X'30,X'36,X'40  ; 1/4. D3   B3   D4   G4
  346 0096 18003600             .BYTE  X'18,X'00,X'36,X'00,X'00  ; 1/8       D4
  347 009B 18242E36             .BYTE  X'18,X'24,X'2E,X'36,X'3C  ; 1/8  F3   B30  D4   F4
  348 00A0 18000000             .BYTE  X'18,X'00,X'00,X'00,X'40  ; 1/8                  G4
  349 00A5 241A2832             .BYTE  X'24,X'1A,X'28,X'32,X'3A  ; 1/8. C3   G3   C4   E4
  350 00AA 0C000000             .BYTE  X'0C,X'00,X'00,X'00,X'32  ; 1/16                 C4
  351 00AF 481E242C             .BYTE  X'48,X'1E,X'24,X'2C,X'36  ; 1/4. D3   F3   A3   D4
  352 00B4 18000000             .BYTE  X'18,X'00,X'00,X'00,X'44  ; 1/8                  A4
  353 00B9 1814323A             .BYTE  X'18,X'14,X'32,X'3A,X'4A  ; 1/8  A2   C4   E4   C5
  354 00BE 1822323A             .BYTE  X'18,X'22,X'32,X'3A,X'4A  ; 1/8  E3   C4   E4   C5
  355 00C3 182C323A             .BYTE  X'18,X'2C,X'32,X'3A,X'4A  ; 1/8  A3   C4   E4   C5
  356 00C8 1800323A             .BYTE  X'18,X'00,X'32,X'3A,X'48  ; 1/8       C4   E4   B4
  357 00CD 182C323C             .BYTE  X'18,X'2C,X'32,X'3C,X'4A  ; 1/8  A3   C4   F4   C5
  358 00D2 182C323C             .BYTE  X'18,X'2C,X'32,X'3C,X'4E  ; 1/8  A3   C4   F4   D5
  359 00D7 24283036             .BYTE  X'24,X'28,X'30,X'36,X'48  ; 1/8. G3   B3   D4   B4
  360 00DC 0C283036             .BYTE  X'0C,X'28,X'30,X'36,X'40  ; 1/16 G3   B3   D4   G4
  361 00E1 3022343A             .BYTE  X'30,X'22,X'34,X'3A,X'44  ; 1/4  E3   C4#  E4   A4
  362 00E6 01                   .BYTE  1             ; DEFINE BEGINNING OF NEXT
  363 00E7 CD01                 .WORD  P1END         ; SEGMENT AT END OF PAGE 1
  364 00E9                      .=     P1END
  365 01CD 3014343A             .BYTE  X'30,X'14,X'34,X'3A,X'44  ; 1/4  A2   C4#  E4   A4
  366 01D2 18140000             .BYTE  X'18,X'14,X'00,X'00,X'00  ; 1/8  A2
  367 01D7 18144400             .BYTE  X'18,X'14,X'44,X'00,X'00  ; 1/8  A2   A4
  368 01DC 1822323A             .BYTE  X'18,X'22,X'32,X'3A,X'4A  ; 1/8  E3   C4   E4   C5
  369 01E1 01                   .BYTE  1             ; DEFINE BEGINNING OF NEXT
  370 01E2 8017                 .WORD  AUXRAM        ; SEGMENT IN 6530 RAM
  371 01E4                      .=     AUXRAM
  372 1780 1822323A             .BYTE  X'18,X'22,X'32,X'3A,X'52  ; 1/8  E3   C4   E4   E5
  373 1785 241E362E             .BYTE  X'24,X'1E,X'36,X'2E,X'4E  ; 1/8. D3   D4   F4#  D5
  374 178A 0C1E362E             .BYTE  X'0C,X'1E,X'36,X'2E,X'44  ; 1/16 D3   D4   F4#  A4
  375 178F 181E362E             .BYTE  X'18,X'1E,X'36,X'2E,X'44  ; 1/8  D3   D4   F4#  A4
  376 1794 181E2C2E             .BYTE  X'18,X'1E,X'2C,X'2E,X'44  ; 1/8  D3   A3   F4#  A4
  377 1799 181E2C2E             .BYTE  X'18,X'1E,X'2C,X'2E,X'44  ; 1/8  D3   A3   F4#  A4
  378 179E 181E2C2E             .BYTE  X'18,X'1E,X'2C,X'2E,X'4E  ; 1/8  D3   A3   F4#  D5
  379 17A3 181E362C             .BYTE  X'18,X'1E,X'36,X'2C,X'44  ; 1/8  D3   D4   F4   A4
  380 17A8 181E362C             .BYTE  X'18,X'1E,X'36,X'2C,X'4E  ; 1/8  D3   D4   F4   D5
  381 17AD 2422323A             .BYTE  X'24,X'22,X'32,X'3A,X'52  ; 1/8. E3   C4   E4   E5
  382 17B2 0C22323A             .BYTE  X'0C,X'22,X'32,X'3A,X'44  ; 1/16 E3   C4   E4   A4
```

```
KIM4V  SIMP KIM-1 4 VOICE
SONG TABLE

  383 17B7 1822323A          .BYTE  X'18,X'22,X'32,X'3A,X'44  ; 1/8  E3   C4   E4   A4
  384 17BC 18222C3A          .BYTE  X'18,X'22,X'2C,X'3A,X'44  ; 1/8  E3   A3   E4   A4
  385 17C1 30222C3A          .BYTE  X'30,X'22,X'2C,X'3A,X'44  ; 1/4  E3   A3   E4   A4
  386 17C6 181A323A          .BYTE  X'18,X'1A,X'32,X'3A,X'40  ; 1/8  C3   C4   E4   G4
  387 17CB 181A323A          .BYTE  X'18,X'1A,X'32,X'3A,X'44  ; 1/8  C3   C4   E4   A4
  388 17D0 181E3640          .BYTE  X'18,X'1E,X'36,X'40,X'46  ; 1/8  D3   D4   G4   B4@
  389 17D5 181E3640          .BYTE  X'18,X'1E,X'36,X'40,X'4A  ; 1/8  D3   D4   G4   C5
  390 17DA 2422323A          .BYTE  X'24,X'22,X'32,X'3A,X'44  ; 1/8. E3   C4   E4   A4
  391 17DF 03                .BYTE  3                         ; RETURN
  392
```

```
KIM4V  SIMP KIM-1 4 VOICE
WAVEFORM TABLE

                           .PAGE  'WAVEFORM TABLE'
  393                ;     WAVEFORM TABLE
  394                ;     EXACTLY ONE PAGE LONG ON A PAGE BOUNDARY
  395                ;     MAXIMUM VALUE OF AN ENTRY IS 63 DECIMAL OR 3F HEX TO AVOID
  396                ;     OVERFLOW WHEN 4 VOICES ARE ADDED UP
  397
  398 17E0                  .=     X'300        ; START WAVEFORM TABLE AT 0300
  399
  400 0300           WAV1TB  =      .            ; VOICE 1 WAVEFORM TABLE
  401 0300           WAV2TB  =      .            ; VOICE 2 WAVEFORM TABLE
  402 0300           WAV3TB  =      .            ; VOICE 3 WAVEFORM TABLE
  403 0300           WAV4TB  =      .            ; VOICE 4 WAVEFORM TABLE
  404                                            ; NOTE THAT ALL 4 VOICES USE THIS TABLE DUE
  405                                            ; TO LACK OF RAM IN BASIC KIM-1
  406
  407                ;       FUNDAMENTAL AMPLITUDE  1.0 (REFERENCE)
  408                ;       SECOND HARMONIC .5, IN PHASE WITH FUNDAMENTAL
  409                ;       THIRD HARMONIC .5, 90 DEGREES LEADING PHASE
  410
  411 0300 33343536  WAVXTB: .BYTE  X'33,X'34,X'35,X'36,X'36,X'37,X'38,X'39
  412 0308 393A3A3B          .BYTE  X'39,X'3A,X'3A,X'3B,X'3B,X'3B,X'3C,X'3C
  413 0310 3C3C3C3C          .BYTE  X'3C,X'3C,X'3C,X'3C,X'3C,X'3C,X'3C,X'3C
  414 0318 3C3C3C3B          .BYTE  X'3C,X'3C,X'3C,X'3B,X'3B,X'3B,X'3B,X'3B
  415 0320 3A3A3A3A          .BYTE  X'3A,X'3A,X'3A,X'3A,X'3A,X'3A,X'39,X'39
  416 0328 39393939          .BYTE  X'39,X'39,X'39,X'39,X'39,X'39,X'39,X'39
  417 0330 3A3A3A3A          .BYTE  X'3A,X'3A,X'3A,X'3A,X'3A,X'3B,X'3B,X'3B
  418 0338 3B3C3C3C          .BYTE  X'3B,X'3C,X'3C,X'3C,X'3D,X'3D,X'3D,X'3D
  419 0340 3E3E3E3E          .BYTE  X'3E,X'3E,X'3E,X'3E,X'3F,X'3F,X'3F,X'3F
  420 0348 3F3F3F3F          .BYTE  X'3F,X'3F,X'3F,X'3F,X'3F,X'3F,X'3F,X'3F
  421 0350 3E3E3E3D          .BYTE  X'3E,X'3E,X'3E,X'3D,X'3D,X'3C,X'3C,X'3B
  422 0358 3B3A3938          .BYTE  X'3B,X'3A,X'39,X'38,X'38,X'37,X'36,X'35
  423 0360 34333231          .BYTE  X'34,X'33,X'32,X'31,X'30,X'2F,X'2E,X'2D
  424 0368 2C2B2A29          .BYTE  X'2C,X'2B,X'2A,X'29,X'28,X'27,X'26,X'25
  425 0370 24232221          .BYTE  X'24,X'23,X'22,X'21,X'21,X'20,X'1F,X'1F
  426 0378 1E1E1D1D          .BYTE  X'1E,X'1E,X'1D,X'1D,X'1D,X'1D,X'1C,X'1C
  427 0380 1C1C1D1D          .BYTE  X'1C,X'1C,X'1D,X'1D,X'1D,X'1D,X'1D,X'1E
  428 0388 1E1F1F20          .BYTE  X'1E,X'1F,X'1F,X'20,X'20,X'21,X'21,X'22
  429 0390 23232424          .BYTE  X'23,X'23,X'24,X'24,X'25,X'26,X'26,X'27
  430 0398 28282929          .BYTE  X'28,X'28,X'29,X'29,X'29,X'2A,X'2A,X'2B
  431 03A0 2B2B2B2B          .BYTE  X'2B,X'2B,X'2B,X'2B,X'2B,X'2B,X'2B,X'2A
  432 03A8 2A2A2929          .BYTE  X'2A,X'2A,X'29,X'29,X'28,X'27,X'27,X'26
  433 03B0 25242322          .BYTE  X'25,X'24,X'23,X'22,X'21,X'20,X'1F,X'1D
  434 03B8 1C1B1918          .BYTE  X'1C,X'1B,X'19,X'18,X'17,X'15,X'14,X'13
  435 03C0 11100F0D          .BYTE  X'11,X'10,X'0F,X'0D,X'0C,X'0B,X'09,X'08
  436 03C8 07060504          .BYTE  X'07,X'06,X'05,X'04,X'03,X'03,X'02,X'01
  437 03D0 01000000          .BYTE  X'01,X'00,X'00,X'00,X'00,X'00,X'00,X'00
  438 03D8 00000101          .BYTE  X'00,X'00,X'01,X'01,X'01,X'02,X'03,X'04
  439 03E0 05060708          .BYTE  X'05,X'06,X'07,X'08,X'09,X'0B,X'0C,X'0D
  440 03E8 0F101213          .BYTE  X'0F,X'10,X'12,X'13,X'15,X'16,X'18,X'1A
  441 03F0 1B1D1F20          .BYTE  X'1B,X'1D,X'1F,X'20,X'22,X'23,X'25,X'27
  442 03F8 282A2B2C          .BYTE  X'28,X'2A,X'2B,X'2C,X'2E,X'2F,X'30,X'31
  443
  444 0000                  .END
NO ERROR LINES
```

KIMFS KIM FOURIER SERIES
DOCUMENTATION

```
                        .PAGE  'DOCUMENTATION'
                ;       COPYRIGHT 1977 BY MICRO TECHNOLOGY UNLIMITED, BOX 4596,
                ;       MANCHESTER, NEW HAMPSHIRE 03108
                ;       PROGRAM WRITTEN BY HAL CHAMBERLIN

                ;       FOURIER SERIES EVALUATION PROGRAM FOR COMPUTING WAVEFORM TABLE
                ;       CONTENTS FROM HARMONIC SPECIFICATIONS.

                ;       ENTER AT SCALE TO COMPUTE ONE CYCLE OF A WAVEFORM SUITABLE FOR
                ;       USE WITH EITHER THE SIMPLIFIED 4 VOICE MUSIC INTERPRETER, THE
                ;       ADVANCED 4 VOICE MUSIC INTERPRETER, OR THE SIMPLIFIED SINGLE
                ;       VOICE MUSIC INTERPRETER WITH EXPRESSION.  ENTER AT WAVE TO
                ;       BYPASS SCALE FACTOR COMPUTATION (USER MUST SUPPLY SCALE FACTOR)

                ;       THE "FOURIER SERIES SPECTRUM PARAMETERS" MUST BE SET TO
                ;       APPROPRIATE VALUES BEFORE EXECUTING THE PROGRAM.

                ;       WAVEAD   ADDRESS OF WAVEFORM TABLE TO FILL.  THE ROUTINE
                ;                COMPUTES 256 SAMPLES ON ONE CYCLE OF THE WAVEFORM AND
                ;                PLACES THEM IN MEMORY STARTING AT THE CONTENTS OF
                ;                WAVEAD.
                ;       PKAMP    THE WAVEFORM RESULTING FROM THE FOURIER SERIES
                ;                EVALUATION IS NORMALIZED SUCH THAT THE NEGATIVE PEAK
                ;                IS EQUAL TO ZERO AND THE POSITIVE PEAK IS EQUAL TO
                ;                PKAMP.  NORMALIZATION CAN BE BYPASSED FOR NON-AUDIO
                ;                APPLICATIONS.
                ;       NHARM    SET EQUAL TO THE HIGHEST HARMONIC TO GENERATE.  THE
                ;                ROUTINE WILL IGNORE FSRAM ENTRIES CORRESPONDING TO
                ;                HARMONICS HIGHER THAN NHARM AND THUS WILL RUN FASTER.
                ;       FSRAM    A TABLE OF HARMONIC AMPLITUDES AND PHASES.  EACH
                ;                HARMONIC IS REPRESENTED BY A PAIR OF BYTES.  THE FIRST
                ;                BYTE OF THE PAIR IS THE AMPLITUDE OF THE CORRESPONDING
                ;                HARMONIC.  THIS IS AN UNSIGNED BINARY FRACTION; X'FF
                ;                IS MAXIMUM (.996), X'80 = .5, X'40 = .25, ETC.  THE
                ;                SECOND BYTE IS THE PHASE ANGLE OF THE CORRESPONDING
                ;                HARMONIC EXPRESSED AS AN UNSIGNED BINARY FRACTION
                ;                MULTIPLIED BY 2*PI RADIANS; 0 = NO PHASE SHIFT (COSINE
                ;                WAVE), X'40 = PI/2 RADIAN SHIFT = 90 DEGREES, ETC.
                ;                THE FIRST BYTE PAIR CORRESPONDS TO THE ZEROTH HARMONIC
                ;                (DC COMPONENT), NEXT PAIR TO THE FUNDAMENTAL, NEXT TO
                ;                THE SECOND HARMONIC, ETC.  NOTE THAT THE NORMALIZATION
                ;                PROCESS DESTROYS ANY EFFECT OF THE DC COMPONENT ON THE
                ;                RESULTING WAVEFORM.  IN NON-AUDIO APPLICATIONS
                ;                NORMALIZATION MAY BE BYPASSED AND THUS THE DC
                ;                COMPONENT WILL BE EFFECTIVE.  THE PHASE OF THE DC
                ;                COMPONENT SHOULD BE ZERO FOR THE EXPECTED RESULTS.

                ;       NOTE THAT WHEN USING THIS PROGRAM TO COMPUTE WAVEFORMS FOR THE
                ;       MUSIC PROGRAMS THAT THE HIGHEST NON-ZERO HARMONIC OF THE
                ;       HIGHEST NOTE PLAYED BY A VOICE USING THE WAVEFORM SHOULD NOT BE
                ;       GREATER THAN 4.5 TO 5.0 KHZ.  IF THIS CONDITION IS NOT MET, THE
                ;       UPPER HARMONICS WILL ALIAS AND CREATE A VERY HARSH SOUND.  IN
                ;       SYSTEMS WITH ADEQUATE MEMORY FOR SEPARATE WAVEFORM TABLES, THE
                ;       LOWER REGISTER VOICES MAY BE SEPARATED FROM THE HIGHER REGISTER
                ;       ONES AND GIVEN A RICHER HARMONIC SPECTRUM.

                ;       NOTE THAT THIS PROGRAM CONTAINS SOME GENERALLY USEFUL
                ;       ARITHMETIC SUBROUTINES FOR DOUBLE PRECISION MULTIPLY AND
                ;       DIVIDE.
```

KIMFS KIM FOURIER SERIES
CONSTANTS AND DATA

```
                        .PAGE  'CONSTANTS AND DATA'
0000                    .=     0            ; KEEP ALL CONSTANTS AND DATA IN PAGE 0

1C22            KIMMON  =      X'1C22       ; ENTRY POINT TO KIM KEYBOARD MONITOR
0300            WAVETB  =      X'0300       ; ADDRESS OF WAVEFORM TABLE IN SIMPLIFIED
                                            ; 4 VOICE MUSIC PROGRAM
1700            DAC     =      X'1700       ; OUTPUT PORT ADDRESS WITH DAC
1701            DACDIR  =      X'1701       ; DATA DRIECTION REGISTER FOR DAC PORT
1702            PTB     =      X'1702       ; PORT B ON APPLICATION CONNECTOR
1703            PTBDIR  =      X'1703       ; DATA DIRECTION REGISTER FOR PORT B

                ;      STORAGE FOR THE ARITHMETIC SUBROUTINES

0000 00000000   PROD:  .WORD  0,0          ; PRODUCT/DIVIDEND FOR ARITHMETIC ROUTINES
0004 0000       MPCD:  .WORD  0            ; MUPTIPLICAND/DIVISOR FOR ARITHMETIC
0006 0000       MPSAVE: .WORD 0            ; TEMPORARY STORAGE FOR MULTIPLY
0000            DVND   =      PROD         ; EQUATES FOR DIVISON ROUTINES
0004            DVSR   =      MPCD

                ;      STORAGE FOR THE FOURIER SERIES EVALUATION

0008 0000       HRMACC: .WORD 0            ; HARMONIC ACCUMULATOR
000A 00         PNTNO:  .BYTE 0            ; POINT NUMBER WITHIN CYCLE OF WAVE
000B 00         NDXACC: .BYTE 0            ; INDEXING ACCUMULATOR
000C 00         HRMCNT: .BYTE 0            ; HARMONIC COUNTER
000D 0000       MAX:    .WORD 0            ; MAXIMUM WAVEFORM AMPLITUDE
000F 0000       MIN:    .WORD 0            ; MINIMUM WAVEFORM AMPLITUDE
0011 0000       SCALEF: .WORD 0            ; SCALE FACTOR FOR AMPLITUDE NORMALIZATION

                ;      FOURIER SERIES SPECTRUM PARAMETERS

0013 0003       WAVEAD: .WORD  WAVETB      ; ADDRESS OF WAVEFORM TABLE TO FILL
0015 3F         PKAMP:  .BYTE  X'3F        ; DESIRED PEAK AMPLITUDE OF WAVEFORM
0016 10         NHARM:  .BYTE  16          ; HIGHEST HARMONIC TO GENERATE (16 IS MAX)
                                           ; EXAMPLE SPECTRUM OF A SAWTOOTH WAVE
                                           ; ALTER AFTER LOADING FOR OTHER WAVEFORMS
0017 0000       FSRAM:  .BYTE  0,0         ; DC COMPONENT
0019 FF40               .BYTE  255,X'40    ; FUNDAMENTAL
001B 7F40               .BYTE  255/2,X'40  ; 2ND HARMONIC
001D 5540               .BYTE  255/3,X'40  ; 3RD HARMONIC
001F 3F40               .BYTE  255/4,X'40  ; 4TH HARMONIC
0021 3340               .BYTE  255/5,X'40  ; 5TH HARMONIC
0023 2A40               .BYTE  255/6,X'40  ; 6TH HARMONIC
0025 2440               .BYTE  255/7,X'40  ; 7TH HARMONIC
0027 1F40               .BYTE  255/8,X'40  ; 8TH HARMONIC
0029 1C40               .BYTE  255/9,X'40  ; 9TH HARMONIC
002B 1940               .BYTE  255/10,X'40 ; 10TH HARMONIC
002D 1740               .BYTE  255/11,X'40 ; 11TH HARMONIC
002F 1540               .BYTE  255/12,X'40 ; 12TH HARMONIC
0031 1340               .BYTE  255/13,X'40 ; 13TH HARMONIC
0033 1240               .BYTE  255/14,X'40 ; 14TH HARMONIC
0035 1140               .BYTE  255/15,X'40 ; 15TH HARMONIC
0037 0F40               .BYTE  255/16,X'40 ; 16TH HARMONIC
```

```
                        .PAGE  'MAIN WAVEFORM COMPUTATION ROUTINE'
                ;       THIS PROGRAM FIRST HAS A FULL CYCLE OF THE WAVEFORM COMPUTED IN
                ;       ORDER TO DETERMINE THE POSITIVE AND NEGATIVE PEAK VALUES.
                ;       THEN IT COMPUTES THE WAVEFORM AGAIN SAMPLE-BY-SAMPLE AND
                ;       NORMALIZES THE AMPLITUDE TO BE BETWEEN 0 AND "PKAMP".
                ;       EACH NORMALIZED SAMPLE IS THEN PLACED INTO A TABLE STARTING AT
                ;       THE ADDRESS IN WAVEAD.  FINALLY, THE TABLE'S CONTENTS ARE
                ;       SCANNED AND SENT TO THE DIGITAL-TO-ANALOG CONVERTER FOR
                ;       MONITORING.

                ;       SCALE FACTOR DETERMINATION ROUTINE
                ;       THIS ROUTINE GOES THROUGH ONE CYCLE OF THE WAVEFORM DEFINED
                ;       BY THE SPECTRUM AT FSRAM AND FINDS THE MOST POSITIVE POINT AND
                ;       PUTS IT IN MAX AND FINDS THE MOST NEGATIVE POINT AND PUTS IT IN
                ;       MIN.  MAX AND MIN ARE DOUBLE PRECISION SIGNED NUMBERS

0039 A900       SCALE:  LDA   #0            ; ZERO THE POINT NUMBER
003B 850A               STA   PNTNO
003D 8510               STA   MIN+1
003F 850E               STA   MAX+1
0041 A9C0               LDA   #X'C0         ; SET MAX TO C000 (TO AVOID OVERFLOW WHEN
0043 850D               STA   MAX           ; COMPARING)
0045 A940               LDA   #X'40         ; SET MIN TO 4000 (TO AVOID OVERFLOW WHEN
0047 850F               STA   MIN           ; COMPARING)
0049 200001     SCALE1: JSR   FSEVAL        ; EVALUATE A WAVEFORM POINT
004C A608               LDX   HRMACC        ; GET WAVEFORM POINT INTO X AND Y
004E A409               LDY   HRMACC+1
0050 E40D               CPX   MAX           ; COMPARE WITH MAX
0052 300A               BMI   SCALE3        ; JUMP IF DEFINITELY LESS THAN MAX
0054 D004               BNE   SCALE2        ; JUMP IF DEFINITELY GREATER THAN MAX
0056 C40E               CPY   MAX+1         ; COMPARE LOW BYTES IF HIGH BYTES EQUAL
0058 9004               BCC   SCALE3        ; JUMP IF LESS THAN MAX
005A 860D       SCALE2: STX   MAX           ; UPDATE MAX IF EQUAL OR GREATER
005C 840E               STY   MAX+1
005E E40F       SCALE3: CPX   MIN           ; COMPARE WITH MIN
0060 3006               BMI   SCALE4        ; JUMP IF DEFINITELY LESS THAN MIN
0062 D008               BNE   SCALE5        ; JUMP IF DEFINITELY GREATER THAN MIN
0064 C410               CPY   MIN+1         ; COMPARE LOW BYTES IF HIGH BYTES EQUAL
0066 B004               BCS   SCALE5        ; JUMP IF GREATER THAN MIN
0068 860F       SCALE4: STX   MIN           ; UPDATE MIN IF EQUAL OR LESS
006A 8410               STY   MIN+1
006C E60A       SCALE5: INC   PNTNO         ; INCREMENT POINT NUMBER
006E D0D9               BNE   SCALE1        ; GO FOR NEXT POINT IF NOT DONE

                ;       DETERMINE THE SCALE FACTOR,   SCALEF=PKAMP/(MAX-MIN)

0070 A50E               LDA   MAX+1         ; COMPUTE MAX-MIN AND PUT INTO DIVISOR
0072 38                 SEC
0073 E510               SBC   MIN+1
0075 8505               STA   DVSR+1
0077 A50D               LDA   MAX
0079 E50F               SBC   MIN
007B 8504               STA   DVSR
007D A515               LDA   PKAMP         ; PUT PKAMP INTO DIVIDEND, NEXT TO MOST
007F 8501               STA   DVND+1        ; SIGNIFICANT BYTE
```



```
0081 A900               LDA   #0            ; ZERO THE OTHER DIVIDEND BYTES
0083 8500               STA   DVND
0085 8502               STA   DVND+2
0087 8503               STA   DVND+3
0089 206A02             JSR   SGNDIV        ; DO THE DIVISON
008C A502               LDA   DVND+2        ; MOVE THE QUOTIENT TO THE SCALE FACTOR
008E 8511               STA   SCALEF
0090 A503               LDA   DVND+3
0092 8512               STA   SCALEF+1

                ;       GENERATE THE WAVEFORM POINTS, SCALE THEM, AND PUT THEM IN THE
                ;       WAVEFORM TABLE

0094 A900       WAVE:   LDA   #0            ; ZERO THE POINT NUMBER
0096 850A               STA   PNTNO
0098 A8                 TAY                 ; ZERO THE WAVEFORM TABLE INDEX
0099 200001     WAVE1:  JSR   FSEVAL        ; EVALUATE A WAVEFORM POINT
009C A509               LDA   HRMACC+1      ; SUBTRACT MIN FROM THE POINT AND PLACE IT
009E 38                 SEC                 ; IN THE MULTIPLIER
009F E510               SBC   MIN+1
00A1 8503               STA   PROD+3
00A3 A511               LDA   SCALEF        ; MULTIPLY BY THE SCALE FACTOR COMPUTED
00A5 8504               STA   MPCD          ; EARLIER
00A7 A512               LDA   SCALEF+1
00A9 8505               STA   MPCD+1
00AB A508               LDA   HRMACC
00AD E50F               SBC   MIN
00AF 8502               STA   PROD+2
00B1 200002             JSR   SGNMPY
00B4 A501               LDA   PROD+1        ; GET SCALED, POSITIVE RESULT
00B6 9113               STA   (WAVEAD),Y    ; PUT IT INTO THE WAVEFORM TABLE
00B8 E60A               INC   PNTNO         ; INCREMENT POINT NUMBER
00BA C8                 INY                 ; INCREMENT WAVE TABLE POINTER
00BB D0DC               BNE   WAVE1         ; GO FOR ANOTHER POINT IF NOT FINISHED

                ;       WAVEFORM PLAYOUT FOR MONITORING
                ;       THIS ROUTINE CONTINUOUSLY SCANS THE WAVEFORM TABLE JUST
                ;       COMPUTED AND OUTPUTS THE WAVEFORM AT A SAMPLE RATE OF 8770 HZ,
                ;       THE SAME AS THE MUSIC PROGRAMS.  FUNDAMENTAL FREQUENCY IS 34 HZ
                ;       IT ALSO PULSES PORT B BIT 7 AT THE START OF EVERY CYCLE TO
                ;       SYNCHRONIZE A STANDARD OSCILLOSCOPE FOR WAVEFORM DISPLAY.
                ;       NOTE THAT PORT B BIT 7 IS PIN 15 ON THE APPLICATION CONNECTOR
                ;       AND THAT IT REQUIRES A 4.7K OHM PULLUP RESISTOR TO +5 TO WORK.

00BD AD0317     MONTOR: LDA   PTBDIR        ; SET PORT B BIT 7 UP AS AN OUTPUT
00C0 0980               ORA   #X'80
00C2 8D0317             STA   PTBDIR
00C5 A9FF               LDA   #X'FF         ; SET UP THE DAC PORT AS AN OUTPUT
00C7 8D0117             STA   DACDIR
00CA AD0217     MONT1:  LDA   PTB           ; PULSE PORT B BIT 7
00CD 0980               ORA   #X'80
00CF 8D0217             STA   PTB
00D2 297F               AND   #X'7F
00D4 8D0217             STA   PTB
00D7 A000               LDY   #0            ; INITIALIZE WAVEFORM TABLE POINTER
```

KIMFS KIM FOURIER SERIES
MAIN WAVEFORM COMPUTATION ROUTINE

```
 224 00D9 B113      MONT2:  LDA   (WAVEAD),Y  ; GET A SAMPLE FROM THE WAVEFORM TABLE
 225 00DB 8D0017            STA   DAC         ; SEND IT TO THE DAC
 226 00DE C8                INY               ; BUMP POINTER TO NEXT SAMPLE
 227 00DF F007              BEQ   MONT4       ; JUMP IF FINISHED WITH CYCLE
 228 00E1 A213              LDX   #19         ; WASTE SOME TIME TO GET LOOP TIME OF 114
 229 00E3 CA        MONT3:  DEX               ; STATES
 230 00E4 D0FD              BNE   MONT3
 231 00E6 F0F1              BEQ   MONT2       ; GO FOR NEXT SAMPLE
 232 00E8 A20F      MONT4:  LDX   #15         ; WASTE LESS TIME TO ACCOUNT FOR OVERHEAD
 233 00EA CA        MONT5:  DEX               ; IN GENERATING THE SYNC PULSE
 234 00EB D0FD              BNE   MONT5
 235 00ED F0DB              BEQ   MONT1       ; GO START ANOTHER WAVEFORM CYCLE
 236
```

KIMFS KIM FOURIER SERIES
FOURIER SERIES POINT EVALUATOR

```
                           .PAGE  'FOURIER SERIES POINT EVALUATOR'
 237                ;      THIS SUBROUTUM EVALUATES A POINT ON THE WAVEFORM SPECIFIED BY
 238                ;      THE SPECTRUM AT FSRAM.
 239                ;      NHARM SPECIFIES THE HIGHEST HARMONIC TO BE INCLUDED
 240                ;      PNTNO IS THE POINT NUMBER TO BE EVALUATED
 241                ;      THE COMPUTED POINT IS RETURNED IN IN HRMACC AS A 16 BIT TWOS
 242                ;      COMPLEMENT NUMBER
 243                ;      DESTROYS A, SAVES INDEX REGISTERS, USES ARITHMETIC SUBROUTINES
 244
 245 00EF                   .=    X'100       ; START FOURIER SERIES ROUTINES AT 100
 246
 247 0100 8A        FSEVAL: TXA               ; SAVE INDEX X
 248 0101 48                PHA
 249 0102 A900              LDA   #0          ; CLEAR HARMONIC ACCUMULATOR
 250 0104 8508              STA   HRMACC
 251 0106 8509              STA   HRMACC+1
 252 0108 850C              STA   HRMCNT      ; ZERO HARMONIC COUNTER
 253 010A 850B              STA   NDXACC      ; ZERO THE INDEXING ACCUMULATOR
 254 010C A50C      FSEV1:  LDA   HRMCNT      ; GET CURRENT HARMONIC NUMBER AND DOUBLE IT
 255 010E 0A                ASLA
 256 010F AA                TAX               ; USE AS AN INDEX TO THE SPECTRUM TABLE
 257 0110 B517              LDA   FSRAM,X     ; GET AMPLITUDE
 258 0112 8503              STA   PROD+3      ; SAVE AS MULTIPLIER
 259 0114 B518              LDA   FSRAM+1,X   ; GET PHASE
 260 0116 650B              ADC   NDXACC      ; ADD IT TO THE INDEXING ACCUMULATOR
 261 0118 205001            JSR   COSINE      ; GET COSINE
 262 011B 8504              STA   MPCD        ; SAVE AS MULTIPLICAND
 263 011D A900              LDA   #0          ; ZERO UNUSED BYTES IN MULTIPLIER AND
 264 011F 8502              STA   PROD+2      ; MULTIPLICAND
 265 0121 8505              STA   MPCD+1
 266 0123 200002            JSR   SGNMPY      ; MULTIPLY AMPLITUDE BY COSINE
 267 0126 A205              LDX   #5          ; SHIFT PRODUCT RIGHT 5 FOR 11 BIT RESULT
 268 0128 205402    FSEV2:  JSR   SRQA        ; RIGHT JUSTIFIED IN PROD+1 AND PROD+2
 269 012B CA                DEX
 270 012C D0FA              BNE   FSEV2
 271 012E 18                CLC               ; ADD RESULT TO HARMONIC ACCUMULATOR
 272 012F A509              LDA   HRMACC+1
 273 0131 6502              ADC   PROD+2
 274 0133 8509              STA   HRMACC+1
 275 0135 A508              LDA   HRMACC
 276 0137 6501              ADC   PROD+1
 277 0139 8508              STA   HRMACC
 278 013B A50C              LDA   HRMCNT      ; TEST IF CURRENT HARMONIC IS LAST ONE TO
 279 013D C516              CMP   NHARM       ; INCLUDE
 280 013F F00C              BEQ   FSEV3       ; GO RETURN IF SO
 281 0141 E60C              INC   HRMCNT      ; INCREMENT TO NEXT HARMONIC
 282 0143 A50A              LDA   PNTNO       ; ADD POINT NUMBER TO THE INDEXING
 283 0145 18                CLC               ; ACCUMULATOR
 284 0146 650B              ADC   NDXACC
 285 0148 850B              STA   NDXACC
 286 014A 4C0C01            JMP   FSEV1       ; LOOP FOR ANOTHER HARMONIC
 287 014D 68        FSEV3:  PLA               ; RESTORE INDEX X
 288 014E AA                TAX
 289 014F 60                RTS               ; RETURN
 290
```

```
                        .PAGE  'COSINE CALCULATE ROUTINE'
                ;       ENTER WITH ARGUMENT IN A, 0=0 DEGREES, 64=90 DEGREES, ETC.
                ;       EXIT WITH SIGNED COSINE IN A
                ;       DESTROYS INDEX X
                
0150 C940       COSINE: CMP    #X'40
0152 900B               BCC    QUAD1        ; BRANCH IF IN QUADRANT 1
0154 C980               CMP    #X'80
0156 900C               BCC    QUAD2        ; BRANCH IF IN QUADRANT 2
0158 C9C0               CMP    #X'C0
015A 9013               BCC    QUAD3        ; BRANCH IF IN QUADRANT 3
015C 207301     QUAD4:  JSR    ANGNEG       ; IN QUADRANT 4, NEGATE THE ANGLE
015F AA         QUAD1:  TAX                 ; IN QUADRANT 1, LEAVE ANGLE ALONE
0160 BD7B01             LDA    COSTAB,X     ; GET COSINE IN QUADRANTS 1 AND 4
0163 60                 RTS                 ; RETURN
0164 207301     QUAD2:  JSR    ANGNEG       ; IN QUADRANT 2 NEGATE THE ANGLE
0167 AA         COS1:   TAX
0168 A900               LDA    #0
016A 38                 SEC
016B FD7B01             SBC    COSTAB,X     ; GET -COSINE IN QUADRANTS 2 AND 3
016E 60                 RTS                 ; RETURN
016F 293F       QUAD3:  AND    #X'3F        ; IN QUADRANT 3 MASK THE ANGLE TO CONVERT
0171 90F4               BCC    COS1         ; TO QUADRANT 1 ANGLE

0173 293F       ANGNEG: AND    #X'3F        ; ISOLATE THE LOWER 6 BITS OF THE ANGLE
0175 493F               EOR    #X'3F        ; FLIP THEM
0177 18                 CLC                 ; ADD 1 FOR TWOS COMPLEMENT
0178 6901               ADC    #1
017A 60                 RTS                 ; RETURN


                ;       65 POINT COSINE TABLE, QUADRANT 1

017B 7F7F7F7F   COSTAB: .BYTE  X'7F,X'7F,X'7F,X'7F,X'7F,X'7F,X'7E,X'7E
0183 7D7D7C7B           .BYTE  X'7D,X'7D,X'7C,X'7B,X'7A,X'79,X'78,X'77
018B 76757372           .BYTE  X'76,X'75,X'73,X'72,X'71,X'6F,X'6D,X'6C
0193 6A686665           .BYTE  X'6A,X'68,X'66,X'65,X'63,X'61,X'5E,X'5C
019B 5A585653           .BYTE  X'5A,X'58,X'56,X'53,X'51,X'4E,X'4C,X'49
01A3 4744413F           .BYTE  X'47,X'44,X'41,X'3F,X'3C,X'39,X'36,X'33
01AB 312E2B28           .BYTE  X'31,X'2E,X'2B,X'28,X'25,X'22,X'1F,X'1C
01B3 191612OF           .BYTE  X'19,X'16,X'12,X'0F,X'0C,X'09,X'06,X'03
01BB 00                 .BYTE  X'00
```



```
                        .PAGE  'MULTIPLY ROUTINES'
                ;       SIGNED MULTIPLY SUBROUTINE
                ;       ENTER WITH SIGNED MULTIPLIER IN PROD+2 AND PROD+3
                ;       ENTER WITH SIGNED MULTIPLICAND IN MPCD AND MPCD+1
                ;       RETURN WITH 16 BIT SIGNED PRODUCT IN PROD (HIGH) THROUGH
                ;       PROD+3 (LOW)
                ;       A DESTROYED, X AND Y PRESERVED

01BC                    .=     X'0200        ; PUT MATH ROUTINES AT 200

0200 A502       SGNMPY: LDA    PROD+2        ; GET MULTIPLIER
0202 8506               STA    MPSAVE        ; AND SAVE IT
0204 A503               LDA    PROD+3
0206 8507               STA    MPSAVE+1
0208 202E02             JSR    UNSMPY        ; DO AN UNSIGNED MULTIPLY
020B A504               LDA    MPCD          ; TEST SIGN OF MULTIPLICAND
020D 100D               BPL    SGNMP1        ; JUMP IF POSITIVE
020F A501               LDA    PROD+1        ; SUBTRACT MULTIPLIER FROM HIGH PRODUCT IF
0211 38                 SEC                  ; NEGATIVE
0212 E507               SBC    MPSAVE+1
0214 8501               STA    PROD+1
0216 A500               LDA    PROD
0218 E506               SBC    MPSAVE
021A 8500               STA    PROD
021C A506       SGNMP1: LDA    MPSAVE        ; TEST SIGN OF MULTIPLIER
021E 100D               BPL    SGNMP2        ; GO RETURN IF POSITIVE
0220 A501               LDA    PROD+1        ; SUBTRACT MULTIPLICAND FROM HIGH PRODUCT
0222 38                 SEC                  ; IF NEGATIVE
0223 E505               SBC    MPCD+1
0225 8501               STA    PROD+1
0227 A500               LDA    PROD
0229 E504               SBC    MPCD
022B 8500               STA    PROD
022D 60         SGNMP2: RTS                  ; RETURN

                ;       16 X 16 UNSIGNED MULTIPLY SUBROUTINE
                ;       ENTER WITH UNSIGNED MULTIPLIER IN PROD+2 AND PROD+3
                ;       ENTER WITH UNSIGNED MULTIPLICAND IN MPCD AND MPCD+1
                ;       RETURN WITH 16 BIT UNSIGNED PRODUCT IN PROD (HIGH) THROUGH
                ;       PROD+3 (LOW)
                ;       A DESTROYED, X AND Y PRESERVED

022E 8A         UNSMPY: TXA                  ; SAVE X INDEX
022F 48                 PHA
0230 A900               LDA    #0            ; CLEAR UPPER PRODUCT
0232 8500               STA    PROD
0234 8501               STA    PROD+1
0236 A211               LDX    #17           ; SET 17 MULTIPLY CYCLE COUNT
0238 18                 CLC                  ; INITIALLY CLEAR CARRY
0239 205702     UNSM1:  JSR    SRQL          ; SHIFT MULTIPLIER AND PRODUCT RIGHT 1
                                             ; PUTTING A MULTIPLIER BIT IN CARRY
023C CA                 DEX                  ; DECREMENT AND CHECK CYCLE COUNT
023D F012               BEQ    UNSM2         ; JUMP OUT IF DONE
023F 90F8               BCC    UNSM1         ; SKIP MULTIPLICAND ADD IF MULTIPLIER BIT
                                             ; IS ZERO
```

KIMFS KIM FOURIER SERIES
MULTIPLY ROUTINES

```
388 0241 A501              LDA    PROD+1       ; ADD MULTIPLICAND TO UPPER PRODUCT
389 0243 18                CLC
390 0244 6505              ADC    MPCD+1
391 0246 8501              STA    PROD+1
392 0248 A500              LDA    PROD
393 024A 6504              ADC    MPCD
394 024C 8500              STA    PROD
395 024E 4C3902            JMP    UNSM1        ; GO FOR NEXT CYCLE
396 0251 68        UNSM2:  PLA                 ; RESTORE X
397 0252 AA                TAX
398 0253 60                RTS                 ; RETURN
399
```

KIMFS KIM FOURIER SERIES
QUAD PRECISION SHIFT ROUTINES

```
                           .PAGE  'QUAD PRECISION SHIFT ROUTINES'
400                ;       QUAD SHIFT RIGHT SUBROUTINE
401                ;       ENTER AT SRQA FOR ALGEBRAIC SHIFT RIGHT
402                ;       ENTER AT SRQL FOR LOGICAL SHIFT
403                ;       ENTER WITH QUAD PRECISION VALUE TO SHIFT IN PROD THROUGH PROD+3
404                ;       DESTROYS A, PRESERVES X AND Y, RETURNS BIT SHIFTED OUT IN CARRY
405 .
406 0254 A500      SRQA:   LDA    PROD         ; GET SIGN BIT OF PROD IN CARRY
407 0256 0A                ASLA
408 0257 6600      SRQL:   ROR    PROD         ; LOGICAL SHIFT RIGHT ENTRY
409 0259 6601              ROR    PROD+1
410 025B 6602              ROR    PROD+2
411 025D 6603              ROR    PROD+3
412 025F 60                RTS                 ; RETURN
413
414
415                ;       QUAD SHIFT LEFT SUBROUTINE
416                ;       ENTER AT SLQL TO SHIFT IN A ZERO BIT
417                ;       ENTER AT RLQL TO SHIFT IN THE CARRY
418                ;       ENTER WITH QUAD PRECISION VALUE TO SHIFT IN PROD THROUGH PROD+3
419                ;       DESTROYS A, PRESERVES X AND Y, RETURNS BIT SHIFTED OUT IN CARRY
420
421 0260 18        SLQL:   CLC                 ; SHIFT IN ZERO BIT ENTRY; CLEAR CARRY
422 0261 2603      RLQL:   ROL    PROD+3       ; SHIFT IN CARRY ENTRY
423 0263 2602              ROL    PROD+2
424 0265 2601              ROL    PROD+1
425 0267 2600              ROL    PROD
426 0269 60                RTS                 ; RETURN
427
```

```
                        .PAGE  'DIVIDE ROUTINES'
  428                ;       DOUBLE PRECISION SIGNED DIVIDE SUBRCUTINE
  429                ;       ENTER WITH SIGNED DIVIDEND IN DVND (HIGH) THROUGH DVND+3 (LOW)
  430                ;       ENTER WITH SIGNED DIVISOR IN DVSR AND DVSR+1
  431                ;       EXIT WITH SIGNED QUOTIENT IN DVND+2 AND DVND+3 AND SIGNED
  432                ;       REMAINDER TIMES 2 IN DVND AND DVND+1
  433                ;       DESTROYS A, PRESERVES INDEX REGISTERS
  434                ;       NO CHECK FOR OVERFLOW OR DIVIDE BY 0
  435
  436 026A 8A        SGNDIV: TXA                 ; SAVE THE INDEX REGISTERS
  437 026B 48                PHA
  438 026C 98                TYA
  439 026D 48                PHA
  440 026E A500      SDIV:   LDA    DVND         ; COMPUTE SIGN OF QUOTIENT
  441 0270 4504              EOR    DVSR
  442 0272 8506              STA    MPSAVE       ; SAVE IT
  443 0274 A504              LDA    DVSR         ; ABSOLUTE VALUE OF DIVISOR
  444 0276 1007              BPL    SDIV1        ; BRANCH IF POSITIVE
  445 0278 A205              LDX    #DVSR+1-DVND ; NEGATE IF NEGATIVE
  446 027A A002              LDY    #2
  447 027C 209D02            JSR    MPNEG
  448 027F A500      SDIV1:  LDA    DVND         ; ABSOLUTE VALUE OF DIVIDEND
  449 0281 1007              BPL    SDIV2        ; JUMP IF POSITIVE
  450 0283 A203              LDX    #DVND+3-DVND ; NEGATE IF NEGATIVE
  451 0285 A004              LDY    #4
  452 0287 209D02            JSR    MPNEG
  453 028A 20A902    SDIV2:  JSR    UNSDIV       ; DO AN UNSIGNED DIVIDE
  454 028D A506              LDA    MPSAVE       ; GET SIGN OF QUOTIENT
  455 028F 1007              BPL    SDIV3        ; GO RETURN IF POSITIVE
  456 0291 A203              LDX    #DVND+3-DVND ; NEGATE IF NEGATIVE
  457 0293 A002              LDY    #2
  458 0295 209D02            JSR    MPNEG
  459 0298 68        SDIV3:  PLA                 ; RESTORE INDEX REGISTERS
  460 0299 A8                TAY
  461 029A 68                PLA
  462 029B AA                TAX
  463 029C 60                RTS                 ; RETURN
  464
  465
  466 029D 38        MPNEG:  SEC                 ; INITIALLY SET CARRY
  467 029E A900      MPNEG1: LDA    #0           ; SUBTRACT A BYTE FROM 0 TO NEGATE IT
  468 02A0 F500              SBC    DVND,X
  469 02A2 9500              STA    DVND,X
  470 02A4 CA                DEX                 ; BUMP TO NEXT MORE SIGNIFICANT BYTE
  471 02A5 88                DEY                 ; CHECK BYTE COUNT
  472 02A6 D0F6              BNE    MPNEG1       ; LOOP IF NOT DONE
  473 02A8 60                RTS                 ; RETURN IF DONE
  474
  475                ;       DOUBLE PRECISION UNSIGNED DIVIDE SUBROUTINE
  476                ;       ENTER WITH UNSIGNED DIVIDEND IN PROD (HIGH) THROUGH PROD+3 (LOW
  477                ;       ENTER WITH UNSIGNED DIVISOR IN MPCD AND MPCD+1
  478                ;       EXIT WITH UNSIGNED QUOTIENT IN PROD+2 AND PROD+3 AND UNSIGNED
  479                ;       REMAINDER TIMES 2 IN PROD AND PROD+1 AND CARRY FLAG
  480                ;       DESTROYS A, PRESERVES INDEX REGISTERS
  481                ;       NO CHECK FOR OVERFLOW OR DIVIDE BY 0
```



```
  482
  483 02A9 8A        UNSDIV: TXA                 ; SAVE X
  484 02AA 48                PHA
  485 02AB 98                TYA                 ; SAVE Y
  486 02AC 48                PHA
  487 02AD A211              LDX    #17          ; SET DIVIDE CYCLE COUNT
  488 02AF 18                CLC                 ; INITIALLY CLEAR CARRY
  489 02B0 A501      UNSDV1: LDA    PROD+1       ; SUBTRACT DIVISOR FROM HIGH DIVIDEND
  490 02B2 38                SEC
  491 02B3 E505              SBC    MPCD+1
  492 02B5 A8                TAY
  493 02B6 A500              LDA    PROD
  494 02B8 E504              SBC    MPCD
  495 02BA 9005              BCC    UNSDV2       ; SKIP IF OVERDRAW
  496 02BC 8500              STA    PROD         ; UPDATE HIGH DIVIDEND IF NOT
  497 02BE 98                TYA
  498 02BF 8501              STA    PROD+1
  499 02C1 206102    UNSDV2: JSR    RLQL         ; SHIFT DIVIDEND LEFT 1 BRINGING IN
  500                                            ; QUOTIENT BIT
  501 02C4 CA                DEX                 ; DECREMENT CYCLE COUNT
  502 02C5 D0E9              BNE    UNSDV1       ; LOOP IF NOT DONE
  503 02C7 68                PLA                 ; RESTORE Y
  504 02C8 A8                TAY
  505 02C9 68                PLA                 ; RESTORE X
  506 02CA AA                TAX
  507 02CB 60                RTS                 ; RETURN
  508
  509 0000                   .END
NO ERROR LINES
```

```
                        .PAGE  'DOCUMENTATION'
                ;       COPYRIGHT 1977 BY MICRO TECHNOLOGY UNLIMITED, BOX 4596,
                ;       MANCHESTER, NEW HAMPSHIRE 03108
                ;       PROGRAM WRITTEN BY HAL CHAMBERLIN

                ;       THIS PROGRAM INTERPRETS NOTRAN OBJECT CODE GENERATED BY THE
                ;       NOTRAN COMPILER.
                ;       4 NON-DYNAMIC VOICES ARE SUPPORTED.
                ;       AN 8-BIT UNSIGNED DIGITAL-TO-ANALOG CONVERTER IS USED FOR THE
                ;       AUDIO OUTPUT.
                ;       OUTPUT SAMPLE RATE IS 8.77 KHZ WITH A 1.0 MHZ 6502 PROCESSOR

                ;       THE INTERPRETIVE OBJECT CODE IS TIME-ORDERED IN MEMORY STARTING
                ;       AT THE ADDRESS IN "SONGA" AND ENDING AT AN END COMMAND BYTE OR
                ;       AN INFINITE LOOP.  EACH COMMAND IN THE OBJECT CODE CONSISTS OF
                ;       1, 2, OR 3 BYTES.  SOME COMMANDS ARE SPECIFICATIONS THAT TAKE
                ;       NO "TIME" THEMSELVES SUCH AS A TEMPO CHANGE SPECIFICATION.
                ;       OTHERS DIRECT THE PLAYING OF NOTES AND REQUIRE AN AMOUNT OF
                ;       TIME DETERMINED BY THEIR DURATION FIELD AND THE CURRENT TEMPO.

                ;       THE FIRST BYTE OF A COMMAND IS THE COMMAND BYTE.  IT IS SPLIT
                ;       INTO TWO 4 BIT FIELDS, THE PITCH FIELD (UPPER HEX DIGIT) AND
                ;       THE DURATION FIELD (LOWER HEX DIGIT).  THE PITCH FIELD IS
                ;       INTERPRETED AS A SIGNED 4 BIT BINARY NUMBER.  IT REPRESENTS A
                ;       PITCH "DISPLACEMENT" FROM THE LAST NOTE PLAYED BY THE VOICE
                ;       BEING PROCESSED.  POSITIVE VALUES SPECIFY AN ASCENDING PITCH
                ;       WHILE NEGATIVE VALUES SPECIFY A DESCENDING PITCH.  THE VALUE -8
                ;       IS USED TO SPECIFY SILENCE FOR THE AFFECTED VOICE.  THUS
                ;       CONSECUTIVE NOTES MAY BE AT INTERVALS OF OVER A HALF OCTAVE
                ;       WITHOUT REQUIRING THE MULTI-BYTE LONG NOTE SPECIFICATION TO BE
                ;       USED.  THE RELATIVE NATURE OF THE NOTES IS ALSO USEFUL IN
                ;       "MUSICAL SUBROUTINES" TO BE DESCRIBED.

                ;       THE DURATION FIELD SPECIFIES THE DURATION OF THE SPECIFIED
                ;       PITCH FOR THE CURRENT VOICE.  THE DURATION CODE IS AS FOLLOWS:

                ;         0 = CONTROL ESCAPE            8 = QUARTER-TRIPLET (1/6)
                ;         1 = WHOLE NOTE                9 = EIGHTH NOTE
                ;         2 = DOTTED HALF NOTE          A = DOTTED SIXTEENTH NOTE
                ;         3 = HALF NOTE                 B = EIGHTH-TRIPLET (1/12)
                ;         4 = DOTTED QUARTER NOTE       C = SIXTEENTH NOTE
                ;         5 = HALF-TRIPLET (1/3)        D = DOTTED THIRTY-SECOND NOTE
                ;         6 = QUARTER NOTE              E = SIXTEENTH-TRIPLET (1/24)
                ;         7 = DOTTED EIGHTH NOTE        F = THIRTY=SECOND NOTE

                ;       NOTE THAT IF THE DURATION CODE IS ZERO THEN THE PITCH FIELD IS
                ;       INTERPRETED AS FOLLOWS:

                ;         0 = END OF SONG, RETURN TO KIM-1 MONITOR
                ;         1 = CHANGE OF TEMPO, NEXT BYTE IS A NEW TEMPO BYTE
                ;         2 = CALL TO A MUSICAL SUBROUTINE, TWO BYTE SUBROUTINE ADDRESS
                ;             FOLLOWS, LOW BYTE FIRST
                ;         3 = RETURN FROM A MUSICAL SUBROUTINE
                ;         4 = UNCONDITIONAL JUMP, TWO BYTE JUMP ADDRESS FOLLOWS
                ;         5 = SET NUMBER OF VOICES, NUMBER BETWEEN 1 AND 4 IN NEXT BYTE
```

NOTINT NOTRAN INTERPRETER
DOCUMENTATION

```
                ;         6 = LONG NOTE SPECIFICATION WITH ABSOLUTE PITCH SPECIFICATION
                ;             2 ADDITIONAL BYTES FOLLOW
                ;         7 = LONG NOTE SPECIFICATION WITH RELATIVE PITCH SPECIFICATION
                ;             2 ADDITIONAL BYTES FOLLOW
                ;         8 = DEACTIVATE VOICE, VOICE NUMBER TO DEACTIVATE IN NEXT BYTE
                ;         9 = ACTIVATE VOICE, VOICE NUMBER TO ACTIVATE IN NEXT BYTE
                ;         A-F RESERVED FOR FUTURE EXPANSION, IF USED CAUSES RETURN TO
                ;             KIM MONITOR

                ;       A LONG NOTE SPECIFICATION CONSISTS OF TWO ADDITIONAL BYTES AS
                ;       FOLLOWS:
                ;       BYTE 1 - PITCH; IF ABSOLUTE, IT IS TREATED AS RELATIVE TO THE
                ;                BEGINNING OF THE NOTE FREQUENCY TABLE.  IF RELATIVE,
                ;                IT IS TREATED AS RELATIVE TO THE LAST PITCH SOUNDED
                ;                BY THE CURRENT VOICE. IN EITHER CASE IT IS A SIGNED,
                ;                TWOS COMPLEMENT NUMBER.
                ;       BYTE 2 - UPPER DIGIT, WAVEFORM TABLE NUMBER.  THIS IS ADDED
                ;                TO THE PAGE ADDRESS OF THE WAVEFORM TABLE AREA TO
                ;                YEILD THE PAGE ADDRESS OF THE WAVEFORM TABLE TO BE
                ;                USED FOR THIS VOICE.
                ;                LOWER DIGIT - DURATION CODE AS LISTED PREVIOUSLY.
                ;                A ZERO DURATION IS ILLEGAL.

                ;       WHEN SELECTING THE WAVEFORM TABLES TO BE ASSIGNED TO THE VOICES
                ;       IT IS IMPORTANT THAT OVERFLOW WHEN THE VOICES ARE COMBINED IS
                ;       AVOIDED.  WHEN CHECKING FOR AN OVERFLOW POSSIBILITY, AN
                ;       INACTIVE VOICE STILL CONTRIBUTES TO THE SUM (AN INACTIVE VOICE
                ;       MERELY HAS A ZERO FREQUENCY) WHEREAS A DISABLED VOICE DOES NOT
                ;       (IT IS NOT SUMMED).  TO DETERMINE IF OVERFLOW IS POSSIBLE, ADD
                ;       UP THE LARGEST SAMPLE FROM THE WAVEFORM TABLE CORRESPONDING TO
                ;       EACH ENABLED VOICE AND VERIFY THAT THE SUM IS 255 OR LESS.  AN
                ;       ABSOLUTELY SAFE WAY TO AVOID OVERFLOW IS TO SPECIFY WHEN THE
                ;       WAVEFORM IS CREATED WITH THE FOURIER SERIES PROGRAM AN
                ;       AMPLITUDE OF 255/N WHERE N IS THE NUMBER OF ENABLED VOICES.
                ;       HOWEVER BY PROPERLY UNDERSTANDING THE OVERFLOW RESTRICTION ONE
                ;       MAY MAKE SOME VOICES LOUDER THAN OTHERS AND EVEN IMPLEMENT
                ;       LIMITED DYNAMICS.  NOTE THAT ACTIVATING OR DEACTIVATING A VOICE
                ;       IS NOISELESS WHEREAS ENABLING OR DISABLING ONE MAY BE
                ;       ACCOMPANIED BY A CLICK.

                ;       VOICE 1 IS ALWAYS ENABLED.  ENABLED VOICES MUST ALWAYS BE IN
                ;       SEQUENCE STARTING FROM VOICE 1.  AS AN EXAMPLE, IF THE NUMBER
                ;       OF VOICES IS SET TO 3, THEN VOICES 1 - 3 ARE EANBLED AND VOICE
                ;       4 IS DISABLED.

                ;             RULES FOR INTERPRETING NOTRAN OBJECT CODE

                ;       WHEN THE INTERPRETER IS INITIALLY ENTERED, ALL 4 VOICES WILL BE
                ;       DEACTIVATED AND INTERPRETATION WILL START AT THE ADDRESS IN
                ;       SONGA.  THE INTERPRETER EXPECTS TO SEE PURE CONTROL COMMANDS
                ;       (DURATION FIELD = 0 AND PITCH FIELD BETWEEN 0 AND B) FIRST.  AT
                ;       THE BEGINNING OF A SONG, AT LEAST ONE WOULD BE USED TO SET UP
                ;       THE TEMPO.  AT OTHER TIMES THEY ARE USED TO CHANGE THE TEMPO,
                ;       DEACTIVATE VOICES THAT WILL BE SILENT FOR A TIME, ACTIVATE
                ;       VOICES THAT HAVE BEEN IDLE, ALTER THE FLOW OF CONTROL THROUGH
```

```
                ;       JUMPS OR SUBROUTINE CALLS, AND TO MARK THE END OF THE SONG.
                ;
                ;       FOLLOWING THE PURE CONTROL COMMANDS, IF ANY, NOTE COMMANDS ARE
                ;       EXPECTED.  AT THE BEGINNING OF A SONG LONG NOTE COMMANDS MUST
                ;       BE USED TO ESTABLISH THE WAVEFORMS TO BE USED WITH EACH VOICE
                ;       AND ESTABLISH A PITCH REFERENCE FOR THE RELATIVE PITCHES THAT
                ;       MAY APPEAR LATER.  AT OTHER TIMES LONG NOTE COMMANDS MAY BE
                ;       USED TO CHANGE WAVEFORMS AND ADDRESS PITCHES THAT ARE
                ;       UNREACHABLE WITH THE RELATIVE PITCH SPECIFICATION IN A SHORT
                ;       NOTE COMMAND.  SHORT NOTE COMMANDS (DURATION FIELD IS NON-ZERO)
                ;       MAY BE FREELY INTERMIXED WITH LONG NOTE COMMANDS AND SHOULD BE
                ;       USED WHEN POSSIBLE TO MINIMIZE THE AMOUNT OF STORAGE REQUIRED
                ;       FOR A SONG.
                ;
                ;       NOTE COMMANDS ARE SUCCESSIVELY FETCHED, INTERPRETED, AND
                ;       ASSINGED IN ORDER TO ACTIVE VOICES WHOSE NOTES HAVE EXPIRED
                ;       UNTIL ALL OF THE ACTIVE VOICES HAVE BEEN SATISFIED.  SINCE ALL
                ;       NOTE COMMANDS LACK A FIELD THAT SPECIFICALLY ASSIGNS THE NOTE
                ;       TO A PARTICULAR VOICE, THEY ARE ASSIGNED TO EXPIRED ACTIVE
                ;       VOICES IN SEQUENCE STARTING WITH VOICE 1.  WHEN ALL EXPIRED
                ;       ACTIVE VOICES HAVE BEEN LOADED UP, THEY PLAY THEIR NOTES UNTIL
                ;       THE ONE WITH THE SHORTEST DURATION RUNS OUT.  THEN THE
                ;       INTERPRETER REGAINS CONTROL EXPECTING PURE CONTROL COMMANDS
                ;       AGAIN.
```



```
                        .PAGE   'EQUATES AND BASE PAGE STORAGE'

0000                    .=      0               ; ORG AT PAGE 0 LOCATION 0

1700            DAC     =       X'1700          ; OUTPUT PORT ADDRESS WITH DAC
1701            DACDIR  =       X'1701          ; DATA DRIECTION REGISTER FOR DAC PORT
1C22            KIMMON  =       X'1C22          ; ENTRY POINT TO KIM KEYBOARD MONITOR
00FF            ISTKPT  =       X'FF            ; INITIAL VALUE OF STACK POINTER

                                                ; FORMAT OF TABLE AREA FOR VOICE X
0000            VXPTF   =       0               ; VOICE WAVE POINTER, FRACTIONAL PART
0001            VXPTI   =       1               ; INTEGER PART
0002            VXPTW   =       2               ; WAVE TABLE PAGE NUMBER
0003            VXNOT   =       3               ; DISPLACEMENT IN NOTE FREQUENCY TABLE
0004            VXIN    =       4               ; WAVE TABLE POINTER INCREMENT (FREQUENCY
                                                ; PARAMETER, DOUBLE PRECISION)
0006            VXDUR   =       6               ; NOTE DURATION

                ;       ADDRESSES OF OBJECT CODE AREA AND WAVEFORM TABLE AREA

0000 0000       SONGA:  .WORD   0               ; ADDRESS OF OBJECT CODE AREA
0002 0000       WAVTBA: .WORD   0               ; ADDRESS OF WAVEFORM TABLE AREA

                ;       STORAGE REQUIRED BY THE INTERPRETER

0004 0000       CODEPT: .WORD   0               ; OBJECT CODE POINTER
0006 00         TEMPO:  .BYTE   0               ; TEMPO CONTROL VALUE
0007 00         DUR:    .BYTE   0               ; EVENT DURATION
0008 00         DURC:   .BYTE   0               ; EVENT DURATION COUNTER
0009            V1TBA:  .=.+    8               ; 8 BYTES FOR VOICE 1 TABLE AREA + 1 SPARE
0011            V2TBA:  .=.+    8               ; 8 BYTES FOR VOICE 2 TABLE AREA + 1 SPARE
0019            V3TBA:  .=.+    8               ; 8 BYTES FOR VOICE 3 TABLE AREA + 1 SPARE
0021            V4TBA:  .=.+    8               ; 8 BYTES FOR VOICE 4 TABLE AREA + 1 SPARE
0029 00         XSAVE:  .BYTE   0               ; AREA FOR QUICK SAVE OF X
002A 00         YSAVE:  .BYTE   0               ; AREA FOR QUICK SAVE OF Y
002B 00         CMSAVE: .BYTE   0               ; AREA FOR SAVING COMMAND BYTE
002C 0F         CTLM:   .BYTE   X'0F            ; MASK FOR TESTING DURATION FIELD
002D 3E00       NULSMP: .WORD   FRQTAB          ; ADDRESS OF NULL SAMPLE FOR USE IN
                                                ; EFFECTIVELY REDUCING NUMBER OF VOICES
                                                ; ADDED UP

                ; NOTE DURATION TABLE

002F C0         DURTAB: .BYTE   192             ; 1 WHOLE NOTE
0030 90                 .BYTE   144             ; 2 DOTTED HALF NOTE
0031 60                 .BYTE   96              ; 3 HALF NOTE
0032 48                 .BYTE   72              ; 4 DOTTED QUARTER NOTE
0033 40                 .BYTE   64              ; 5 HALF NOTE TRIPLET
0034 30                 .BYTE   48              ; 6 QUARTER NOTE
0035 24                 .BYTE   36              ; 7 DOTTED EIGHTH NOTE
0036 20                 .BYTE   32              ; 8 QUARTER NOTE TRIPLET
0037 18                 .BYTE   24              ; 9 EIGHTH NOTE
0038 12                 .BYTE   18              ; A DOTTED SIXTEENTH NOTE
0039 10                 .BYTE   16              ; B EIGHTH NOTE TRIPLET
003A 0C                 .BYTE   12              ; C SIXTEENTH NOTE
```

NOTINT NOTRAN INTERPRETER
EQUATES AND BASE PAGE STORAGE

```
                        .BYTE  9            ; D DOTTED 1/32 NOTE
                        .BYTE  8            ; E SIXTEENTH NOTE TRIPLET
                        .BYTE  6            ; F 1/32 NOTE

                ;       NOTE FREQUENCY TABLE FOR 8.772 KHZ SAMPLE RATE
                ;       RANGE FROM C1 (32.7 HZ) TO C6 (1046.5 HZ)
                ;                                ID  NOTE  FREQ.    INCR.
003E 0000       FRQTAB: .BYTE  0,0          ;  00  SILENCE
0040 00F4               .BYTE  0,244        ;  02  C1    32.702  .95445
0042 0103               .BYTE  1,3          ;  04  C1#   34.647  1.0112
0044 0112               .BYTE  1,18         ;  06  D1    36.707  1.0713
0046 0123               .BYTE  1,35         ;  08  D1#   38.891  1.1350
0048 0134               .BYTE  1,52         ;  0A  E1    41.204  1.2025
004A 0146               .BYTE  1,70         ;  0C  F1    43.654  1.2740
004C 015A               .BYTE  1,90         ;  0E  F1#   46.249  1.3498
004E 016E               .BYTE  1,110        ;  10  G1    48.999  1.4300
0050 0184               .BYTE  1,132        ;  12  G1#   51.915  1.5151
0052 019B               .BYTE  1,155        ;  14  A1    55.000  1.6052
0054 01B3               .BYTE  1,179        ;  16  A1#   58.270  1.7006
0056 01CD               .BYTE  1,205        ;  18  B1    61.735  1.8017
0058 01E9               .BYTE  1,233        ;  1A  C1    65.405  1.9089
005A 0206               .BYTE  2,6          ;  1C  C2#   69.295  2.0224
005C 0225               .BYTE  2,37         ;  1E  D2    73.415  2.1427
005E 0245               .BYTE  2,69         ;  20  D2#   77.783  2.2701
0060 0268               .BYTE  2,104        ;  22  E2    82.408  2.4051
0062 028C               .BYTE  2,140        ;  24  F2    87.308  2.5481
0064 02B3               .BYTE  2,179        ;  26  F2#   92.498  2.6996
0066 02DC               .BYTE  2,220        ;  28  G2    97.998  2.8601
0068 0308               .BYTE  3,8          ;  2A  G2#   103.83  3.0302
006A 0336               .BYTE  3,54         ;  2C  A2    110.00  3.2104
006C 0367               .BYTE  3,103        ;  2E  A2#   116.54  3.4013
006E 039A               .BYTE  3,154        ;  30  B2    123.47  3.6035
0070 03D1               .BYTE  3,209        ;  32  C3    130.81  3.8178
0072 040B               .BYTE  4,11         ;  34  C3#   138.59  4.0448
0074 0449               .BYTE  4,73         ;  36  D3    146.83  4.2854
0076 048A               .BYTE  4,138        ;  38  D3#   155.57  4.5402
0078 04CF               .BYTE  4,207        ;  3A  E3    164.82  4.8102
007A 0519               .BYTE  5,25         ;  3C  F3    174.62  5.0962
007C 0566               .BYTE  5,102        ;  3E  F3#   185.00  5.3992
007E 05B8               .BYTE  5,184        ;  40  G3    196.00  5.7203
0080 060F               .BYTE  6,15         ;  42  G3#   207.65  6.0604
0082 066C               .BYTE  6,108        ;  44  A3    220.00  6.4208
0084 06CD               .BYTE  6,205        ;  46  A3#   233.08  6.8026
0086 0735               .BYTE  7,53         ;  48  B3    246.94  7.2071
0088 07A3               .BYTE  7,163        ;  4A  C4    261.62  7.6356
008A 0817               .BYTE  8,23         ;  4C  C4#   277.18  8.0897
008C 0892               .BYTE  8,146        ;  4E  D4    293.66  8.5707
008E 0915               .BYTE  9,21         ;  50  D4#   311.13  9.0804
0090 099F               .BYTE  9,159        ;  52  E4    329.63  9.6203
0092 0A31               .BYTE  10,49        ;  54  F4    349.23  10.1924
0094 0ACC               .BYTE  10,204       ;  56  F4#   369.99  10.7984
0096 0B71               .BYTE  11,113       ;  58  G4    391.99  11.4405
0098 0C1F               .BYTE  12,31        ;  5A  G4#   415.30  12.1208
009A 0CD7               .BYTE  12,215       ;  5C  A4    440.00  12.8416
009C 0D9B               .BYTE  13,155       ;  5E  A4#   466.16  13.6052
```

NOTINT NOTRAN INTERPRETER
EQUATES AND BASE PAGE STORAGE

```
009E 0E6A               .BYTE  14,106       ;  60  B4    493.88  14.4142
00A0 0F45               .BYTE  15,69        ;  62  C5    523.24  15.2713
00A2 102E               .BYTE  16,46        ;  64  C5#   554.36  16.1794
00A4 1124               .BYTE  17,36        ;  66  D5    587.32  17.1414
00A6 1229               .BYTE  18,41        ;  68  D5#   622.26  18.1607
00A8 133E               .BYTE  19,62        ;  6A  E5    659.26  19.2406
00AA 1462               .BYTE  20,98        ;  6C  F5    698.46  20.3847
00AC 1599               .BYTE  21,153       ;  6E  F5#   739.98  21.5969
00AE 16E2               .BYTE  22,226       ;  70  G5    783.98  22.8811
00B0 183E               .BYTE  24,62        ;  72  G5#   830.60  24.2417
00B2 19AF               .BYTE  25,175       ;  74  A5    880.00  25.6831
00B4 1B36               .BYTE  27,54        ;  76  A5#   932.32  27.2103
00B6 1CD4               .BYTE  28,212       ;  78  B5    987.76  28.8283
00B8 1E8B               .BYTE  30,139       ;  7A  C6    1046.5  30.5426

                ;       JUMP TABLE FOR INTERPRETING PURE CONTROL COMMANDS

00BA 0000       JADDR:  .WORD  0            ; INDIRECT JUMP POINTER
00BC 221C       JTAB:   .WORD  KIMMON       ; 00 RETURN TO KIM MONITOR
00BE 4D02               .WORD  PCC10        ; 10 TEMPO SET
00C0 5602               .WORD  PCC20        ; 20 MUSICAL SUBROUTINE CALL
00C2 7302               .WORD  PCC30        ; 30 RETURN FROM MUSICAL SUBROUTINE
00C4 5E02               .WORD  PCC40        ; 40 UNCONDITIONAL JUMP
00C6 8102               .WORD  PCC50        ; 50 SET NUMBER OF VOICES
00C8 B302               .WORD  NOTE         ; 60 LONG NOTE WITH ABSOLUTE PITCH
00CA B302               .WORD  NOTE         ; 70 LONG NOTE WITH RELATIVE PITCH
00CC 8B02               .WORD  PCC80        ; 80 ACTIVATE VOICE
00CE A202               .WORD  PCC90        ; 90 DEACTIVATE VOICE
00D0 221C               .WORD  KIMMON       ; A0 UNDEFINED
00D2 221C               .WORD  KIMMON       ; B0 UNDEFINED
00D4 221C               .WORD  KIMMON       ; C0 UNDEFINED
00D6 221C               .WORD  KIMMON       ; D0 UNDEFINED
00D8 221C               .WORD  KIMMON       ; E0 UNDEFINED
00DA 221C               .WORD  KIMMON       ; F0 UNDEFINED
```

```
                                .PAGE  'INITIALIZATION'
  281                   ;       INITIALIZATION ROUTINE
  282                   ;       INITIALIZE THE MUSIC HARDWARE, SET UP THE OBJECT CODE POINTER,
  283                   ;       AND DEACTIVATE ALL 4 VOICES
  284
  285 00DC                      .=      X'200           ; START INTERPRETER AT 200
  286
  287 0200 A9FF         MUSIC:  LDA     #X'FF           ; SET DAC PORT DATA DIRECTION REGISTER TO
  288 0202 8D0117               STA     DACDIR          ; OUTPUT
  289 0205 D8                   CLD                     ; INSURE BINARY ARITHMETIC
  290 0206 A2FF                 LDX     #ISTKPT         ; INITIALIZE STACK POINTER
  291 0208 9A                   TXS
  292 0209 A900                 LDA     #0              ; ZERO THE PREVIOUS DURATION
  293 020B 8507                 STA     DUR
  294 020D A500                 LDA     SONGA           ; INITIALIZE OBJECT CODE POINTER TO
  295 020F 8504                 STA     CODEPT          ; BEGINNING OF OBJECT CODE
  296 0211 A501                 LDA     SONGA+1
  297 0213 8505                 STA     CODEPT+1
  298
  299 0215 A200                 LDX     #0              ; INITIALIZE ALL 4 VOICES
  300 0217 A900         DEACT:  LDA     #0
  301 0219 950D                 STA     V1TBA+VXIN,X    ; ZERO THE WAVE TABLE INCREMENT FOR SILENCE
  302 021B 950E                 STA     V1TBA+VXIN+1,X
  303 021D A9FF                 LDA     #X'FF           ; SET THE VOICE DURATION TO FF TO
  304 021F 950F                 STA     V1TBA+VXDUR,X;  DEACTIVATE
  305 0221 A503                 LDA     WAVTBA+1        ; INITIALLY ASSIGN WAVEFORM 1 TO ALL VOICES
  306 0223 950B                 STA     V1TBA+VXPTW,X
  307 0225 8A                   TXA                     ; BUMP INDEX UP TO NEXT VOICE
  308 0226 18                   CLC
  309 0227 6908                 ADC     #8
  310 0229 AA                   TAX
  311 022A C920                 CMP     #32             ; TEST IF DONE
  312 022C D0E9                 BNE     DEACT           ; LOOP UNTIL 4 VOICES DONE
  313 022E A904                 LDA     #4              ; SET NUMBER OF VOICES TO 4
  314 0230 20CB03               JSR     SETNV
  315
```



```
                                .PAGE  'CODE SEGMENT INTERPRETER'
  316                   ;       INTERPRET CODE SEGMENTS UNTIL ALL ACTIVE, UNEXPIRED VOICES HAVE
  317                   ;       BEEN SATISFIED AND THEN GO TO THE SOUND GENERATION ROUTINE
  318
  319                   ;       INITIALLY A "PURE CONTROL COMMAND" IS EXPECTED
  320
  321 0233 A000         PCC:    LDY     #0              ; INITIALIZE INDEX Y TO POINT TO FIRST BYTE
  322                                                   ; OF CODE SEGMENT
  323 0235 B104         PCC1:   LDA     (CODEPT),Y      ; GET FIRST BYTE OF CODE SEGMENT
  324 0237 242C                 BIT     CTLM            ; TEST IF LOW HEX DIGIT IS ZERO
  325 0239 F003                 BEQ     PCC2
  326 023B 4CB302               JMP     NOTE            ; JUMP IF NOT, IT IS A SHORT NOTE COMMAND
  327 023E 4A           PCC2:   LSRA                    ; PERFORM A VECTOR JUMP ON THE HIGH HEX
  328 023F 4A                   LSRA                    ; DIGIT
  329 0240 4A                   LSRA
  330 0241 AA                   TAX
  331 0242 B5BC                 LDA     JTAB,X
  332 0244 85BA                 STA     JADDR
  333 0246 B5BD                 LDA     JTAB+1,X
  334 0248 85BB                 STA     JADDR+1
  335 024A 6CBA00               JMP     (JADDR)
  336
  337                   ;       PROCESS TEMPO CHANGE COMMAND
  338
  339 024D C8           PCC10:  INY                     ; GET NEXT BYTE FROM CODE STRING
  340 024E B104                 LDA     (CODEPT),Y
  341 0250 8506                 STA     TEMPO           ; UPDATE THE TEMPO
  342 0252 C8                   INY
  343 0253 4C3502               JMP     PCC1            ; GO FOR NEXT COMMAND
  344
  345                   ;       PROCESS MUSICAL SUBROUTINE CALL
  346
  347 0256 A504         PCC20:  LDA     CODEPT          ; SAVE CURRENT VALUE OF OBJECT CODE POINTER
  348 0258 48                   PHA                     ; ON THE STACK
  349 0259 A505                 LDA     CODEPT+1
  350 025B 48                   PHA                     ; AND PROCESS THE NEXT 2 BYTES AS AN
  351 025C 98                   TYA                     ; UNCONDITIONAL JUMP
  352 025D 48                   PHA
  353
  354                   ;       PROCESS AN UNCONDITIONAL JUMP
  355
  356 025E C8           PCC40:  INY                     ; GET NEXT TWO BYTES FROM THE CODE STRING
  357 025F B104                 LDA     (CODEPT),Y
  358 0261 18                   CLC                     ; AND ADD THEM TO THE CODE STRING ORIGIN
  359 0262 6500                 ADC     SONGA           ; FOR RELATIVE ADDRESSING WITHIN OBJECT
  360 0264 AA                   TAX                     ; CODE
  361 0265 C8                   INY
  362 0266 B104                 LDA     (CODEPT),Y
  363 0268 6501                 ADC     SONGA+1
  364 026A 8505                 STA     CODEPT+1        ; STORE THEM IN THE CODE STRING POINTER
  365 026C 8604                 STX     CODEPT
  366 026E A000                 LDY     #0              ; ZERO THE INDEX FACTOR
  367 0270 4C3502               JMP     PCC1            ; GO FOR NEXT COMMAND
  368
  369                   ;       PROCESS A RETURN FROM MUSICAL SUBROUTINE
```

```
370
371 0273 68        PCC30:  PLA                 ; POP THE SAVED CODE POINTER FROM THE STAC
372 0274 A8                TAY
373 0275 68                PLA
374 0276 8505              STA   CODEPT+1
375 0278 68                PLA
376 0279 8504              STA   CODEPT
377 027B C8                INY                 ; INCREMENT THE SAVED INDEX FACTOR BY 3
378 027C C8                INY
379 027D C8                INY
380 027E 4C3502            JMP   PCC1          ; AND GO FOR THE NEXT COMMAND
381
382                ;       PROCESS A SET NUMBER OF VOICES COMMAND
383
384 0281 C8        PCC50:  INY                 ; GET SPECIFIED NUMBER OF VOICES
385 0282 B104              LDA   (CODEPT),Y    ; FROM NEXT CODE BYTE
386 0284 20CB03            JSR   SETNV         ; DO INSTRUCTION MODIFICATION IN SOUND
387                                            ; GENERATE ROUTINE TO SET THE NUMBER OF
388 0287 C8                INY                 ; VOICES
389 0288 4C3502            JMP   PCC1          ; GO FOR NEXT COMMAND
390
391                ;       PROCESS A DEACTIVATE VOICE COMMAND
392
393 028B C8        PCC80:  INY                 ; GET SPECIFIED VOICE NUMBER TIMES 8
394 028C B104              LDA   (CODEPT),Y
395 028E 2903              AND   #X'03         ; RESTRICT RANGE TO 0 - 3
396 0290 0A                ASLA
397 0291 0A                ASLA
398 0292 0A                ASLA
399 0293 AA                TAX                 ; PUT IT INTO X
400 0294 A9FF              LDA   #X'FF         ; SET THE DURATION BYTE IN THE SPECIFIED
401 0296 950F              STA   V1TBA+VXDUR,X; VOICE TO FF TO DEACTIVATE IT
402 0298 A900              LDA   #0            ; ZERO THE WAVE TABLE INCREMENT FOR SILENCE
403 029A 950D              STA   V1TBA+VXIN,X
404 029C 950E              STA   V1TBA+VXIN+1,X
405 029E C8                INY                 ; INCREMENT THE CODE STRING POINTER
406 029F 4C3502            JMP   PCC1          ; GO FOR NEXT COMMAND
407
408                ;       PROCESS AN ACTIVATE VOICE COMMAND
409
410 02A2 C8        PCC90:  INY                 ; COMPUTE SPECIFIED VOICE NUMBER TIMES 8
411 02A3 B104              LDA   (CODEPT),Y
412 02A5 2903              AND   #X'03
413 02A7 0A                ASLA
414 02A8 0A                ASLA
415 02A9 0A                ASLA
416 02AA AA                TAX                 ; PUT IT INTO INDEX X
417 02AB A900              LDA   #0            ; SET THE DURATION BYTE IN THE SPECIFIED
418 02AD 950F              STA   V1TBA+VXDUR,X; VOICE TO 0 TO ACTIVATE IT
419 02AF C8                INY                 ; INCREMENT THE CODE STRING POINTER
420 02B0 4C3502            JMP   PCC1          ; GO FOR NEXT COMMAND
421
422                ;       NOTE COMMANDS ARE EXPECTED HERE.  SCAN THROUGH THE VOICES IN
423                ;       SEQUENCE SKIPPING ANY INACTIVE ONES.  IF VXDUR IS ZERO, FETCH A
424                ;       COMMAND FROM THE CODE STRING WHICH IS EXPECTED TO BE A NOTE
```

NOTINT NOTRAN INTERPRETER
CODE SEGMENT INTERPRETER

```
425                ;       COMMAND AND INTERPRET IT ASSIGNING THE RESULTS TO THE CURRENT
426                ;       VOICE.  WHEN ALL VOICES HAVE BEEN PROCESSED, GO PLAY THE NOTES.
427
428 02B3 A200       NOTE:   LDX   #0            ; INITIALIZE CURRENT VOICE POINTER
429 02B5 B50F      NOTE1:  LDA   V1TBA+VXDUR,X; GET DURATION BYTE OF CURRENT VOICE
430 02B7 F00B              BEQ   NOTE2         ; JUMP IF NEWLY ACTIVATED
431 02B9 C9FF              CMP   #X'FF
432 02BB F038              BEQ   NOTE9         ; SKIP THIS VOICE IF INACTIVE
433 02BD 38                SEC                 ; SUBTRACT PREVIOUS DURATION
434 02BE E507              SBC   DUR
435 02C0 950F              STA   V1TBA+VXDUR,X; STORE RESULT BACK
436 02C2 D031              BNE   NOTE9         ; SKIP THIS VOICE IF UNEXPIRED
437 02C4 B104      NOTE2:  LDA   (CODEPT),Y    ; GET A COMMAND BYTE
438 02C6 852B              STA   CMSAVE        ; SAVE THE COMMAND BYTE
439 02C8 290F              AND   #X'0F         ; ISOLATE THE DURATION FIELD AND
440 02CA F035              BEQ   LNOTE         ; JUMP OUT IF NOT A SHORT NOTE COMMAND
441 02CC 842A              STY   YSAVE         ; SAVE Y
442 02CE A8                TAY                 ; GET DURATION VALUE CORRESPONDING TO
443 02CF B92E00            LDA   DURTAB-1,Y    ; CONTENTS OF DURATION FIELD AND
444 02D2 950F              STA   V1TBA+VXDUR,X; PUT INTO DURATION BYTE OF CURRENT VOICE
445 02D4 A52B              LDA   CMSAVE        ; GET THE COMMAND BYTE BACK
446 02D6 29F0              AND   #X'F0         ; ISOLATE THE PITCH DISPLACEMENT
447 02D8 18                CLC                 ; CONVERT TWO'S COMPLEMENT TO EXCESS 8
448 02D9 6980              ADC   #X'80
449 02DB F00A              BEQ   NOTE3         ; SKIP AHEAD IF -8 = REST
450 02DD 4A                LSRA                ; RIGHT JUSTIFY THE PITCH DISPLACEMENT
451 02DE 4A                LSRA                ; TIMES TWO
452 02DF 4A                LSRA
453 02E0 750C              ADC   V1TBA+VXNOT,X; ADD RESULT TO CURRENT PITCH BYTE
454 02E2 18                CLC
455 02E3 69F0              ADC   #-16          ; SUBTRACT OUT THE EXCESS 8 OFFSET
456 02E5 950C              STA   V1TBA+VXNOT,X; UPDATE THE CURRENT PITCH BYTE
457 02E7 A8        NOTE3:  TAY                 ; LOOKUP IN THE FREQUENCY TABLE FOR
458 02E8 B93E00            LDA   FRQTAB,Y      ; CORRESPONDING FREQUENCY PARAMETER
459 02EB 950D              STA   V1TBA+VXIN,X  ; AND PUT IT IN THE WAVE TABLE POINTER
460 02ED B93F00            LDA   FRQTAB+1,Y    ; INCREMENT FIELD OF THE CURRENT VOICE
461 02F0 950E              STA   V1TBA+VXIN+1,X
462 02F2 A42A              LDY   YSAVE         ; RESTORE SAVED Y
463 02F4 C8                INY                 ; INCREMENT Y TO POINT TO THE NEXT CODE
464                                            ; STRING ELEMENT
465 02F5 E018      NOTE9:  CPX   #24           ; TEST IF ALL VOICES HAVE BEEN SCANNED
466 02F7 F054              BEQ   PLAY          ; GO PLAY THE NOTES IF SO
467 02F9 8A                TXA                 ; BUMP THE POINTER TO THE NEXT VOICE IF NOT
468 02FA 18                CLC
469 02FB 6908              ADC   #8
470 02FD AA                TAX
471 02FE 4CB502            JMP   NOTE1         ; GO PROCESS THE NEXT VOICE
472
473                ;       PROCESS LONG NOTE COMMANDS
474
475 0301 A52B      LNOTE:  LDA   CMSAVE        ; GET FIRST BYTE OF CODE SEGMENT BACK
476 0303 29F0              AND   #X'F0         ; ISOLATE PITCH FIELD
477 0305 C960              CMP   #X'60         ; TEST IF LONG NOTE WITH ABSOLUTE PITCH
478 0307 F007              BEQ   PCC60         ; GO PROCESS IT IF SO
479 0309 C970              CMP   #X'70         ; TEST IF LONG NOTE WITH RELATIVE PITCH
```

```
 480 030B F00B              BEQ    PCC70          ; GO PROCESS IT IF SO
 481 030D 4C3302            JMP    PCC            ; GO PROCESS PURE CONTROL COMMAND
 482
 483                ;       PROCESS ABSOLUTE PITCH BYTE
 484
 485 0310 C8        PCC60:  INY                    ; GET PITCH FROM NEXT BYTE
 486 0311 B104              LDA    (CODEPT),Y
 487 0313 950C              STA    V1TBA+VXNOT,X; PUT IT IN THE FREQUENCY TABLE
 488 0315 4C2003            JMP    LNOTE3         ; DISPLACEMENT BYTE OF THE CURRENT VOICE
 489
 490                ;       PROCESS RELATIVE PITCH BYTE
 491
 492 0318 C8        PCC70:  INY                    ; GET PITCH FROM NEXT BYTE
 493 0319 B104              LDA    (CODEPT),Y
 494 031B 18                CLC                    ; ADD TO THE FREQUENCY TABLE DISPLACEMENT
 495 031C 750C              ADC    V1TBA+VXNOT,X; BYTE OF THE CURRENT VOICE
 496 031E 950C              STA    V1TBA+VXNOT,X; UPDATE THE FREQUENCY TABLE DISPLACEMENT
 497
 498                ;       GET CORRESPONDING FREQUENCY PARAMETER FROM THE NOTE FREQUENCY
 499                ;       TABLE
 500
 501 0320 842A      LNOTE3: STY    YSAVE          ; SAVE INDEX Y TEMPORARILY
 502 0322 A8                TAY                    ; LOOKUP IN FREQUENCY TABLE THE DOUBLE
 503 0323 B93E00            LDA    FRQTAB,Y       ; PRECISION FREQUENCY PARAMETER AND MOVE
 504 0326 950D              STA    V1TBA+VXIN,X ; IT TO VXIN
 505 0328 B93F00            LDA    FRQTAB+1,Y
 506 032B 950E              STA    V1TBA+VXIN+1,X
 507 032D A42A              LDY    YSAVE          ; RESTORE INDEX Y
 508
 509                ;       PROCESS WAVEFORM FIELD
 510
 511 032F C8                INY                    ; GET WAVEFORM TABLE NUMBER FROM NEXT BYTE
 512 0330 B104              LDA    (CODEPT),Y
 513 0332 4A                LSRA                   ; POSITION IT CORRECTLY
 514 0333 4A                LSRA
 515 0334 4A                LSRA
 516 0335 4A                LSRA
 517 0336 18                CLC
 518 0337 6503              ADC    WAVTBA+1       ; ADD TO PAGE NUMBER OF WAVEFORM ZERO
 519 0339 950B              STA    V1TBA+VXPTW,X; STORE IN WAVEFORM TABLE PAGE NUMBER
 520                                               ; BYTE OF THE CURRENT VOICE
 521
 522                ;       PROCESS DURATION FIELD
 523
 524 033B B104              LDA    (CODEPT),Y     ; GET THE SECOND BYTE BACK
 525 033D 290F              AND    #X'0F          ; ISOLATE THE DURATION DIGIT
 526 033F 842A              STY    YSAVE          ; SAVE Y AGAIN
 527 0341 A8                TAY
 528 0342 B92E00            LDA    DURTAB-1,Y     ; GET THE CORRESPONDING DURATION BYTE
 529 0345 950F              STA    V1TBA+VXDUR,X; STORE IT IN THE DURATION BYTE OF THE
 530                                               ; SPECIFIED VOICE
 531 0347 A42A              LDY    YSAVE          ; RESTORE Y
 532 0349 C8                INY                    ; UPDATE CODE POINTER
 533 034A 4CF502            JMP    NOTE9          ; GO UPDATE VOICE POINTER
 534
```



```
                            .PAGE  'PLAY ROUTINE'
 535                ;       PLAY THE NOTES SET UP BY THE CODE SEGMENT INTERPRETER
 536                ;       FIRST MERGE THE Y INDEX INTO THE CODE POINTER SO THAT IT POINTS
 537                ;       TO THE NEXT CODE SEGMENT AFTER THE NOTES ARE PLAYED.
 538                ;       NEXT SCAN THE DURATION FIELDS OF THE 4 VOICES TO FIND THE
 539                ;       SHORTEST.  THEN GO THROUGH THEM AGAIN AND SUBTRACT THE SHORTEST
 540                ;       DURATION FROM EACH LEAVING AT LEAST ONE OF THEM ZERO.
 541                ;       FINALLY, PLAY THE NOTES FOR THE SHORTEST DURATION AND GO BACK
 542                ;       TO THE CODE SEGMENT INTERPRETER WHEN DONE
 543
 544 034D 98        PLAY:   TYA                    ; UPDATE THE CODE POINTER IN MEMORY BY
 545 034E 18                CLC                    ; ADDING THE Y INDEX TO IT
 546 034F 6504              ADC    CODEPT
 547 0351 8504              STA    CODEPT
 548 0353 9002              BCC    PLAY1          ; INCREMENT UPPER BYTE IF CARRY FROM LOW
 549 0355 E605              INC    CODEPT+1       ; BYTE
 550
 551                ;       SCAN THE DURATION FIELDS OF ALL OF THE VOICES TO FIND THE
 552                ;       SHORTEST
 553
 554 0357 A9FF      PLAY1:  LDA    #X'FF          ; INITIALIZE SHORTEST DURATION
 555 0359 C50F              CMP    V1TBA+VXDUR    ; TEST VOICE 1 DURATION
 556 035B 9002              BCC    PLAY2          ; SKIP IF NOT LESS THAN CURRENT MIN
 557 035D A50F              LDA    V1TBA+VXDUR
 558 035F C517      PLAY2:  CMP    V2TBA+VXDUR    ; TEST VOICE 2
 559 0361 9002              BCC    PLAY3
 560 0363 A517              LDA    V2TBA+VXDUR
 561 0365 C51F      PLAY3:  CMP    V3TBA+VXDUR    ; TEST VOICE 3
 562 0367 9002              BCC    PLAY4
 563 0369 A51F              LDA    V3TBA+VXDUR
 564 036B C527      PLAY4:  CMP    V4TBA+VXDUR    ; TEST VOICE 4
 565 036D 9002              BCC    PLAY5
 566 036F A527              LDA    V4TBA+VXDUR
 567 0371 8507      PLAY5:  STA    DUR            ; SAVE SHORTEST DURATION = PLAY DURATION
 568 0373 8508              STA    DURC           ; SET DURATION COUNTER FOR SOUND GENERATE
 569
 570                ;       4 VOICE SOUND GENERATE ROUTINE
 571                ;       ENTER WITH VARIOUS TABLE POINTERS ALREADY SET UP
 572                ;       LOOPS TEMPO*DUR TIMES
 573                ;       TOTAL LOOP TIME = 114 STATES = 8770 HZ
 574
 575 0375 A000      SOUND:  LDY    #0             ; SET Y TO ZERO FOR STRAIGHT INDIRECT
 576 0377 A606              LDX    TEMPO          ; SET X TO TEMPO COUNT
 577                                               ; COMPUTE AND OUTPUT A COMPOSITE SAMPLE
 578 0379 18        SOUND1: CLC                    ; CLEAR CARRY
 579 037A B10A              LDA    (V1TBA+VXPTI),Y ; ADD UP 4 VOICE SAMPLES USING
 580 037C 7112      ADV2:   ADC    (V2TBA+VXPTI),Y ; INDIRECT ADDRESSING THROUGH VOICE
 581 037E 711A      ADV3:   ADC    (V3TBA+VXPTI),Y ; POINTERS INTO WAVEFORM TABLES
 582 0380 7122      ADV4:   ADC    (V4TBA+VXPTI),Y ; STRAIGHT INDIRECT WHEN Y INDEX = 0
 583 0382 8D0017            STA    DAC            ; SEND SUM TO DIGITAL-TO-ANALOG CONVERTER
 584 0385 A509              LDA    V1TBA+VXPTF    ; ADD INCREMENTS TO POINTERS FOR
 585 0387 650E              ADC    V1TBA+VXIN+1 ; THE 4 VOICES
 586 0389 8509              STA    V1TBA+VXPTF    ; FIRST FRACTIONAL PART
 587 038B A50A              LDA    V1TBA+VXPTI
 588 038D 650D              ADC    V1TBA+VXIN
```

NOTINT NOTRAN INTERPRETER
PLAY ROUTINE

```
 589 038F 850A              STA     V1TBA+VXPTI  ; THEN INTEGER PART
 590 0391 A511              LDA     V2TBA+VXPTF  ; DO THE SAME FOR VOICE 2
 591 0393 6516              ADC     V2TBA+VXIN+1
 592 0395 8511              STA     V2TBA+VXPTF
 593 0397 A512              LDA     V2TBA+VXPTI
 594 0399 6515              ADC     V2TBA+VXIN
 595 039B 8512              STA     V2TBA+VXPTI
 596 039D A519              LDA     V3TBA+VXPTF  ; VOICE 3
 597 039F 651E              ADC     V3TBA+VXIN+1
 598 03A1 8519              STA     V3TBA+VXPTF
 599 03A3 A51A              LDA     V3TBA+VXPTI
 600 03A5 651D              ADC     V3TBA+VXIN
 601 03A7 851A              STA     V3TBA+VXPTI
 602 03A9 A521              LDA     V4TBA+VXPTF  ; VOICE 4
 603 03AB 6526              ADC     V4TBA+VXIN+1
 604 03AD 8521              STA     V4TBA+VXPTF
 605 03AF A522              LDA     V4TBA+VXPTI
 606 03B1 6525              ADC     V4TBA+VXIN
 607 03B3 8522              STA     V4TBA+VXPTI
 608 03B5 CA                DEX                  ; DECREMENT & CHECK TEMPO COUNT
 609 03B6 D008              BNE     TIMWAS       ; BRANCH TO TIME WASTE IF NOT RUN OUT
 610 03B8 C608              DEC     DURC         ; DECREMENT & CHECK DURATION COUNTER
 611 03BA F00C              BEQ     JPCC         ; JUMP OUT IF END OF NOTE
 612 03BC A606              LDX     TEMPO        ; RESTORE TEMPO COUNT
 613 03BE D0B9              BNE     SOUND1       ; CONTINUE PLAYING
 614 03C0 D000  TIMWAS:     BNE     TMWS1        ; 3  WASTE 12 STATES
 615 03C2 D000  TMWS1:      BNE     TMWS2        ; 3
 616 03C4 D000  TMWS2:      BNE     TMWS3        ; 3
 617 03C6 D0B1  TMWS3:      BNE     SOUND1       ; 3  CONTINUE PLAYING
 618
 619 03C8 4C3302  JPCC:     JMP     PCC          ; GO BACK TO INTERPRETER
 620
```

NOTINT NOTRAN INTERPRETER
MISCELLANEOUS SUBROUTINES

```
                           .PAGE  'MISCELLANEOUS SUBROUTINES'
 621                ;       SET NUMBER OF VOICES SUBROUTINE
 622                ;       THIS ROUTINE MODIFIES INSTRUCTIONS IN THE SOUND GENERATE
 623                ;       ROUTINE TO CONTROL THE NUMBER OF VOICES INCLUDED IN THE SAMPLE
 624                ;       CALCULATION.  ENTER WITH NUMBER OF VOICES TO INCLUDE IN A,
 625                ;       RANGE IS 1 TO 4.  0 IS INTERPRETED AS 1. USES INDEX X
 626
 627 03CB AA        SETNV:  TAX
 628 03CC A912              LDA     #V2TBA+VXPTI ; FIRST SET TO INCLUDE ALL 4 VOICES
 629 03CE 8D7D03            STA     ADV2+1
 630 03D1 A91A              LDA     #V3TBA+VXPTI
 631 03D3 8D7F03            STA     ADV3+1
 632 03D6 A922              LDA     #V4TBA+VXPTI
 633 03D8 8D8103            STA     ADV4+1
 634 03DB A92D              LDA     #NULSMP      ; SET UP TO ELIMINATE VOICES
 635 03DD E002              CPX     #2           ; TEST ARGUMENT TO DETERMINE HOW MANY TO
 636 03DF F00A              BEQ     ELIM3        ; ELIMINATE
 637 03E1 3005              BMI     ELIM2
 638 03E3 E003              CPX     #3
 639 03E5 F007              BEQ     ELIM4
 640 03E7 60                RTS
 641 03E8 8D7D03  ELIM2:    STA     ADV2+1       ; ELIMINATE REQUIRED NUMBER OF VOICES
 642 03EB 8D7F03  ELIM3:    STA     ADV3+1
 643 03EE 8D8103  ELIM4:    STA     ADV4+1
 644 03F1 60                RTS
 645
 646 0000                   .END
NO ERROR LINES
```

```
*
*  KIM-1 NOTRAN DEMONSTRATION SCORE
*  PURPOSE IS TO DEMONSTRATE NOTRAN
*  CODING TECHNIQUES AND CAPABILITIES.
*  NOTE THAT A VARIETY OF STATEMENT
*  FORMATS ARE USED AND THAT REDUNDANT
*  INFORMATION IS OFTEN GIVEN FOR
*  INSTRUCTIONAL PURPOSES.
*
*  SETUP FOR 4 VOICES MAXIMUM, ACTIVATE
*  VOICE 1; ASSIGN IT TO WAVEFORM 1,
*  AND SET A MODERATE TEMPO
   NVC 4;  ACT 1;  WAV 1,1;  TPO 120
262F  50 04 90 00 10 78
*
*  DEMONSTRATION 1  SIMPLE 1 VOICE SCALE
   1C4H; 1D4Q; E4Q; FQ; GQ; AQ; BQ; C5Q
2635  60 4A 03 26 26 16 26 26 26 16
   B4Q; AQ; GQ; FQ; EQ; DQ; CH.
263F  F6 E6 E6 E6 F6 E6 E2
   1RQ
2646  86
   CH; C#Q; DQ; E@Q; EQ; FQ; F#Q; GQ; G#Q
2647  03 16 16 16 16 16 16 16 16
   AQ; B@Q; C@Q; C5H
2650  16 16 16 13
   B4Q; A#Q; AQ; G#Q; GQ; F#Q; FQ; EQ; D#Q
2654  F6 F6 F6 F6 F6 F6 F6 F6 F6
   DQ; C#Q; CH.; RH
265D  F6 F6 F2 83
*
*  DEMONSTRATION 2  2 VOICE SCALE WITH
*  EIGHTH NOTES AGAINST QUARTER NOTES
*  TO DEMONSTRATE SIMPLE OVERLAYS
*  ACTIVATE VOICE 2 AND ASSIGN IT TO
*  WAVEFORM 1; REASSIGN VOICE 1 TO WAVE 3
   ACT 2; WAV 3,1; WAV 1,2
2661  90 01
   1C2Q;  2C4E
2663  60 1A 26 60 4A 09
         2C4E
2669  09
   1D2Q;  2D4E
266A  26 29
         2D4E
266C  09
   EQ; EE
266D  26 29
        EE
266F  09
   FQ; FE; FE
2670  16 19 09
   GQ; GE; GE;  AQ; AE; AE;   BQ; BE; BE
2673  26 29 09 26 29 09 26 29 09
   1C3Q; 2C5E; 2C5E
267C  16 19 09
   B2Q; B4E; BE
267F  F6 F9 09
   AQ; AE; AE;   GQ; GE; GE;   FQ; FE; FE
2682  E6 E9 09 E6 E9 09 E6 E9 09
   EQ; EE; EE;   DQ; DE; DE;   1CH; 2CQ; CQ
268B  F6 F9 09 E6 E9 09 E3 E6 06
   1RH; 2RH
2694  83 83
```



```
*
*  DEMONSTRATION 3  4 VOICE SCALE WITH EIGHTH
*  NOTE TRIPLETS AND EIGHTH NOTES IN THE TREBLE
*  AGAINST QUARTER NOTE CHORDS IN THE BASS
*  ACTIVATE VOICES 3 AND 4 AND ASSIGN THEM TO
*  WAVE 1
*  REASSIGN VOICES 1 AND 2 TO WAVE 3 FOR RICH BASS
   ACT 3,4; WAV 1,3; WAV 1,4; WAV 3,1; WAV 3,2
2696  90 02 90 03
   1C2Q;  2G2Q;  3C4E;  4C5E3
269A  60 1A 26 60 28 26 60 4A 09 60 62 0B
                        4C5E3
26A6  0B
                  3C4E
26A7  09
                        4C5E3
26A8  0B
   1D2Q;  2A2Q;  DE;     DE3
26A9 26 26 29 2B
                        DE3
26AD  0B
                  DE
26AE  09
                        DE3
26AF  0B
   EQ;    BQ;    EE;    EE3
26B0  26 26 29B
                        EE3
26B4  0B
                  EE
26B5  09
                        EE3
26B6  0B
   FQ;    C3Q;   FE;    FE3
26B7  16 16 19 1B
                        FE3
26BB  0B
                  FE
26BC  09
                        FE3
26BD  0B
   GQ;    DQ;    GE;    GE3
26BE  26 26 29 2B
                        GE3
26C2  0B
                  GE
26C3  09
                        GE3
26C4  0B
   AQ;    EQ;    AE;    AE3
26C5  26 26 29 2B
                        AE3
26C9  0B
                  AE
26CA  09
                        AE3
26CB  0B
   BQ;    F#Q;   BE;    BE3
26CC  26 26 29 2B
                        BE3
26D0  0B
                  BE
26D1  09
                        BE3
26D2  0B
   1C3H;  2G3H;  3C5Q;  4C6Q3
26D3  13 13 16 18
                        CQ3
26D7  08
                CQ
26D8  06
                        CQ3
26D9  08
   CH;    GH;    CH;    CH
26DA  03 03 03 03
   RH;    RH;    RH;    RH
26DE  83 83 83 83
*
*  DEMONSTRATION 4  ARPEGGIO DEMONSTRATION
*  ASSIGN WAVEFORM 2 TO ALL 4 VOICES
   WAV 2,1;  WAV 2,2;  WAV 2,3;  WAV 2,4
26E2
   1C4Q.; 2RS;   3RE;   4RE.
26E2  60 4A 14 8C 89 87
         2E4S
26E8  60 52 1C
         2E4Q;  3G4Q
26EB  06 B6
                        4C5E.
26ED  60 62 17
   1C4Q.; 2RS;   3RE;   4RE.
26F0  04 8C 89 87
         2F4S
26F4  1C
         2F4Q;  3A4Q
26F5  06 26
                        4C5E.
26F7  07
   1C4Q.; 2RS;   3RE;   4RE.
26F8  04 8C 89 87
         2E4S
26FC  FC
         2E4Q;  3G4Q
26FD  06 E6
                        4C5E.
26FF  07
   1B3Q.; 2RS;   3RE;   4RE.
2700  F4 8C 89 87
         2D4S
2704  EC
         2D4Q;  3G4Q
2705  06 06
                        4B4E.
2707  F7
   1C4H.; 2RE;   3RQ;   4RQ.
2708  12 89 86 84
         2E4E
270C  29
         2E4H;  3G4H
270D  03 03
                        4C5Q.
270F  14
   1RH;   2RH;   3RH;   4RH
2710  83 83 83 83
```

```
*
*  DEMONSTRATION 5  "LEAPFROG" EFFECT
*  AND DEMONSTRATION OF PITCH RANGE
*  LEAVE VOICE DEFINITIONS ALONE
   1C1W;  2RQ;   3RH;   4RH.
2714  60 02 11 86 83 82
         2E1W
271A  60 0A 11
                  3G1W
271D  60 10 11
                        4C2W
2720  60 1A 11
   1E2W
2723  60 22 11
         2G2W
2726  60 28 11
                  3C3W
2729  60 32 11
                        4E3W
272C  60 3A 11
   1G3W
272F  60 40 11
         2C4W
2732  60 4A 11
                  3E4W
2735  60 52 11
                        4G4W
2738  60 58 11
   1C5W
273B  60 62 11
         2E5W
273E  60 6A 11
                  3G5W
2741  60 70 11
                        4C6W
2744  60 7A 11
   1C6W;  2C6W;  3C6W;  4G5W
2747  60 7A 11 60 7A 11 51 B1
   1E5W;  2C5W;  3G4W;  4E4W
274F  60 6A 11 60 62 11 60 58 11 60 52 11
   1C4W;  2G3W;  3E3W;  4C3W
275B  60 4A 11 60 40 11 60 3A 11 60 32 11
   1G2W;  2E2W;  3C2W;  4G1W
2767  60 28 11 60 22 11 60 1A 11 60 10 11
   1E1W;  2C1W;  3RH.;  4RH
2773  60 0A 11 60 02 11 82 83
   1RQ;
277B  86 ER-06
   1RH;   2RH;   3RH;   4RH
277C  83 83 83 83
```

```
*
*  DEMONSTRATION 6  TEMPO CHANGE
*  AND MORE COMPLEX MUSIC
   WAV 1,1 WAV 1,2 WAV 1,3 WAV 1,4
2780
   TPO 150
2780  10 96
   1F3T; 2RQ; 3RQ; 4RQ
2782  60 3C 0F 86 86 86
   1B@T; 1D4T; 1FT
2788  5F 4F 3F
   1B@E
278B  59
   1B@E; 2F4E; 3D4E; 4F3E
278C  09 60 54 19 60 4E 19 60 3C 19
   1B@E; 2GE;  3EE;  4EE
2796  09 29 29 F9
   TPO 160
279A  10 A0
   1E3T; 2RQ;  3RQ;  4RQ
279C  60 3A 0F 86 86 86
   1GT;  1B@T; 1C#4T;1B@E
27A2  3F 3F 3F 60 5E 09
   TPO 170
27A8  10 AA
   1B@E; 2GE;  3EE;  4C#4E
27AA  09 09 09 60 4C 19
   1AE;  2F#E; 3DE;  4F#3E
27B0  F9 F9 E9 99
   TPO 185
27B4  10 B9
   1D3T; 2RQ;  3RQ;  4RQ
27B6  60 36 0F 86 86 86
   1AT;  TPO 190; 1D4T; TPO 195
27BC  7F 10 BE 5F 10 C3
   1F#4T; TPO 200; 1A4E
27C2  4F 10 C8 39
   1A4Q.;2F#4E;3D4E; 4F#3E
27C6  04 09 09 09
   TPO 210
27CA  10 D2
         2D4H; 3B@3H;4G3H
27CC  C3 C3 13
   1G4H
27CF  E3
         2E4Q; 3C#4Q;4B@3Q
27D0  26 36 36
   1RQ;  2F4Q; 3D4H; 4A#Q
27D3  86 16 13 06
   1RW;  2E4H;       4A3H
27D7  81 F3 F3
                  3C4H
27DA  E3
         2D4H;       4F3Q
27DB  E3 C6
                  3B@3Q;4G3Q
27DD  E6 26
   TPO 250
27DF  10 FA
   1D4W; 2A3W; 3F3W; 4D3W
27E1  B1 B1 B1 B1
   1RW;  2RW;  3RW;  4RW
27E5  81 81 81 81
```

```
*
*  DEMONSTRATION 7  USE OF NVC STATEMENT
*  SET NUMBER OF VOICES TO GENERATE TO 1
   NVC 1
27E9  50 01
*  ASSIGN THE VOICE TO WAVE 4 WHICH IS A
*  FULL AMPLITUDE WAVE 4 TIMES LOUDER THAN
*  THE WAVES PREVIOUSLY USED
   WAV 4,1
27EB
*  DEACTIVATE THE OTHER 3 VOICES (STUPID PROGRAM!)
   DCT 2,3,4
27EB  80 01 80 02 80 03
   TPO 80
27F1  10 50
*  PLAY A HARD DRIVING BASS LINE
   D2Q; D2Q; F2E; E2E; D2E; E@2Q
27F3  60 1E 36 06 39 F9 E9 16
   E@3Q; D3Q; D@3Q; B2E
27FB  60 38 36 F6 F6 E9
   D2Q; D2Q; F2E; E2E; D2E; E@2H
2801  60 1E 36 06 39 F9 E9 13
   RW
2809  81
*
*  DEMONSTRATION 8   WAVEFORM CHANGE
*  FIRST GET THINGS BACK TO NORMAL
   NVC 4
280A  50 04
   TPO 160
280C  10 A0
   WAV 1,1; C3S; WAV 2,1; C3S
280E  60 32 0C 60 32 1C
   WAV 3,1; C3S; WAV 2,1; C3S
2814  60 32 2C 60 32 1C
   WAV 1,1; E3S; WAV 2,1; E3S
281A  60 3A 0C 60 3A 1C
   WAV 3,1; E3S; WAV 2,1; E3S
2820  60 3A 2C 60 3A 1C
   WAV 1,1; G3S; WAV 2,1; G3S
2826  60 40 0C 60 40 1C
   WAV 3,1; G3S; WAV 2,1; G3S
282C  60 40 2C 60 40 1C
   WAV 1,1; C4S; WAV 2,1; C4S
2832  60 4A 0C 60 4A 1C
   WAV 3,1; C4S; WAV 2,1; C4S
2838  60 4A 2C 60 4A 1C
   RH
283E 83
*
*  DEMONSTRATION 9  MUSICAL SUBROUTINE USAGE
*  DEFINE SUBROUTINE AREA
   SUB
283F  40 00 00
*  SHOULD GENERALLY FORCE ABSOLUTE PITCH
*  AT SUBROUTINE ENTRIES
10 ABS
2842
   C5Q; C5Q; G5Q; G5Q; A5Q; A5Q; G5H
2842  60 62 16 06 76 06 26 06 E3
   RTS
284B  30
*
20 ABS
284C
```

```
   F5Q; F5Q; E5Q; E5Q; D5Q; D5Q; C5H
284C  60 6C 16 06 F6 06 E6 06 E3
   RTS
2855  30
*
30 ABS
2856
   G5Q; G5Q; F5Q; F5Q; E5Q; E5Q; D5H
2856  60 70 16 06 E6 06 F6 06 E3
   RTS
285F  30
*  END SUBROUTINE AREA
   ESB
2860
*  NOW SETUP VOICES AND PLAY "TWINKLE STAR"
*  AS A SERIES OF SUBROUTINE CALLS
   WAV 2,1; TPO 80
2860  10 50
   JSR 10; JSR 20; JSR 30; JSR 30
2862  20 13 02 20 1D 02 20 27 02 20 27 02
   JSR 10; JSR 20
286E  20 13 02 20 1D 02
   RW
2874  81
*
*  DEMONSTRATION 10  INFINITE LOOP WITH RELATIVE
*  PITCH CAUSING ASCENDING CHORD SEQUENCE
   ACT 1,2,3,4; WAV 1,1; WAV 1,2; WAV 1,3; WAV 1,4
2875  90 00 90 01 90 02 90 03
   TPO 100
287D  10 64
   1C2Q; 2E2Q; 3G2Q; 4C3Q
287F  60 1A 06 60 22 06 60 28 06 60 32 06
50 1C2Q; 2F2Q; 3A2Q; 4C3Q
288B  06 16 26 06
   1C2Q; 2E2Q; 3G2Q; 4C3Q
288F  06 F6 E6 06
   1B1Q; 2D2Q; 3G2Q; 4B2Q
2893  F6 E6 06 F6
   1C2H; 2E2H; 3G2H; 4C3H
2897  13 23 03 13
*  ALL NOTES UP 1/2 STEP HERE
   1C#2Q;2F2Q; 3G#2Q;4C#3Q
289B  16 16 16 16
   JMP 50
289F  40 5C 02
*  THE CHORDS WILL EVENTUALLY SLIDE OFF THE UPPER
*  END OF THE PITCH TABLE AND CREATE SOME
*  ODDBALL SOUNDS BUT WILL WRAPAROUND
*  AND BECOME A CHORD PROGRESSION AGAIN.
*
   END
28A2  00
```

```
                        .PAGE  'DOCUMENTATION'
 3              ;       COPYRIGHT 1977 BY MICRO TECHNOLOGY UNLIMITED, BOX 4596,
 4              ;       MANCHESTER, NEW HAMPSHIRE 03108
 5              ;       PROGRAM WRITTEN BY HAL CHAMBERLIN
 6
 7              ;       THIS PROGRAM COMPILES A SIMPLIFIED SUBSET OF THE NOTRAN MUSIC
 8              ;       LANGUAGE INTO AN OBJECT CODE STRING FOR PLAYING BY THE KIM
 9              ;       NOTRAN INTERPRETER.
10
11              ;       LINES OF NOTRAN SOURCE CODE ARE PROCESSED BY THE COMPILER AND
12              ;       CONVERTED INTO A SERIES OF OBJECT CODE BYTES WHICH ARE STORED
13              ;       IN MEMORY AND LINES OF OBJECT CODE LISTING WHICH ARE PRINTED BY
14              ;       AN OUTPUT DEVICE.  IF A CONTROLLABLE INPUT DEVICE IS AVAILABLE
15              ;       (ONE THAT CAN BE STARTED AND STOPPED UNDER PROGRAM CONTROL)
16              ;       THEN ONLY ONE LINE OF SOURCE CODE NEEDS TO BE IN MEMORY AT
17              ;       ONCE.  STORAGE SPACE IS ALSO REQUIRED FOR A SMALL SYMBOL TABLE
18              ;       WHICH NEEDS 3 BYTES PER SYMBOL.  SEE THE PROGRAM LISTING FOR
19              ;       ADDRESS VECTORS AND SIZES OF THE SYMBOL TABLE, OBJECT CODE
20              ;       AREA, LINE INPUT BUFFER, AND LINE OUTPUT BUFFER.
21
22              ;       NOTRAN SOURCE IS MADE UP OF LINES.  A LINE MAY HAVE AN OPTIONAL
23              ;       IDENTIFIER AND MAY CONSIST OF ONE OR MORE SPECIFICATIONS.  THE
24              ;       IDENTIFIER, IF PRESENT, MUST BE IN THE FIRST CHARACTER POSITION
25              ;       OF THE LINE.  AN IDENTIFIER IS A UNIQUE NUMBER BETWEEN 1 AND
26              ;       255.  IF THERE IS NO IDENTIFIER, THE FIRST CHARACTER POSITION
27              ;       MUST BE BLANK.  IF THE FIRST CHARACTER OF A LINE IS AN *, THEN
28              ;       THE ENTIRE LINE IS TREATED AS A COMMENT.
29
30              ;       A SPECIFICATION MAY EITHER BE A KEYWORD SPECIFICATION OR A NOTE
31              ;       SPECIFICATION.  A KEYWORD SPECIFICATION MAY BE EITHER
32              ;       EXECUTABLE IF OBJECT CODE BYTES ARE GENERATED FOR IT OR
33              ;       NON-EXECUTABLE IF IT ONLY INFLUENCES FUTURE COMPILER ACTIONS. A
34              ;       KEYWORD SPECIFICATION CONSISTS OF A THREE LETTER KEYWORD
35              ;       FOLLOWED BY OPTIONAL PARAMETERS SEPARATED BY COMMAS.  A NOTE
36              ;       SPECIFICATION CONSISTS OF A STRING OF CHARACTERS TO BE DEFINED
37              ;       LATER.  MULTIPLE SPECIFICATIONS ON ONE LINE MUST BE SEPARATED
38              ;       BY SEMICOLONS.  ANY NUMBER OF BLANKS (INCLUDING ZERO) MAY BE
39              ;       INSERTED BEFORE AND AFTER SEMICOLONS, BETWEEN A KEYWORD AND ITS
40              ;       PARAMETERS, AND BEFORE THE FIRST SPECIFICATION ON A LINE.
41              ;       BLANKS MAY NOT APPEAR IN THE MIDDLE OF A NOTE SPECIFICATION.
42              ;       MAXIMUM INPUT LINE LENGTH IS 72 CHARACTERS.
43
44              ;       DUE TO THE ONE-PASS SIMPLIFIED NATURE OF THE COMPILER, FORWARD
45              ;       REFERENCING IN JUMP AND JUMP TO SUBROUTINE STATEMENTS IS NOT
46              ;       SUPPORTED.  A BLOCK OF STATEMENTS MAY HOWEVER BE DEFINED AND A
47              ;       JUMP CREATED TO SKIP FORWARD OVER IT.  THE BEGINNING OF THE
48              ;       BLOCK IS DEFINED BY A "SUB" STATEMENT AND THE END IS DEFINED BY
49              ;       AN "ESB" STATEMENT.  A JUMP IS CREATED AT THE "SUB" STATEMENT
50              ;       WHICH IS DIRECTED TO THE STATEMENT FOLLOWING THE "ESB".  THIS
51              ;       IN EFFECT DEFINES A FORWARD REFERENCE JUMP THAT MAY BE USED TO
52              ;       JUMP OVER SUBROUTINE DEFINITIONS.  ONLY ONE "SUB-ESB" PAIR MAY
53              ;       BE ACTIVE AT A TIME.
54
55              ;       THE KEYWORD SPECIFICATIONS RECOGNIZED ARE AS FOLLOWS (X IS A
56              ;       NUMERICAL PARAMETER):
```



```
57              ;
58              ;               NVC X           DEFINE NUMBER OF VOICES TO MIX, X IS IN
59              ;                               RANGE IF 1-4 (SEE INTERPRETER LISTING)
60              ;                               (EXECUTABLE)
61              ;               ACT X,X,..,X    ACTIVATE VOICE(S) X, X IN RANGE OF 1-4.
62              ;                               (EXECUTABLE)
63              ;               DCT X,X,..,X    DEACTIVATE VOICE(S) X  (EXECUTABLE)
64              ;               WAV X,Y         ASSIGN WAVEFORM X TO VOICE Y. X IN RANGE OF
65              ;                               1-16, Y IN RANGE OF 1-4
66              ;                               (NOTE: CORRESPONDANCE BETWEEN WAVEFORM
67              ;                               NUMBERS AND ACTUAL WAVEFORMS DEFINED WHEN
68              ;                               THE INTERPRETER IS RUN)
69              ;                               (NON-EXECUTABLE)
70              ;               TPO X           SET TEMPO TO X, X IN RANGE OF 1-255
71              ;                               QUARTER NOTE DURATION (MILLISECONDS) =
72              ;                               X*5.472  (EXECUTABLE)
73              ;               ABS             FORCES ABSOLUTE PITCH IN OBJECT CODE OF THE
74              ;                               NEXT NOTE SOUNDED BY EACH VOICE
75              ;                               (NON-EXECUTABLE)
76              ;               JMP X           UNCONDITIONAL JUMP TO THE LINE WHOSE
77              ;                               IDENTIFIER MATCHES X WHILE PLAYING.
78              ;                               (EXECUTABLE)
79              ;               JSR X           JUMP TO MUSICAL SUBROUTINE TO THE LINE
80              ;                               WHOSE IDENTIFIER MATCHES X WHILE PLAYING.
81              ;                               (EXECUTABLE)
82              ;               RTS             RETURN FROM MUSICAL SUBROUTINE (EXECUTABLE)
83              ;               SUB             DEFINE BEGINNING OF SKIPPED STATEMENT BLOCK
84              ;                               (EXECUTABLE)
85              ;               ESB             DEFINE END OF SKIPPED STATEMENT BLOCK
86              ;                               (EXECUTABLE)
87              ;               END             END OF NOTRAN SOURCE (EXECUTABLE)
88
89              ;       NOTE SPECIFICATIONS HAVE THE FOLLOWING FORMAT:
90
91              ;       (OPTIONAL VOICE DIGIT)(NOTE LETTER)(OPTIONAL # OR @)(OPTIONAL
92              ;       OCTAVE NUMBER)(DURATION LETTER)(OPTIONAL DURATION MODIFIER)
93
94              ;       THE INDIVIDUAL FIELDS ARE AS FOLLOWS:
95
96              ;       OPTIONAL VOICE DIGIT  A DIGIT IN THE RANGE OF 1 TO 4. DOES NOT
97              ;                             REALLY DO ANYTHING BUT WHEN PRESENT IS
98              ;                             COMPARED WITH THE VOICE NUMBER THAT THE
99              ;                             INTERPRETER WOULD ACTUALLY ASSIGN TO THIS
100             ;                             NOTE AND AN ERROR IS SIGNALLED IF THERE
101             ;                             IS A MISMATCH.
102             ;       NOTE LETTER           A, B, C, D, E, F, G  ARE LEGAL
103             ;       OPTIONAL # OR @       # FOR A HALF-STEP INCREMENT IN PITCH
104             ;                             @ FOR A HALF-STEP DECREMENT IN PITCH
105             ;       OPTIONAL OCTAVE NUMBER  IF PRESENT SPECIFIES THE OCTAVE, RANGE
106             ;                             IS 1-6, C4 IS MIDDLE C, C6 IS HIGHEST
107             ;                             ALLOWABLE NOTE, AN OCTAVE SPANS FROM C TO
108             ;                             B.  IF NOT PRESENT, THE OCTAVE LAST
109             ;                             PLAYED BY THE VOICE IS USED.
110             ;       DURATION LETTER       W = WHOLE, H = HALF, Q = QUARTER,
111             ;                             E = EIGHTH, S = SIXTEENTH, T = THIRTY-
```

```
                                SECOND
OPTIONAL DURATION MODIFIER  . SPECIFIES 3/2 OF THE SPECIFIED
                              DURATION, 3 SPECIFIES 2/3 OF THE
                              SPECIFIED DURATION.  W.  W3  T3  NOT
                              AVAILABLE.

                    RULES FOR ASSIGNING NOTES TO VOICES

FOR PURPOSES OF THE KIM MUSIC SYSTEM, MUSIC IS BROKEN UP INTO A
SERIES OF EVENTS.  AN EVENT IS DEFINED AS THE INTERVAL BETWEEN
*ANY* CHANGE IN THE NOTES OF A CHORD BEING PLAYED.  THE
ORIGINAL FORM OF THE MUSIC SYSTEM REQUIRED 5 BYTES FOR EVERY
EVENT EVEN IF ONLY ONE NOTE OF A 4 NOTE CHORD CHANGED BETWEEN
TWO SUCCESSIVE EVENTS.  LETS ASSUME FOR THE MONENT THE
EXISTENCE OF A LONG STRING OF EVENTS WITH NO CONTROL
SPECIFICATIONS (TEMPO CHANGE, ETC.) INTERSPERSED.  AT THE
CONCLUSION OF THE PREVIOUS EVENT THE INTERPRETER WILL FETCH
NOTE CODE BYTES FROM THE CODE STRING ONE-AT-A-TIME TO BUILD UP
4 NOTES FOR THE NEXT EVENT.  HOWEVER IF, AS IS OFTEN THE CASE
IN MUSIC, THE NOTES OF THE PREVIOUS EVENT WERE OF DIFFERENT
DURATIONS, ONLY THE CHANGED NOTES NEED TO BE FETCHED FROM THE
CODE STRING.  THE "UNEXPIRED" NOTES FROM THE PREVIOUS EVENT CAN
BE CARRIED OVER TO THE CURRENT EVENT.  THE RULE FOR ASSIGNING
VOICE NUMBERS (1, 2, 3, OR 4 WHICH HAVE BEEN EQUATED WITH
DIFFERENT WAVEFORMS) IS THAT THEY ARE ASSIGNED TO CODE BYTES
SEQUENTIALLY STARTING WITH VOICE 1.  HOWEVER, IF VOICE 1 IS
BEING CARRIED OVER FROM THE PREVIOUS EVENT THEN IT IS SKIPPED
AND VOICE 2 IS TRIED.  THIS PROCESS, WHICH IS CALLED "EVENT
BUILDING" CONTINUES UNTIL ALL "EXPIRED" VOICES ARE ASSIGNED NEW
NOTES FROM THE CODE STRING.  IF A VOICE IS TO BE SILENT FOR THE
EVENT, A REST SPECIFICATION IS PLACED IN THE CODE STRING.  IF A
VOICE IS TO BE SILENT FOR AN EXTENDED PERIOD OF TIME, IT MAY BE
"DEACTIVATED".  INACTIVE VOICES ARE SKIPPED WHEN NOTES ARE
ASSIGNED TO VOICES.

AFTER THE NOTES ARE ASSIGNED, THE EVENT LENGTH IS COMPUTED.
THE EVENT LENGTH IS SIMPLY THE DURATION OF THE VOICE WITH THE
SHORTEST REMAINING DURATION.  THE INTERPRETER THEN PLAYS THE
CHORD THAT WAS SET UP FOR THIS SHORTEST DURATION AND THEN LOOPS
BACK TO SET UP FOR THE NEXT EVENT.

CONTROL SPECIFICATIONS SUCH AS A TEMPO CHANGE SHOULD ONLY OCCUR
BETWEEN THE GROUPS OF NOTE CODES THAT COMPRISE AN EVENT.  IF
THE INTERPRETER SEES SUCH A CONTROL COMMAND WHILE BUILDING AN
EVENT, IT ABORTS THE PARTIALLY BUILT EVENT, PROCESSES THE
CONTROL COMMAND, AND THEN STARTS ANEW BUILDING AN EVENT.  THE
COMPILER ALERTS THE USER THROUGH AN ERROR MESSAGE IF AN ATTEMPT
HAS BEEN MADE TO INSERT A CONTROL COMMAND IN THE MIDDLE OF AN
EVENT GROUP.  THE SAME IS TRUE IF AN IDENTIFIER IS PLACED ON A
NOTE SPECIFICATION THAT IS IN THE MIDDLE OF AN EVENT GROUP.

IN ORDER TO CONSERVE SPACE IN THE OBJECT CODE FORMAT AN
EXPLICIT VOICE NUMBER SPECIFICATION IS NOT INCLUDED IN THE NOTE
CODES.  THE ASSIGNMENT OF VOICES TO NOTE CODES IS PERFORMED BY
THE NOTRAN OBJECT CODE INTERPRETER.  IN ORDER TO MINIMIZE
```


NOTCMP KIM NOTRAN COMPILER
DOCUMENTATION

```
CODING ERRORS BY THE NOTRAN USER, THE COMPILER DUPLICATES THE
ASSIGNMENT ACTIONS OF THE INTERPRETER AND SIGNALS ANY MISMATCH
BETWEEN THE VOICE THAT WOULD BE ASSIGNED BY THE INTERPRETER AND
THE VOICE INTENDED BY THE USER.

ACTUALLY THIS COMPILER IS MORE LIKE AN ASSEMBLER IN THAT NOTE
AND CONTROL SPECIFICATIONS ARE TRANSLATED AND PASSED
ONE-FOR-ONE INTO OBJECT CODE FOR THE NOTRAN INTERPRETER.
CERTAINLY A MORE SOPHISTICATED COMPILER IS POSSIBLE WHICH, FOR
EXAMPLE, MIGHT ALLOW THE USER TO SPECIFY THE MUSIC
"HORIZONTALLY" ONE VOICE LINE AT A TIME RATHER THAN
"VERTICALLY" ONE CHORD AT A TIME.

ERROR MESSAGES ARE AS FOLLOWS:

ER-01  ARGUMENT OUT OF RANGE, EITHER TOO BIG OR TOO SMALL
ER-02  UNDEFINED IDENTIFIER IN A JMP OR JSR SPECIFICATION
ER-03  IDENTIFIER HAS ALREADY BEEN USED, IT IS IGNORED
ER-04  SYMBOL TABLE OVERFLOW, THE IDENTIFIER IS IGNORED
ER-05  OBJECT CODE OVERFLOW, THE OBJECT CODE BYTE IS NOT STORED
ER-06  INCOMPRENENSIBLE SPECIFICATION, SKIP TO NEXT
       SPECIFICATION OR STATEMENT
ER-07  SPECIFIED VOICE NUMBER AND ASSUMED VOICE NUMBER DO NOT
       AGREE, ASSUMED VOICE NUMBER IS USED
ER-08  NOTE PITCH OUT OF RANGE, EITHER TOO HIGH OR TOO LOW OR
       OCTAVE NUMBER OMITTED AND VOICE HAS NOT BEEN PREVIOUSLY
       SOUNDED, USE HIGHEST AVAILABLE PITCH
ER-09  ILLEGAL DURATION, NOTE IS SKIPPED
ER-10  EXECUTABLE CONTROL SPECIFICATION IN MIDDLE OF EVENT
       BUILD, PARTIAL EVENT IS SKIPPED AND CONTROL SPEC IS
       EXECUTED.
ER-11  IDENTIFIER OCCURS IN THE MIDDLE OF AN EVENT, THE
       IDENTIFIER IS PROCESSED.
ER-12  ATTEMPT MADE TO NEST SUB-ESB PAIR, SPECIFICATION IGNORED
ER-13  ESB WITHOUT A MATCHING SUB SPECIFICATION, THE ESB IS
       IGNORED
ER-14  HANGING SUB AT END OF SONG
ER-15  NOTES ENCOUNTERED BUT NO VOICES ARE ACTIVE, COMPILATION
       ABORTED

SINCE THE COMPILER IS NOT VERY SOPHISTICATED, OCCASIONALLY AN
ERROR MESSAGE MAY BE INAPPROPRIATE OR SEPARATED FROM THE ACTUAL
ERROR LOCATION.  BE ON THE LOOKOUT FOR ANYTHING WHEN THE ERROR
CODE IS 1, 6, 8, OR 9.
```

```
                                .PAGE  'CONSTANTS AND DATA'
  213 0000                      .=     0            ; KEEP ALL CONSTANTS AND DATA IN PAGE 0
  214
  215 1C22          KIMMON      =      X'1C22       ; ENTRY POINT TO KIM MONITOR
  216 1C00          KIMSAV      =      X'1C00       ; SAVE STATUS ENTRY POINT TO KIM MONITOR
  217 1E5A          KIMICH      =      X'1E5A       ; KIM TTY INPUT A CHARACTER INTO A, KEEPS X
  218 1EA0          KIMOCH      =      X'1EA0       ; KIM TTY OUTPUT A CHAR FROM A, KEEPS X
  219 1ED4          KIMDLY      =      X'1ED4       ; KIM TTY DELAY ONE BIT TIME, KEEPS X
  220
  221               ;           NOTE: THESE ARE PARAMETERS THAT MUST BE SET UP BY THE USER
  222               ;           ACCORDING TO THE SYSTEM MEMORY CONFIGURATION
  223               ;           DEFAULT SETTINGS SHOWN FOR 4K EXPANSION STARTING AT 2000 (HEX)
  224               ;           AND USE OF DEFAULT I/O ROUTINES
  225
  226 0000 0003     INBFA:      .WORD  X'300        ; INPUT BUFFER ADDRESS, 73 BYTES USED
  227 0002 4903     OUBFA:      .WORD  X'300+73     ; OUTPUT BUFFER ADDRESS, 74 BYTES USED
  228 0004 9303     SYMTA:      .WORD  X'300+73+74  ; SYMBOL TABLE ADDRESS
  229 0006 2F26     CODEA:      .WORD  CMPEND       ; OBJECT CODE AREA ADDRESS
  230 0008 D009     CODELN:     .WORD  X'2FFF-CMPEND; MAXIMUM NUMBER OF MUSIC OBJECT CODE BYTES
  231                                              ; ALLOWED
  232 000A 21       SYMTLN:     .BYTE  100-1/3      ; MAXIMUM NUMBER OF SYMBOLS ALLOWED, 3
  233                                              ; BYTES OF MEMORY USED FOR EACH SYMBOL
  234                                              ; PLUS 1 FOR AN END MARK
  235
  236               ;           THESE ADDRESSES MAY BE ALTERED BY THE USER TO USE ALTERNATE I/O
  237               ;           ROUTINES IF A TELETYPE IS NOT AVAILABLE
  238
  239 000B 0102     INP:        .WORD  DFIN         ; INPUT ROUTINE ADDRESS
  240 000D 0002     INPINT:     .WORD  DFINIT       ; INPUT ROUTINE INITIALIZE ADDRESS
  241 000F 5402     OUT:        .WORD  DFOU         ; OUTPUT ROUTINE ADDRESS
  242 0011 5302     OUTINT:     .WORD  DFOUIT       ; OUTPUT ROUTINE INITIALIZE ADDRESS
  243 0013 6802     ERR:        .WORD  DFERR        ; ERROR NOTIFICATION ROUTINE, CALLED WHEN
  244                                              ; AN ERROR IS ENCOUNTERED AND LISTING IS
  245                                              ; SUPPRESSED
  246
  247               ;           LISTING CONTROL PARAMETERS
  248
  249 0015 01       LIST:       .BYTE  1            ; MUSIC OBJECT CODE LISTING IN HEX IF NOT 0
  250 0016 01       ECHO:       .BYTE  1            ; ECHO INPUT STATEMENT IN LISTING IF NOT 0
  251
  252               ;           ADDITIONAL CONSTANTS AND VARIABLES USED BY THE COMPILER
  253
  254 0017 00       INBFPT:     .BYTE  0            ; INPUT BUFFER POINTER, OFFSET FROM (INBFA)
  255 0018 00       OUBFPT:     .BYTE  0            ; OUTPUT BUFFER POINTER,OFFSET FROM (OUBFA)
  256 0019 0000     CODEPT:     .WORD  0            ; OBJECT CODE POINTER, ABS ADDRESS
  257 001B 0000     CODECT:     .WORD  0            ; COUNT OF OBJECT CODE BYTES GENERATED
  258 001D 0000     SYMTPT:     .WORD  0            ; SYMBOL TABLE POINTER, ABS ADDRESS
  259 001F 00       SYMTCT:     .BYTE  0            ; COUNT OF SYMBOLS ENTERED INTO THE TABLE
  260 0020 0000     INDJMP:     .WORD  0            ; TEMPORARY STORAGE FOR INDIRECT JUMP
  261 0022 00       NBF:        .BYTE  0            ; RESULT OF NUMBER COLLECT ROUTINE
  262 0023 00       ENDFLG:     .BYTE  0            ; SET NON-ZERO WHEN END STATEMENT PROC.
  263 0024 0000     SUBSKP:     .WORD  0            ; ADDR OF "SUB" STATEMNENT IN OBJECT CODE
  264
  265                                              ; RETURN PARAMETERS FROM NOTE COLLECT
  266 0026 00       NTVOIC:     .BYTE  0            ; VOICE NUMBER, IF SPECIFIED, 0 OTHERWISE
```

NOTCMP KIM NOTRAN COMPILER
CONSTANTS AND DATA

```
  267 0027 00       NTDUR:      .BYTE  0            ; DURATION CODE FOR THE NOTE
  268 0028 00       NTDURX:     .BYTE  0            ; DURATION TIME FOR THE NOTE
  269 0029 00       NTPTC:      .BYTE  0            ; COMPOSITE PITCH FOR THE NOTE
  270 002A 00       NTOCT:      .BYTE  0            ; OCTAVE NUMBER FOR THE NOTE
  271
  272 002B 00       NTERR:      .BYTE  0            ; ERROR FLAG FOR NOTE COLLECT ROUTINE
  273 002C 00       NMERR:      .BYTE  0            ; ERROR FLAG FOR NUMBER COLLECT ROUTINE
  274
  275                                              ; DATA FOR PROCESSING MUSICAL EVENTS
  276 002D 00       EVTBLD:     .BYTE  0            ; EVENT BEING BUILT IF NOT ZERO
  277 002E 00       DUR:        .BYTE  0            ; DURATION OF SHORTEST NOTE
  278 002F 00       VPPNT:      .BYTE  0            ; VOICE NUMBER BEING PROCESSED
  279 0030          VXPTC:      .=.+   4            ; CURRENT VOICE COMPOSITE PITCHES
  280 0034          VXOCT:      .=.+   4            ; CURRENT VOICE PITCH OCTAVES
  281 0038          VXABS:      .=.+   4            ; FORCE ABSOLUTE PITCH FLAGS, NOT 0 = FORCE
  282 003C          VXWAV:      .=.+   4            ; CURRENT VOICE WAVEFORM NUMBERS
  283 0040          VXDUR:      .=.+   4            ; CURRENT VOICE DURATIONS
  284
  285 0044 00       KWTP:       .BYTE  0            ; MISCELLANEOUS TEMPORARY STORAGE FOR LOW
  286 0045 00       KWMC:       .BYTE  0            ; LEVEL ROUTINES
  287 0046 00       CDSAV:      .BYTE  0
  288
```

```
                                .PAGE  'DEFAULT INPUT, OUTPUT, AND ERROR  ROUTINES'
 289                    ;       DFIN   DEFAULT INPUT ROUTINE
 290
 291                    ;       THIS INPUT ROUTINE IS DESIGNED TO BE USED WITH A SERIAL DEVICE
 292                    ;       SUCH AS A #33 TELETYPE WITH PAPER TAPE.  THE PAPER TAPE SHOULD
 293                    ;       BE FORMATTED INTO LINES WITH THE CONTROL CHARACTER SEQUENCE [CR
 294                    ;       LF DC1 DEL] BETWEEN THE TEXT OF EACH LINE.  THE [DC1 DEL]
 295                    ;       SEQUENCE IS USED TO STOP THE TTY READER AFTER AN INPUT LINE HAS
 296                    ;       BEEN READ.  TO INITIATE READING OF A LINE THIS ROUTINE FIRST
 297                    ;       SENDS A DC2 CONTROL CHARACTER WHICH STARTS THE TTY READER.
 298                    ;       TEXT READ IN IS STORED IN THE INPUT BUFFER.  ANY CONTROL
 299                    ;       CHARACTERS PRECEEDING THE TEXT ARE THROWN AWAY SO THAT THE
 300                    ;       FIRST CHARACTER IN THE BUFFER IS THE FIRST PRINTABLE CHARACTER
 301                    ;       IN THE TEXT.  EXCESS CHARACTERS IN THE INPUT LINE BEYOND 72 ARE
 302                    ;       THROWN AWAY UNTIL A "CR" IS ENCOUNTERED.  BS (X'08) OR < (X'5F)
 303                    ;       APPEARING IN THE STRING CAUSES THE PREVIOUS CHARACTER IN THE
 304                    ;       BUFFER TO BE DELETED THUS EFFECTING A BACKSPACE.  LOWER CASE
 305                    ;       ALPHABETIC CHARACTERS ARE TRANSLATED TO THEIR UPPER CASE
 306                    ;       EQUIVALENT.  WHEN THE DC1 CHARACTER IS READ AND ECHOED BACK
 307                    ;       THROUGH THE KIM TTY PORT, THE TTY READER STOPS THUS ALLOWING
 308                    ;       TIME TO COMPILE THE LINE AND PRINT THE LISTING IF ENABLED.  THE
 309                    ;       TEXT STORED IN THE INPUT BUFFER IS TERMINATED WITH A "CR"
 310                    ;       CHARACTER.  THE KIM TTY INPUT AND OUTPUT ROUTINES ARE UTILIZED
 311                    ;       FOR INDIVIDUAL CHARACTER INPUT AND OUTPUT.  NOTE THAT TTY MODE
 312                    ;       OF OPERATION (JUMPER FROM A-XX TO A-XX ON THE KIM BOARD) MUST
 313                    ;       BE SELECTED FOR THE TTY INPUT ROUTINE TO FUNCTION PROPERLY.
 314
 315                    ;       THE USER MAY SUBSTITUTE HIS OWN INPUT ROUTINE. THE COMPILER
 316                    ;       ONLY REQUIRES THAT THE TEXT IN THE INPUT BUFFER START WITH THE
 317                    ;       FIRST CHARACTER TO BE PROCESSED AND BE TERMINATED WITH A "CR".
 318                    ;       THE USER ROUTINE MAY FREELY UTILIZE ALL OF THE REGISTERS, AND
 319                    ;       ANY BASE PAGE LOCATIONS NOT APPEARING IN THE COMPILER LISTING.
 320
 321 0047                       .=     X'200        ; PUT DEFAULT I/O ROUTINES IN PAGE 2
 322
 323 0200 60        DFINIT:     RTS                 ; DUMMY INPUT INITIALIZE ROUTINE
 324
 325 0201 A200      DFIN:       LDX    #0           ; INITIALIZE INPUT BUFFER POINTER
 326 0203 A912                  LDA    #X'12        ; TRANSMIT AN XON (DC2) TO START TTY READER
 327 0205 20A01E                JSR    KIMOCH
 328 0208 205A1E    DFIN1:      JSR    KIMICH       ; WAIT FOR AN INPUT CHARACTER
 329 020B C920                  CMP    #X'20        ; TEST IF IT IS ANY KIND OF CONTROL CHAR
 330 020D 90F9                  BCC    DFIN1        ; IGNORE IT IF SO
 331 020F C97F                  CMP    #X'7F        ; IGNORE DEL ALSO
 332 0211 F0F5                  BEQ    DFIN1
 333 0213 D00F                  BNE    DFIN3        ; GO STORE FIRST TEXT CHARACTER
 334 0215 205A1E    DFIN2:      JSR    KIMICH       ; GET A TEXT CHARACTER
 335 0218 C97F                  CMP    #X'7F        ; TEST IF DEL
 336 021A F0F9                  BEQ    DFIN2        ; IGNORE IF SO
 337 021C C90D                  CMP    #X'0D        ; TEST IF CR
 338 021E F024                  BEQ    DFIN4        ; JUMP IF SO
 339 0220 E048                  CPX    #72          ; TEST IF 72 TEXT CHARS ALREADY BEEN STORED
 340 0222 F0F1                  BEQ    DFIN2        ; IGNORE BEYOND 72 UNTIL CR
 341 0224 C908      DFIN3:      CMP    #X'08        ; TEST IF IT IS A BACKSPACE CONTROL
 342 0226 F004                  BEQ    DFIN32       ; JUMP IF IT IS
```



```
 343 0228 C93C                  CMP    #'<'         ; TEST IF IT IS A BACKARROW
 344 022A D005                  BNE    DFIN35       ; SKIP AHEAD IF NOT
 345 022C E000      DFIN32:     CPX    #0           ; EXECUTE BACKSPACE, TEST IF AT START OF
 346 022E F001                  BEQ    DFIN35       ; BUFFER, NO BACKSPACE IF SO
 347 0230 CA                    DEX
 348 0231 C97B      DFIN35:     CMP    #X'7B        ; TEST IF LOWER CASE ALPHABETIC
 349 0233 B006                  BCS    DFIN37       ; JUMP AHEAD IF NOT
 350 0235 C961                  CMP    #X'61
 351 0237 9002                  BCC    DFIN37       ; JUMP AHEAD IF NOT
 352 0239 E920                  SBC    #X'20        ; TRANSLATE LOWER CASE TO UPPER CASE
 353 023B 48        DFIN37:     PHA                 ; STORE CHARACTER IN INPUT BUFFER
 354 023C 8A                    TXA
 355 023D A8                    TAY
 356 023E 68                    PLA
 357 023F 9100                  STA    (INBFA),Y
 358 0241 E8                    INX                 ; INCREMENT BUFFER INDEX
 359 0242 D0D1                  BNE    DFIN2        ; GO FOR NEXT CHARACTER
 360
 361 0244 48        DFIN4:      PHA                 ; STORE THE CR IN THE BUFFER
 362 0245 8A                    TXA
 363 0246 A8                    TAY
 364 0247 68                    PLA
 365 0248 9100                  STA    (INBFA),Y
 366 024A A22C                  LDX    #44          ; WAIT 4 CHARACTER TIMES AT CRRRENT BAUD
 367 024C 20D41E    DFIN5:      JSR    KIMDLY       ; RATE TO AVOID POSSIBLE CONFLICT BETWEEN
 368 024F CA                    DEX                 ; INPUT TERMINATION SEQUENCE AND
 369 0250 D0FA                  BNE    DFIN5        ; LISTING OUTPUT
 370 0252 60                    RTS                 ; RETURN WITH INPUT BUFFER LOADED
 371
 372
 373                    ;       DFOU   DEFAULT OUTPUT ROUTINE
 374
 375                    ;       THIS OUTPUT ROUTINE IS DESIGNED TO BE USED WITH A SERIAL DEVICE
 376                    ;       SUCH AS A #33 TELETYPE.  THE COMPILER SUPPLIES AN OUTPUT BUFFER
 377                    ;       CONTAINING A FORMATTED CHARACTER STRING TERMINATED BY ASCII
 378                    ;       "CR" AND "LF" CONTROL CHARACTERS.  THE LINE HAS A MAXIMUM OF 72
 379                    ;       PRINTABLE CHARACTERS.  THE KIM TTY OUTPUT ROUTINE IS USED FOR
 380                    ;       INDIVIDUAL CHARACTER OUTPUT.
 381
 382                    ;       THE USER MAY SUBSTITUTE HIS OWN OUTPUT ROUTINE.  THE USER
 383                    ;       ROUTINE MAY FREELY UTILIZE ALL OF THE REGISTERS, AND ANY BASE
 384                    ;       PAGE LOCATIONS NOT APPEARING IN THE COMPILER LISTING.
 385
 386 0253 60        DFOUIT:     RTS                 ; DUMMY OUTPUT INITIALIZE ROUTINE
 387
 388 0254 A200      DFOU:       LDX    #0           ; INITIALIZE OUTPUT POINTER
 389 0256 8A        DFOUT1:     TXA                 ; GET A CHARACTER FROM THE BUFFER
 390 0257 A8                    TAY
 391 0258 B102                  LDA    (OUBFA),Y
 392 025A C90A                  CMP    #X'0A        ; TEST IF IT IS A LF
 393 025C F006                  BEQ    DFOUT2       ; JUMP OUT IF SO
 394 025E 20A01E                JSR    KIMOCH       ; OUTPUT THE CHARACTER
 395 0261 E8                    INX                 ; INCREMENT OUTPUT POINTER
 396 0262 D0F2                  BNE    DFOUT1       ; GO OUTPUT NEXT CHAR
 397 0264 20A01E    DFOUT2:     JSR    KIMOCH       ; OUTPUT THE LF
```

NOTCMP KIM NOTRAN COMPILER
DEFAULT INPUT, OUTPUT, AND ERROR ROUTINES

```
398 0267 60                 RTS                 ; AND RETURN
399
400
401                 ;       DFERR  DEFAULT ERROR ROUTINE TO BE USED WHEN LISTING IS
402                 ;       SUPPRESSED ERROR CODE IS IN A.
403                 ;       A "SAVE STATUS" CALL IS MADE TO THE KIM MONITOR WHERE THE USER
404                 ;       CAN EXAMINE THE ERROR CODE AND EITHER CONTINUE OR ABORT THE
405                 ;       COMPILATION.
406
407                 ;       THE USER MAY SUPPLY HIS OWN ERROR ROUTINE. THE INDEX REGISTERS
408                 ;       X AND Y, SHOULD BE SAVED WHILE THE ERROR REPORT IS BEING MADE
409                 ;       AND RESTORED BEFORE RETURNING TO THE COMPILER.
410
411 0268 4C001C    DFERR:  JMP     KIMSAV      ; GO TO KIM SAVE STATUS ENTRY POINT
412 026B 60                 RTS                 ; RETURN TO NOTRAN COMPILER
413
```

NOTCMP KIM NOTRAN COMPILER
INITIALIZATION ROUTINE

```
                            .PAGE   'INITIALIZATION ROUTINE'
414 026C                    .=      X'2000      ; START IN EXTENDED MEMORY
415
416 2000 D8         NOTRAN: CLD                 ; INSURE BINARY ARITHMETIC
417 2001 A2F0               LDX     #X'F0       ; INITIALIZE STACK POINTER
418 2003 9A                 TXS
419 2004 A900               LDA     #0          ; ZERO SOME PARAMETERS
420 2006 851F               STA     SYMTCT      ; NUMBER OF SYMBOLS IN TABLE
421 2008 8518               STA     CODECT      ; NUMBER OF OBJECT CODE BYTES GENERATED
422 200A 851C               STA     CODECT+1
423 200C 8524               STA     SUBSKP      ; CLEAR SKIP ADDRESS IN SUB STATEMENT
424 200E 8525               STA     SUBSKP+1
425 2010 852D               STA     EVTBLD      ; TURN EVENT BEING BUILT FLAG OFF
426 2012 8523               STA     ENDFLG      ; TURN END FLAG OFF
427 2014 A8                 TAY                 ; CLEAR THE SYMBOL TABLE
428 2015 9104               STA     (SYMTA),Y
429 2017 852F               STA     VPPNT       ; INITIALIZE VOICE PROCESSING POINTER
430 2019 AA                 TAX
431 201A A900       NOTR1:  LDA     #0
432 201C 953C               STA     VXWAV,X     ; DEFAULT TO WAVEFORM 1 ON ALL VOICES
433 201E A9FF               LDA     #X'FF
434 2020 9540               STA     VXDUR,X     ; DEACTIVATE ALL VOICES
435 2022 9538               STA     VXABS,X     ; FORCE ABSOLUTE PITCH ON ALL VOICES
436 2024 E8                 INX
437 2025 E004               CPX     #4
438 2027 D0F1               BNE     NOTR1
439 2029 A506               LDA     CODEA       ; INITIALIZE OBJECT CODE POINTER
440 202B 8519               STA     CODEPT
441 202D A507               LDA     CODEA+1
442 202F 851A               STA     CODEPT+1
443 2031 20F325             JSR     INIT        ; INITIALIZE INPUT ROUTINE
444 2034 20F625             JSR     OUIT        ; INITIALIZE OUTPUT ROUTINE
445
```

```
                                .PAGE  'MAIN STATEMENT PROCESSOR'
 446
 447 2037 20ED25    STAT:    JSR    IN          ; READ AN INPUT LINE
 448 203A A000               LDY    #0
 449 203C A516               LDA    ECHO        ; TEST IF ECHO TO OUTPUT ENABLED
 450 203E D00C               BNE    STAT1       ; SKIP AHEAD IF SO
 451 2040 B100               LDA    (INBFA),Y   ; LOOK AT FIRST CHAR. OF INPUT LINE IF NOT
 452 2042 C92A               CMP    #'*'        ; TEST IF IT IS A COMMENT LINE
 453 2044 D018               BNE    STAT3       ; JUMP AHEAD IF NOT A COMMENT
 454 2046 A515               LDA    LIST        ; IF A COMMENT, SEE IF CODE LISTING ENABLED
 455 2048 F0ED               BEQ    STAT        ; IGNORE COMMENT IF LISTING DISABLED
 456 204A A000               LDY    #0          ; COPY INPUT BUFFER INTO OUTPUT BUFFER
 457 204C B100      STAT1:   LDA    (INBFA),Y   ; GET A CHARACTER FROM THE INPUT BUFFER
 458 204E 9102               STA    (OUBFA),Y   ; PUT IT INTO THE OUTPUT BUFFER
 459 2050 C8                 INY                ; INCREMENT X TO NEXT CHARACTER
 460 2051 C90D               CMP    #X'0D       ; TEST IF A CARRIAGE RETURN WAS MOVED
 461 2053 F002               BEQ    STAT2       ; JUMP OUT IF SO
 462 2055 D0F5               BNE    STAT1       ; GO MOVE ANOTHER CHARACTER
 463 2057 A90A      STAT2:   LDA    #X'0A       ; PUT A LINE FEED AT END OF OUTPUT BUFFER
 464 2059 9102               STA    (OUBFA),Y
 465 205B 20F025             JSR    OU          ; PRINT AN OUTPUT LINE
 466
 467 205E A000      STAT3:   LDY    #0          ; INITIALIZE INPUT BUFFER POINTER
 468 2060 8417               STY    INBFPT
 469 2062 8418               STY    OUBFPT      ; INITIALIZE OUTPUT BUFFER POINTER
 470 2064 202A23             JSR    OULOC       ; GENERATE CURRENT LOCATION ON LISTING
 471 2067 B100               LDA    (INBFA),Y   ; GET FIRST CHARACTER OF INPUT LINE
 472 2069 C92A               CMP    #'*'        ; TEST IF COMMENT LINE
 473 206B F0CA               BEQ    STAT        ; GO FOR NEXT STATEMENT IF A COMMENT LINE
 474 206D C920               CMP    #' '        ; TEST IF A BLANK
 475 206F F03B               BEQ    STAT8       ; JUMP IF SO, NO IDENTIFIER FOR LINE
 476 2071 C930               CMP    #'0'        ; TEST IF A DIGIT
 477 2073 9004               BCC    STAT4       ; JUMP IF NOT
 478 2075 C93A               CMP    #'9'+1
 479 2077 9008               BCC    STAT6       ; JUMP IF IT IS A DIGIT
 480 2079 A906      STAT4:   LDA    #6          ; SIGNAL ER 6 IF NOT A *, DIGIT, OR BLANK
 481 207B 20F925    STAT5:   JSR    ER
 482 207E 4CC221             JMP    ESTAT       ; GO TO END OF STATEMENT PROCESSING
 483
 484 2081 A52D      STAT6:   LDA    EVTBLD      ; TEST IF IDENTIFIER IN MIDDLE OF EVENT
 485 2083 F005               BEQ    STAT65
 486 2085 A911               LDA    #X'11       ; GENERATE ER-11 IF SO
 487 2087 20F925             JSR    ER          ; AND PROCESS IDENTIFIER ANYWAY
 488 208A 20B724    STAT65:  JSR    NCOL        ; EVALUATE THE STATEMENT IDENTIFIER
 489 208D A52C               LDA    NMERR       ; TEST FOR OVERFLOW
 490 208F D0EA               BNE    STAT5       ; JUMP IF SO
 491 2091 A522               LDA    NBF         ; GET THE IDENTIFIER
 492 2093 F0E4               BEQ    STAT4       ; ZERO NOT ALLOWED FOR AN IDENTIFIER
 493 2095 206625             JSR    SYMSR       ; CHECK FOR DUPLICATE SYMBOL IN THE TABLE
 494 2098 9008               BCC    STAT7       ; JUMP IF NOT
 495 209A A903               LDA    #3          ; SIGNAL ER 3 IF DUPLICATE IDENTIFIER
 496 209C 20F925             JSR    ER
 497 209F 4CAC20             JMP    STAT8       ; CONTINUE PROCESSING THE STATEMENT
 498 20A2 209C25    STAT7:   JSR    SYMAD       ; ADD THE IDENTIFIER TO THE SYMBOL TABLE
 499 20A5 B005               BCS    STAT8       ; JUMP IF SUCCESSFUL ADD
```



```
 500 20A7 A904               LDA    #4          ; GENERATE ER-4 IF SYMBOL TABLE OVERFLOW
 501 20A9 20F925             JSR    ER
 502
 503                 ;       PROCESS REMAINDER OF STATEMENT AFTER LEADING BLANK, IDENTIFIER,
 504                 ;       OR SEMICOLON
 505                                            .
 506 20AC 20D225    STAT8:   JSR    NNB         ; MOVE TO NEXT NON-BLANK CHARACTER
 507 20AF 20F924             JSR    KWCOL       ; TRY TO RECOGNIZE A KEYWORD
 508 20B2 900D               BCC    STAT10      ; SKIP AHEAD IF NO KEYWORD
 509 20B4 BD2925             LDA    KWTAB,X     ; MOVE JUMP ADDRESS FROM KEYWORD TABLE TO
 510 20B7 8520               STA    INDJMP      ; INDIRECT JUMP POINTER IN PAGE 0
 511 20B9 BD2A25             LDA    KWTAB+1,X
 512 20BC 8521               STA    INDJMP+1
 513 20BE 6C2000             JMP    (INDJMP)    ; JUMP TO APPROPRIATE KEYWORD PROCESSING
 514                                            ; ROUTINE
 515
 516 20C1 20BF23    STAT10:  JSR    NTCOL       ; TRY TO RECOGNIZE A VALID NOTE OR REST
 517 20C4 A52B               LDA    NTERR       ; GET ERROR CODE
 518 20C6 F006               BEQ    STAT11      ; SKIP AHEAD IF NO ERROR
 519 20C8 20F925             JSR    ER          ; SIGNAL ERROR IF NTCOL DETECTED ONE
 520 20CB 4CAE21             JMP    ESPEC       ; GO TO END OF SPECIFICATION PROCESSING
 521
 522                 ;       THE FOLLOWING CODE FIGURES OUT THE VOICE NUMBER THAT WILL BE
 523                 ;       ASSUMED BY THE OBJECT CODE INTERPRETER WHEN THE MUSIC IS PLAYED
 524
 525 20CE A52D      STAT11:  LDA    EVTBLD      ; TEST IF THE START OF A NEW EVENT
 526 20D0 D016               BNE    STAT12      ; SKIP AHEAD IF NOT
 527 20D2 852F               STA    VPPNT       ; START SCAN WITH VOICE 1
 528 20D4 C62D               DEC    EVTBLD      ; SET EVENT BEING BUILT FLAG
 529 20D6 A540               LDA    VXDUR       ; MAKE SURE AT LEAST ONE VOICE IS ACTIVE
 530 20D8 2541               AND    VXDUR+1     ; AND HAS A ZERO REMAINING DURATION
 531 20DA 2542               AND    VXDUR+2
 532 20DC 2543               AND    VXDUR+3
 533 20DE F008               BEQ    STAT12      ; SKIP AHEAD IF OK
 534 20E0 A915               LDA    #X'15       ; SIGNAL ER-15 IF NOT, EITHER NO VOICES
 535 20E2 20F925             JSR    ER          ; ACTIVE OR COMPILER ERROR
 536 20E5 4C221C             JMP    KIMMON      ; ABORT COMPILATION
 537
 538 20E8 A62F      STAT12:  LDX    VPPNT       ; GET CURRENT VOICE NUMBER
 539 20EA B540      STA122:  LDA    VXDUR,X     ; GET DURATION BYTE OF CURRENT VOICE
 540 20EC F003               BEQ    STA125      ; GO ASSIGN THIS NOTE SPEC TO IT IF ZERO
 541 20EE E8                 INX                ; INCREMENT TO NEXT VOICE NUMBER
 542 20EF D0F9               BNE    STA122      ; GO TRY IT
 543
 544 20F1 A526      STA125:  LDA    NTVOIC      ; TEST IF A VOICE NUMBER GIVEN EXPLICITLY
 545 20F3 F00C               BEQ    STAT13      ; JUMP IF NOT
 546 20F5 C626               DEC    NTVOIC      ; CHANGE EXPLICIT VOICE NUMBER TO ZERO ORG
 547 20F7 8A                 TXA                ; COMPARE WITH IMPLICIT VOICE NUMBER
 548 20F8 C526               CMP    NTVOIC
 549 20FA F005               BEQ    STAT13      ; JUMP AHEAD IF EQUAL, OK
 550 20FC A907               LDA    #7          ; SIGNAL ER 7 IF NOT EQUAL
 551 20FE 20F925             JSR    ER          ; AND CONTINUE USING ASSUMED VOICE NUMBER
 552
 553 2101 A529      STAT13:  LDA    NTPTC       ; TEST IF A REST WAS SPECIFIED
 554 2103 F056               BEQ    STA155      ; JUMP IF SO
```

```
555 2105 A52A              LDA   NTOCT        ; TEST IF EXPLICIT OCTAVE NUMBER GIVEN
556 2107 D002              BNE   STAT14       ; JUMP IF SO
557 2109 B534              LDA   VXOCT,X      ; IF NOT, USE LAST USED OCTAVE NUMBER
558 210B 9534      STAT14: STA   VXOCT,X      ; UPDATE THE LAST USED OCTAVE NUMBER
559 210D 0A                ASLA               ; COMPUTE 12 TIMES THE OCTAVE NUMBER
560 210E 0A                ASLA
561 210F 852A              STA   NTOCT
562 2111 0A                ASLA
563 2112 652A              ADC   NTOCT
564 2114 6529              ADC   NTPTC        ; ADD IN THE PITCH WITHIN THE OCTAVE
565 2116 69F4              ADC   #-12         ; ADD IN CORRECTION FACTOR SO THAT C1=1
566 2118 C93E              CMP   #62          ; TEST IF NOTE IS IN RANGE
567 211A 9007              BCC   STA145       ; JUMP IF OK
568 211C A908              LDA   #8           ; SIGNAL ER 8 IF OUT OF RANGE
569 211E 20F925            JSR   ER
570 2121 A93D              LDA   #61          ; AND USE HIGHEST POSSIBLE NOTE
571 2123 8529      STA145: STA   NTPTC        ; SAVE COMPOSITE PITCH IN NTPTC
572 2125 B538              LDA   VXABS,X      ; TEST IF ABSOLUTE PITCH REQUIRED
573 2127 D019              BNE   STAT15       ; JUMP IF SO
574 2129 A529              LDA   NTPTC        ; TEST IF DIFFERENCE BETWEEN PREVIOUS PITCH
575 212B 38                SEC                ; AND NEXT PITCH IS SMALL ENOUGH FOR A
576 212C F530              SBC   VXPTC,X      ; SHORT NOTE SPECIFICATION
577 212E C908              CMP   #8
578 2130 1010              BPL   STAT15       ; JUMP IF GREATER THAN +7
579 2132 C9F9              CMP   #-7
580 2134 300C              BMI   STAT15       ; JUMP IF LESS THAN -7
581
582 2136 0A                ASLA               ; FORMAT A SHORT NOTE SPECIFICATION
583 2137 0A                ASLA
584 2138 0A                ASLA
585 2139 0A                ASLA               ; PITCH DISPLACEMENT IN UPPER HEX DIGIT
586 213A 0527              ORA   NTDUR        ; DURATION IN LOWER HEX DIGIT
587 213C 208623            JSR   CODOU        ; OUTPUT A BYTE OF OBJECT CODE
588 213F 4C6521            JMP   STAT16       ; GO CLEANUP AFTER NOTE SPECIFICATION
589
590 2142 A960      STAT15: LDA   #X'60        ; FORMAT AN ABSOLUTE LONG NOTE SPEC
591 2144 208623            JSR   CODOU        ; OUTPUT A 60 TO SIGNAL LONG NOTE SPEC
592 2147 A529              LDA   NTPTC        ; GET COMPOSITE PITCH
593 2149 0A                ASLA               ; MULTIPLY IT BY 2
594 214A 208623            JSR   CODOU        ; OUTPUT AS A BYTE OF OBJECT CODE
595 214D B53C              LDA   VXWAV,X      ; GET WAVEFORM NUMBER FOR THE VOICE
596 214F 0A                ASLA               ; SHIFT LEFT 4 FOR LEFT HEX DIGIT
597 2150 0A                ASLA
598 2151 0A                ASLA
599 2152 0A                ASLA
600 2153 0527              ORA   NTDUR        ; DURATION IN LOWER HEX DIGIT
601 2155 208623            JSR   CODOU        ; OUTPUT AS A BYTE OF OBJECT CODE
602 2158 4C6521            JMP   STAT16       ; GO CLEANUP AFTER NOTE SPEC
603
604 215B A980      STA155: LDA   #X'80        ; FORMAT A REST SPEC
605 215D 0527              ORA   NTDUR        ; OR IN DURATION
606 215F 208623            JSR   CODOU        ; OUTPUT AS A BYTE OF OBJECT CODE
607 2162 4C6921            JMP   STA165       ; SKIP PITCH UPDATE ON RESTS
608
609 2165 A529      STAT16: LDA   NTPTC        ; CLEANUP AFTER NOTE SPEC
```



```
610 2167 9530              STA   VXPTC,X      ; UPDATE CURRENT PITCH FOR THE VOICE
611 2169 A900      STA165: LDA   #0           ; CLEAR FORCE ABSOLUTE PITCH FLAG
612 216B 9538              STA   VXABS,X
613 216D A528              LDA   NTDURX       ; UPDATE DURATION FOR THE VOICE
614 216F 9540              STA   VXDUR,X
615
616                ;       DETERMINE IF A COMPLETE MUSICAL EVENT HAS BEEN BUILT, IF NOT
617                ;       LEAVE VPPNT POINTING TO NEXT EXPIRED VOICE. IF SO, CLOSE OUT
618                ;       THIS EVENT BY SUBTRACTING SHORTEST DURATION IN THIS EVENT
619                ;       FROM THE DURATION OF EACH ACTIVE VOICE AND TURNING THE EVENT
620                ;       BEING BUILT FLAG OFF
621
622 2171 E8        STAT17: INX                ; INCREMENT TO NEXT VOICE
623 2172 E004              CPX   #4           ; TEST IF ALL VOICES SCANNED
624 2174 F008              BEQ   STAT18       ; JUMP IF SO
625 2176 B540              LDA   VXDUR,X      ; TEST IF THIS NEXT VOICE WILL REQUIRE A
626                                           ; NOTE  SPEC
627 2178 D0F7              BNE   STAT17       ; GO TO NEXT VOICE IF NOT
628 217A 862F              STX   VPPNT        ; SAVE THE NEXT VOICE NUMBER
629 217C F030              BEQ   ESPEC        ; GO TO END OF SPEC PROCESSING
630
631 217E A9FF      STAT18: LDA   #X'FF        ; SCAN THE VOICES TO FIND THE ONE WITH THE
632 2180 D53F      STAT19: CMP   VXDUR-1,X    ; SHORTEST DURATION
633 2182 9002              BCC   STAT20
634 2184 B53F              LDA   VXDUR-1,X    ; UPDATE CURRENT SHORTEST IN A
635 2186 CA        STAT20: DEX
636 2187 D0F7              BNE   STAT19       ; LOOP UNTIL ALL 4 VOICES SCANNED
637
638 2189 852E              STA   DUR          ; SCAN THE VOICES AGAIN AND SUBTRACT THE
639 218B B540      STAT21: LDA   VXDUR,X      ; SHORTEST DURATION FROM EACH ONE THAT IS
640 218D C9FF              CMP   #X'FF        ; ACTIVE
641 218F F005              BEQ   STAT22       ; SKIP INACTIVE VOICES
642 2191 38                SEC
643 2192 E52E              SBC   DUR
644 2194 9540              STA   VXDUR,X
645 2196 E8        STAT22: INX
646 2197 E004              CPX   #4
647 2199 D0F0              BNE   STAT21
648
649 219B A900              LDA   #0           ; CLEAR THE EVENT BEING BUILT FLAG SINCE AN
650 219D 852D              STA   EVTBLD       ; EVENT HAS JUST BEEN COMPILED
651 219F F00D              BEQ   ESPEC        ; GO TO END OF SPEC PROCESSING
652
653                ;       CLEANUP AFTER A KEYWORD OR NOTE SPECIFICATION
654
655 21A1 A52D      EXSPEC: LDA   EVTBLD       ; TEST IF AN EXECUTABLE KEYWORD
656                                           ; SPECIFICATION PROCESSED IN THE MIDDLE OF
657                                           ; A MUSICAL EVENT
658 21A3 F009              BEQ   ESPEC        ; OK IF NOT; GO TO END OF SPEC PROCESSING
659 21A5 A910              LDA   #X'10        ; GENERATE ER-10 IF SO
660 21A7 20F925            JSR   ER
661 21AA A900              LDA   #0           ; EMULATE INTERPRETER ACTION IN THIS
662 21AC 852D              STA   EVTBLD       ; SITUATION BY ABORTING CURRENT EVENT AND
663                                           ; STARTING A NEW ONE
664
```

```
 665 21AE A417      ESPEC:   LDY    INBFPT       ; SCAN FOR A SEMICOLON OR CR
 666 21B0 B100      ESPEC1:  LDA    (INBFA),Y    ; GET CURRENT CHARACTER FROM INPUT BUFFER
 667 21B2 C8                 INY                  ; MOVE TO NEXT CHARACTER IN INPUT BUFFER
 668 21B3 C90D               CMP    #X'0D        ; TEST IF A CR
 669 21B5 F00B               BEQ    ESTAT        ; GO TO END OF STATEMENT CLEANUP IF SO
 670 21B7 C93B               CMP    #';'         ; TEST IF A ";"
 671 21B9 F002               BEQ    ESPEC2       ; SKIP AHEAD IF SO
 672 21BB D0F3               BNE    ESPEC1       ; GO TEST NEXT CHARACTER IF NOT
 673 21BD 8417      ESPEC2:  STY    INBFPT       ; UPDATE INPUT BUFFER POINTER
 674 21BF 4CAC20             JMP    STAT8        ; GO PROCESS CONTINUATION OF STATEMENT
 675
 676                ;        CLEANUP AT END OF INPUT LINE
 677
 678 21C2 A90D      ESTAT:   LDA    #X'0D        ; END LISTING LINE WITH A CR AND LF
 679 21C4 205D23             JSR    OUCH
 680 21C7 A90A               LDA    #X'0A
 681 21C9 205D23             JSR    OUCH
 682 21CC A515               LDA    LIST         ; TEST IF LISTING ENABLED
 683 21CE F003               BEQ    ESTAT1       ; SKIP PRINT IF NOT
 684 21D0 20F025             JSR    OU           ; PRINT THE LISTING LINE IF LISTING ENABLED
 685 21D3 A523      ESTAT1:  LDA    ENDFLG       ; TEST IF END STATEMENT JUST PROCESSED
 686 21D5 D003               BNE    ESTAT2
 687 21D7 4C3720             JMP    STAT         ; GO PROCESS NEXT STATEMENT IF NOT
 688 21DA 4C221C    ESTAT2:  JMP    KIMMON       ; GO RETURN TO KIM MONITOR IF SO
 689
```



```
                                .PAGE  'KEYWORD PROCESSOR ROUTINES'
 690                ;        NVCP   DEFINE NUMBER OF VOICES, 1 ARGUMENT, RANGE 1-4
 691
 692 21DD 20D225    NVCP:    JSR    NNB          ; MOVE TO NEXT NON-BLANK CHARACTER
 693 21E0 202123             JSR    GETARG       ; PROCESS THE NUMBER THAT IS NEXT EXPECTED
 694                                             ; AND TEST IT FOR OVERFLOW ERRORS
 695 21E3 A522               LDA    NBF          ; TEST FOR LEGITIMATE RANGE
 696 21E5 F06E               BEQ    ARGER        ; ZERO NOT ALLOWED
 697 21E7 C905               CMP    #5           ; GREATER THAN 4 NOT ALLOWED
 698 21E9 B06A               BCS    ARGER
 699 21EB A950               LDA    #X'50        ; FORMAT A SET NUMBER OF VOICES SPEC
 700 21ED 208623             JSR    CODOU
 701 21F0 A522               LDA    NBF
 702 21F2 208623             JSR    CODOU
 703 21F5 4CA121             JMP    EXSPEC       ; GO TO END OF EXEC UTABLE SPEC PROCESSING
 704
 705                ;        ACTP   ACTIVATE VOICES, MULTIPLE ARGS ALLOWED, RANGE 1-4
 706
 707 21F8 A990      ACTP:    LDA    #X'90        ; SET ACTIVATE VOICE CODE
 708 21FA D002               BNE    CTP
 709
 710                ;        DCTP   DEACTIVATE VOICES, MULTIPLE ARGS ALLOWED, RANGE 1-4
 711
 712 21FC A980      DCTP:    LDA    #X'80        ; SET DEACTIVATE VOICE CODE
 713
 714                ;        COMMON CODE FOR ACTIVATE AND DEACTIVATE VOICE KEYWORDS
 715
 716 21FE 8546      CTP:     STA    CDSAV        ; SAVE ACTIVATE/DEACTIVATE CODE
 717 2200 202123    CTP1:    JSR    GETARG       ; PROCESS THE NUMBER THAT IS NEXT EXPECTED
 718                                             ; AND TEST IT FOR OVERFLOW ERRORS
 719 2203 C622               DEC    NBF          ; DECREMENT VOICE # FOR ZERO ORIGIN IN CODE
 720 2205 A522               LDA    NBF          ; TEST FOR LEGITIMATE RANGE
 721 2207 C904               CMP    #4           ; GREATER THAN 4 (BEFORE DECR) NOT ALLOWED
 722 2209 B04A               BCS    ARGER
 723 220B A546               LDA    CDSAV        ; FORMAT AN ACTIVATE/DEACTIVATE VOICE SPEC
 724 220D 208623             JSR    CODOU
 725 2210 A522               LDA    NBF
 726 2212 208623             JSR    CODOU
 727 2215 A622               LDX    NBF          ; GET VOICE NUMBER BACK
 728 2217 A546               LDA    CDSAV        ; GET ACTIVATE/DEACTIVATE CODE BACK
 729 2219 4990               EOR    #X'90        ; FORM FF IF DCT, 00 IF ACT
 730 221B F002               BEQ    CTP2
 731 221D A9FF               LDA    #X'FF
 732 221F 9540      CTP2:    STA    VXDUR,X      ; UPDATE ACTIVE/INACTIVE STATUS OF VOICE
 733 2221 A417               LDY    INBFPT       ; TEST IF ARGUMENT TERMINATOR WAS A ,
 734 2223 B100               LDA    (INBFA),Y
 735 2225 C92C               CMP    #','
 736 2227 F003               BEQ    CTP3         ; SKIP AHEAD IF SO
 737 2229 4CA121             JMP    EXSPEC       ; GO TO END OF SPEC PROCESSING IF NOT
 738 222C E617      CTP3:    INC    INBFPT       ; SKIP THE COMMA
 739 222E 4C0022             JMP    CTP1         ; AND GO PROCESS NEXT ARGUMENT
 740
 741                ;        WAVP   ASSIGN WAVEFORM TO VOICE, 2 ARGS, FIRST IS WAVEFORM
 742                ;        NUMBER RANGE 1-16, SECOND IS VOICE NUMBER, RANGE 1-4
 743
```

```
 744 2231 202123   WAVP:   JSR   GETARG       ; PROCESS THE NUMBER THAT IS NEXT EXPECTED
 745                                          ; AND TEST IT FOR OVERFLOW ERRORS
 746 2234 C622             DEC   NBF          ; CONVERT WAVE # TO ZERO ORIGIN
 747 2236 A522             LDA   NBF          ; TEST FOR LEGITIMATE RANGE FOR WAVE NUMBER
 748 2238 C910             CMP   #16          ; GREATER THAN 16 (BEFORE DECR) NOT ALLOWED
 749 223A B019             BCS   ARGER
 750 223C 48               PHA                ; SAVE WAVE NUMBER IF VALID
 751 223D E617             INC   INBFPT       ; SKIP OVER COMMA ARGUMENT SEPARATOR
 752 223F 202123           JSR   GETARG       ; PROCESS THE EXPECTED VOICE NUMBER
 753 2242 C622             DEC   NBF          ; CONVERT VOICE # TO ZERO ORIGIN
 754 2244 A522             LDA   NBF          ; TEST FOR LEGITIMATE RANGE FOR VOICE NO.
 755 2246 C904             CMP   #4
 756 2248 B00B             BCS   ARGER
 757 224A AA               TAX                ; PUT VOICE NUMBER IN X
 758 224B 68               PLA                ; GET WAVEFORM NUMBER BACK
 759 224C 953C             STA   VXWAV,X      ; UPDATE CURRENT WAVEFORM NUMBER IN VOICE
 760 224E A9FF             LDA   #X'FF
 761 2250 9538             STA   VXABS,X      ; SET FORCE ABSOLUTE PITCH SO THAT
 762                                          ; NEXT NOTE GENERATED FOR THE VOICE WILL
 763                                          ; UPDATE THE WAVEFORM NUMBER IN THE
 764                                          ; INTERPRETER
 765 2252 4CAE21           JMP   ESPEC        ; GO TO END OF SPEC PROCESSING
 766
 767 2255 A901     ARGER:  LDA   #1           ; COMMON ARGUMENT ERROR ROUTINE
 768 2257 20F925           JSR   ER
 769 225A 4CAE21           JMP   ESPEC        ; GO TO END OF SPEC PROCESSING
 770
 771               ;       JSRP  JUMP TO MUSICAL SUBROUTINE, 1 ARGUMENT, RANGE 1-255
 772
 773 225D A920     JSRP:   LDA   #X'20        ; SET JSR CODE
 774 225F D002             BNE   JP           ; GO TO COMMON JMP/JSR CODE
 775
 776               ;       JMP   UNCONDITIONAL JUMP, 1 ARGUMENT, RANGE 1-255
 777
 778 2261 A940     JMPP:   LDA   #X'40        ; SET JMP CODE
 779
 780               ;       COMMON JMP AND JSR CODE
 781
 782 2263 8546     JP:     STA   CDSAV        ; SAVE JSR/JMP CODE
 783 2265 202123           JSR   GETARG       ; EVALUATE ARGUMENT WHICH IS A NUMERICAL
 784 2268 A522             LDA   NBF          ; SYMBOL
 785 226A 206625           JSR   SYMSR        ; SEARCH SYMBOL TABLE FOR SYMBOL
 786 226D 901C             BCC   JP1          ; JUMP IF SYMBOL NOT FOUND
 787 226F A546             LDA   CDSAV        ; GENERATE JSR/JMP SPEC IN OBJECT CODE
 788 2271 208623           JSR   CODOU
 789 2274 A000             LDY   #0           ; GET LOW BYTE OF SYMBOL VALUE
 790 2276 B11D             LDA   (SYMTPT),Y
 791 2278 38               SEC                ; SUBTRACT CODE ORIGIN FROM IT FOR RELATIVE
 792 2279 E506             SBC   CODEA        ; ADDRESSING WITHIN OBJECT CODE
 793 227B 08               PHP                ; SAVE CARRY OUT STATUS
 794 227C 208623           JSR   CODOU        ; PUT LOW BYTE IN OBJECT CODE
 795 227F C8               INY                ; GET HIGH BYTE OF SYMBOL VALUE
 796 2280 B11D             LDA   (SYMTPT),Y
 797 2282 28               PLP                ; RESTORE CARRY FROM LOW BYTE
 798 2283 E507             SBC   CODEA+1
```

NOTCMP KIM NOTRAN COMPILER
KEYWORD PROCESSOR ROUTINES

```
 799 2285 208623           JSR   CODOU        ; PUT HIGH BYTE IN OBJECT CODE
 800 2288 4CAE21           JMP   ESPEC        ; GO TO END OF SPEC PROCESSING
 801 228B A902     JP1:    LDA   #2           ; GENERATE UNDEFINED SYMBOL ERROR
 802 228D 20F925           JSR   ER
 803 2290 4CA121           JMP   EXSPEC       ; GO TO END OF EXECUTABLE SPEC PROCESSING
 804
 805               ;       RTSP  RETURN FROM SUBROUTINE, NO ARGS
 806
 807 2293 A930     RTSP:   LDA   #X'30        ; GENERATE A RTS SPEC IN THE OBJECT CODE
 808 2295 208623           JSR   CODOU
 809 2298 4CA121           JMP   EXSPEC       ; GO TO END OF EXECUTABLE SPEC PROCESSING
 810
 811               ;       ENDP  END OF SONG, NO ARGS
 812
 813 229B A900     ENDP:   LDA   #0           ; GENERATE AN END SPEC IN THE OBJECT CODE
 814 229D 208623           JSR   CODOU
 815 22A0 A9FF             LDA   #X'FF        ; SET END FLAG
 816 22A2 8523             STA   ENDFLG
 817 22A4 A524             LDA   SUBSKP       ; TEST IF HANGING SUB LEFT OVER
 818 22A6 0525             ORA   SUBSKP+1
 819 22A8 F005             BEQ   ENDP1        ; SKIP AHEAD IF OK
 820 22AA A914             LDA   #X'14        ; FORMAT ER-14 IF SO
 821 22AC 201300           JSR   ERR
 822 22AF 4CAE21   ENDP1:  JMP   ESPEC        ; GO TO END OF SPEC PROCESSING
 823
 824               ;       TPOP  SET TEMPO, 1 ARGUMENT
 825
 826 22B2 202123   TPOP:   JSR   GETARG       ; PROCESS THE NUMBER THAT IS NEXT EXPECTED
 827                                          ; AND TEST IT FOR OVERFLOW ERRORS
 828 22B5 D09E             BNE   ARGER        ; JUMP OUT IF OVERFLOW ERROR
 829 22B7 A522             LDA   NBF          ; TEST FOR LEGITIMATE RANGE
 830 22B9 F09A             BEQ   ARGER        ; ZERO NOT ALLOWED
 831 22BB A910             LDA   #X'10        ; GENERATE A CHANGE TEMPO SPEC IN CODE
 832 22BD 208623           JSR   CODOU
 833 22C0 A522             LDA   NBF          ; GET THE TEMPO ARGUMENT
 834 22C2 208623           JSR   CODOU        ; *** USE IN RAW FORM ***  TEMPORARY
 835 22C5 4CA121           JMP   EXSPEC       ; GO TO END OF EXECUTABLE SPEC PROCESSING
 836
 837               ;       ABSP  FORCE ABSOLUTE PITCH UPDATE OF ALL VOICES, NO ARGS
 838
 839 22C8 A204     ABSP:   LDX   #4           ; SET UP TO SET FORCE ABSOLUTE PITCH FLAG
 840 22CA A9FF             LDA   #X'FF        ; IN ALL 4 VOICES
 841 22CC 9537     ABSP1:  STA   VXABS-1,X
 842 22CE CA               DEX
 843 22CF D0FB             BNE   ABSP1
 844 22D1 4CAE21           JMP   ESPEC        ; GO TO END OF SPEC PROCESSING
 845
 846               ;       SUBP  DEFINE BEGINNING OF SUBROUTINE AREA
 847
 848 22D4 A524     SUBP:   LDA   SUBSKP       ; TEST IF ALREADY A PENDING SUB
 849 22D6 0525             ORA   SUBSKP+1
 850 22D8 D018             BNE   SUBP1        ; GO TO ERROR IF SO
 851 22DA A940             LDA   #X'40        ; FORMAT AN UNCONDITIONAL JUMP IN THE
 852 22DC 208623           JSR   CODOU        ; OBJECT CODE
 853 22DF A519             LDA   CODEPT       ; SAVE PRESENT OBJECT CODE ADDRESS
```

```
 854 22E1 8524              STA    SUBSKP        ; IN SUBSKP
 855 22E3 A51A              LDA    CODEPT+1
 856 22E5 8525              STA    SUBSKP+1
 857 22E7 A900              LDA    #0            ; PUT BLANK ADDRESS FIELD IN THE JUMP
 858 22E9 208623            JSR    CODOU
 859 22EC 208623            JSR    CODOU
 860 22EF 4CA121            JMP    EXSPEC        ; GO TO END OF EXECUTABLE SPEC PROCESSING
 861 22F2 A912      SUBP1:  LDA    #X'12         ; GENERATE ER-13 IF ATTEMPT TO NEST SUB
 862 22F4 201300            JSR    ERR
 863 22F7 4CA121            JMP    EXSPEC        ; DO NOTHING
 864
 865                ;       ESBP   DEFINE END OF SUBROUTINE AREA
 866
 867 22FA A524      ESBP:   LDA    SUBSKP        ; TEST IF A PREVIOUS MATCHING "SUB"
 868 22FC 0525              ORA    SUBSKP+1
 869 22FE F019              BEQ    ESBP1         ; JUMP TO ERROR IF NOT
 870 2300 A000              LDY    #0
 871 2302 38                SEC                  ; SET CARRY FOR SUBTRACTION
 872 2303 A519              LDA    CODEPT        ; COMPUTE RELATIVE ADDRESS IN OBJECT CODE
 873 2305 E506              SBC    CODEA         ; AND FILL IN ADDRESS FIELD OF JUMP
 874 2307 9124              STA    (SUBSKP),Y    ; GENERATED BY "SUB" STATEMENT
 875 2309 C8                INY
 876 230A A51A              LDA    CODEPT+1
 877 230C E507              SBC    CODEA+1
 878 230E 9124              STA    (SUBSKP),Y
 879 2310 A900              LDA    #0            ; CLEAR SUB DEFINITION
 880 2312 8524              STA    SUBSKP
 881 2314 8525              STA    SUBSKP+1
 882 2316 4CA121            JMP    EXSPEC        ; GO TO END OF EXECUTABLE SPEC PROCESSING
 883 2319 A913      ESBP1:  LDA    #X'13         ; GENERATE ER-12 IF ESB WITHOUT SUB
 884 231B 201300            JSR    ERR
 885 231E 4CA121            JMP    EXSPEC        ; DO NOTHING
 886
 887                ;       GETARG GET ARGUMENT, CHECK FOR OVERFLOW, A=0 IF OK, NOT ZERO
 888                ;       IF OVEFFLOW ERROR
 889
 890 2321 20D225    GETARG: JSR    NNB           ; FIRST MOVE TO NEXT NON-BLANK CHARACTER
 891 2324 20B724            JSR    NCOL          ; COLLECT WHAT IS ASSUMED TO BE AN ARGUMENT
 892 2327 A52C              LDA    NMERR         ; GET ERROR FLAG FROM NCOL
 893 2329 60                RTS                  ; RETURN WITH ERROR CODE IN A
 894
```



```
                           .PAGE  'MISCELLANEOUS LITTLE OUTPUT ROUTINES'
 895
 896                ;       OULOC  OUTPUT LOCATION  PLACE IN LISTING THE CURRENT VALUE OF
 897                ;              THE OBJECT CODE POINTER AS 4 DIGIT HEX FOLLOWED BY 2
 898                ;              BLANKS
 899                ;              NO REGISTERS BOTHERED
 900
 901 232A 48        OULOC:  PHA                 ; SAVE A
 902 232B A51A              LDA    CODEPT+1     ; OUTPUT UPPER BYTE FIRST FOR NATURAL ORDER
 903 232D 203F23            JSR    OUHX
 904 2330 A519              LDA    CODEPT       ; OUTPUT LOWER BYTE
 905 2332 203F23            JSR    OUHX
 906 2335 A920              LDA    #' '         ; FOLLOW WITH 2 BLANKS
 907 2337 205D23            JSR    OUCH
 908 233A 205D23            JSR    OUCH
 909 233D 68                PLA                 ; RESTORE A
 910 233E 60                RTS                 ; RETURN
 911
 912
 913                ;       OUHX   OUTPUT CONTENTS OF A AS 2 HEX DIGITS
 914                ;       NO REGISTERS BOTHERED
 915
 916 233F 48        OUHX:   PHA                 ; SAVE A
 917 2340 4A                LSRA                ; RIGHT JUSTIFY HIGH 4 BITS
 918 2341 4A                LSRA
 919 2342 4A                LSRA
 920 2343 4A                LSRA
 921 2344 205023            JSR    OUHX1        ; OUTPUT AS HEX DIGIT
 922 2347 68                PLA                 ; RESTORE A
 923 2348 48                PHA                 ; BUT KEEP IT ON THE STACK
 924 2349 290F              AND    #X'0F        ; ISOLATE LOW 4 BITS AND OUTPUT AS HEX
 925 234B 205023            JSR    OUHX1
 926 234E 68                PLA                 ; RESTORE A
 927 234F 60                RTS                 ; RETURN
 928
 929 2350 18        OUHX1:  CLC                 ; CONVERT 4 BIT NUMBER IN A TO ASCII HEX
 930 2351 6930              ADC    #'0'         ; DIGIT
 931 2353 C93A              CMP    #'9'+1
 932 2355 3002              BMI    OUHX2
 933 2357 6906              ADC    #'A'-'0'-X'A-1
 934 2359 205D23    OUHX2:  JSR    OUCH         ; OUTPUT THE ASCII HEX DIGIT
 935 235C 60                RTS                 ; RETURN
 936
 937                ;       OUCH   OUTPUT ASCII CHARACTER IN A WITH AUTOMATIC NEW LINE IF
 938                ;              LINE IS TOO LONG AND LISTING ENABLED
 939                ;              NO REGISTERS BOTHERED
 940
 941 235D 48        OUCH:   PHA                 ; SAVE CHARACTER TO OUTPUT
 942 235E 8A                TXA                 ; AND INDEX X
 943 235F 48                PHA
 944 2360 98                TYA                 ; AND INDEX Y
 945 2361 48                PHA
 946 2362 A418              LDY    OUBFPT       ; TEST IF 72 CHARACTERS ALREADY IN OUTPUT
 947 2364 C048              CPY    #72          ; BUFFER
 948 2366 9010              BCC    OUCH1        ; JUMP AHEAD IF LESS THAN 72
```

NOTCMP KIM NOTRAN COMPILER
MISCELLANEOUS LITTLE OUTPUT ROUTINES

```
 949 2368 A90D              LDA   #X'0D        ; PUT A CARRIAGE RETURN AND LINE FEED IN
 950 236A 9102              STA   (OUBFA),Y    ; THE OUTPUT BUFFER
 951 236C C8                INY
 952 236D A90A              LDA   #X'0A
 953 236F 9102              STA   (OUBFA),Y
 954 2371 20F025            JSR   OU           ; PRINT THE CONTENTS OF THE OUTPUT BUFFER
 955 2374 A000              LDY   #0           ; REINITIALIZE OUTPUT BUFFER POINTER
 956 2376 8418              STY   OUBFPT
 957 2378 BA        OUCH1:  TSX                ; RESTORE CHARACTER TO BE OUTPUT
 958 2379 BD0301            LDA   X'103,X
 959 237C 9102              STA   (OUBFA),Y    ; PUT IT IN THE OUTPUT BUFFER
 960 237E E618              INC   OUBFPT       ; INCREMENT OUTPUT BUFFER POINTER
 961 2380 68                PLA                ; RESTORE REGISTERS
 962 2381 A8                TAY
 963 2382 68                PLA
 964 2383 AA                TAX
 965 2384 68                PLA
 966 2385 60                RTS                ; RETURN
 967
```

NOTCMP KIM NOTRAN COMPILER
OBJECT CODE OUTPUT ROUTINE

```
                            .PAGE  'OBJECT CODE OUTPUT ROUTINE'
 968                ;       CODOU  CODE OUTPUT   ADDS A BYTE OF OBJECT CODE TO THE OBJECT
 969                ;              CODE AREA IN MEMORY AND ALSO CONVERTS IT TO HEX AND ADDS
 970                ;              TWO HEX DIGITS FOLLOWED BY A BLANK TO THE LISTING OUTPUT
 971                ;              BUFFER.  GENERATES AN ERROR MESSAGE IF THE OBJECT CODE
 972                ;              AREA OVERFLOWS
 973                ;              ENTER WITH CODE BYTE IN A
 974                ;              PRESERVES ALL REGISTERS
 975                ;              ON CODE OVERFLOW ERROR, A NORMAL RETURN IS TAKEN SO ONCE
 976                ;              GENERATED, THE ERROR WILL REPEAT UNTIL THE COMPILATION
 977                ;              IS INTERRUPTED
 978                ;              NO REGISTERS BOTHERED
 979
 980 2386 48        CODOU:  PHA                ; SAVE REGISTERS
 981 2387 8A                TXA
 982 2388 48                PHA
 983 2389 98                TYA
 984 238A 48                PHA
 985 238B A509              LDA   CODELN+1     ; TEST IF OBJECT CODE AREA IS FULL
 986 238D C51C              CMP   CODECT+1
 987 238F D00E              BNE   CODOU1       ; JUMP IF NOT
 988 2391 A508              LDA   CODELN
 989 2393 C51B              CMP   CODECT
 990 2395 D008              BNE   CODOU1       ; JUMP IF NOT
 991 2397 A905              LDA   #5           ; GENERATE ER 5 IF FULL
 992 2399 20F925            JSR   ER
 993 239C 4CB923            JMP   CODOU2       ; GO RESTORE REGISTERS AND RETURN
 994
 995 239F BA        CODOU1: TSX                ; GET THE CODE BYTE BACK
 996 23A0 BD0301            LDA   X'103,X
 997 23A3 A000              LDY   #0           ; STORE CODE BYTE IN OBJECT CODE AREA
 998 23A5 9119              STA   (CODEPT),Y
 999 23A7 203F23            JSR   OUHX         ; PRINT THE CODE BYTE IN HEX FOLLOWED BY A
1000 23AA A920              LDA   #' '         ; BLANK
1001 23AC 205D23            JSR   OUCH
1002 23AF A219              LDX   #CODEPT      ; INCREMENT OBJECT CODE POINTER
1003 23B1 20E625            JSR   DINC
1004 23B4 A21B              LDX   #CODECT      ; INCREMENT OBJECT CODE BYTE COUNT
1005 23B6 20E625            JSR   DINC
1006 23B9 68        CODOU2: PLA                ; RESTORE REGISTERS
1007 23BA A8                TAY
1008 23BB 68                PLA
1009 23BC AA                TAX
1010 23BD 68                PLA
1011 23BE 60                RTS                ; RETURN
1012
```

```
                                .PAGE  'NOTE COLLECT ROUTINE'
1013                    ;       NTCOL  NOTE COLLECT ROUTINE
1014                    ;              ENTER WITH INBFPT POINTING TO WHAT IS BELIEVED TO BE A
1015                    ;              NOTE SPECIFICATION
1016                    ;              EXIT IWTH THE FOLLOWING BASE PAGE VARIABLES SET:
1017                    ;              NTERR - NTCOL ERROR CODE, 0 = NO ERROR
1018                    ;              NTVOIC - VOICE NUMBER, ZERO IF NOT EXPLICITLY SPECIFIED
1019                    ;              NTDUR - NOTE DURATION, 1-15 AS IN OBJECT CODE
1020                    ;              NTPTC - NOTE PITCH, 0-12, C=1, B=12, REST=0
1021                    ;              NTOCT - NOTE OCTAVE, 1-6, 0 IF NOT SPECIFIED
1022                    ;              INBFPT POINTING TO CHARACTER FOLLOWING NOTE
1023                    ;              SPECIFICATION, EITHER BLANK, ;, OR CR FOR NO ERROR
1024                    ;              X AND Y REGISTERS PRESERVED
1025
1026 23BF 8A       NTCOL:  TXA                    ; SAVE X AND Y REGISTERS
1027 23C0 48               PHA
1028 23C1 98               TYA
1029 23C2 48               PHA
1030 23C3 A900             LDA   #0            ; INITIALIZE SOME RETURN ARGUMENTS
1031 23C5 852B             STA   NTERR
1032 23C7 8526             STA   NTVOIC
1033 23C9 852A             STA   NTOCT
1034 23CB 8529             STA   NTPTC
1035 23CD A417             LDY   INBFPT
1036 23CF B100             LDA   (INBFA),Y     ; GET CURRENT CHARACTER
1037 23D1 C935             CMP   #'4'+1        ; TEST IF A VALID VOICE # DIGIT
1038 23D3 B009             BCS   NTCOL1        ; JUMP IF NOT
1039 23D5 C931             CMP   #'1'
1040 23D7 9005             BCC   NTCOL1        ; JUMP IF NOT
1041 23D9 E930             SBC   #'0'          ; IF A VALID VOICE # DIGIT, STORE IN BINARY
1042 23DB 8526             STA   NTVOIC        ; AT NTVOIC
1043 23DD C8               INY                 ; SKIP OVER THE VOICE NUMBER DIGIT
1044 23DE B100     NTCOL1: LDA   (INBFA),Y     ; TEST IF VALID NOTE LETTER
1045 23E0 C948             CMP   #'G'+1
1046 23E2 B038             BCS   NTCOL5        ; JUMP IF NOT
1047 23E4 C941             CMP   #'A'
1048 23E6 9034             BCC   NTCOL5        ; JUMP IF NOT
1049 23E8 E940             SBC   #'A'-1        ; CONVERT NOTE LETTER INTO A NUMBER; A=1,
1050 23EA 8529             STA   NTPTC         ; G=7 AND MULTIPLY IT BY 3
1051 23EC 0A               ASLA
1052 23ED 6529             ADC   NTPTC
1053 23EF 8529             STA   NTPTC         ; PLACE IT IN NTPTC UNTIL SHARPS AND FLATS
1054                                           ; EVALUATED
1055 23F1 C8               INY                 ; SKIP OVER THE NOTE LETTER
1056 23F2 B100             LDA   (INBFA),Y     ; GET NEXT CHARACTER
1057 23F4 C923             CMP   #'#'          ; TEST IF A SHARP
1058 23F6 D004             BNE   NTCOL2        ; JUMP IF NOT
1059 23F8 E629             INC   NTPTC         ; INCREMENT NTPTC IF SHARP
1060 23FA D006             BNE   NTCOL3
1061 23FC C940     NTCOL2: CMP   #'@'          ; TEST IF FLAT
1062 23FE D003             BNE   NTCOL4        ; JUMP IF NOT
1063 2400 C629             DEC   NTPTC         ; DECREMENT NTPTC IF FLAT
1064 2402 C8       NTCOL3: INY                 ; SKIP OVER # OR @ IF PRESENT
1065 2403 A629     NTCOL4: LDX   NTPTC         ; CONVERT VALUE IN NTPTC INTO FINAL FORM
1066 2405 BD7824           LDA   NTPTCT-2,X
```


NOTCMP KIM NOTRAN COMPILER
NOTE COLLECT ROUTINE

```
1067 2408 8529             STA   NTPTC
1068 240A B100             LDA   (INBFA),Y     ; TEST IF VALID OCTAVE DIGIT PRESENT
1069 240C C937             CMP   #'6'+1
1070 240E B018             BCS   NTCOL7        ; JUMP IF NOT
1071 2410 C931             CMP   #'1'
1072 2412 9014             BCC   NTCOL7        ; JUMP IF NOT
1073 2414 E930             SBC   #'0'          ; CONVERT THE DIGIT TO BINARY AND PLACE IN
1074 2416 852A             STA   NTOCT         ; NTOCT
1075 2418 C8               INY                 ; SKIP OVER THE OCTAVE DIGIT
1076 2419 4C2824           JMP   NTCOL7        ; GO TO DURATION INTERPRET
1077 241C C952     NTCOL5: CMP   #'R'          ; TEST IF REST SPECIFICATION
1078 241E F007             BEQ   NTCOL6        ; JUMP IF SO
1079 2420 A906     NTCOLO: LDA   #6            ; ER 6 IF NOT
1080 2422 852B             STA   NTERR
1081 2424 4C6D24           JMP   NTCOLD        ; GO RETURN
1082 2427 C8       NTCOL6: INY                 ; SKIP OVER THE R IF PRESENT, NTPTC WILL BE
1083                                           ; ZERO SIGNIFYING A REST
1084
1085 2428 A201     NTCOL7: LDX   #1            ; SCAN NTDURT TO DETERMINE IF CURRENT CHAR.
1086 242A BD8E24   NTCOL8: LDA   NTDURT-1,X     ; IS A VAILD DURATION
1087 242D F045             BEQ   NTCOLE        ; JUMP OUT IF END OF TABLE, ILLEGAL DUR.
1088 242F D100             CMP   (INBFA),Y
1089 2431 F003             BEQ   NTCOL9        ; JUMP OUT IF MATCH
1090 2433 E8               INX                 ; INDEX TO NEXT TABLE ENTRY
1091 2434 D0F4             BNE   NTCOL8        ; LOOP TO TRY NEXT ENTRY
1092 2436 8A       NTCOL9: TXA                 ; GET 3 TIMES X AND PUT IN NTDUR
1093 2437 8527             STA   NTDUR
1094 2439 0A               ASLA
1095 243A 6527             ADC   NTDUR
1096 243C 8527             STA   NTDUR
1097 243E C8               INY                 ; SKIP OVER THE DURATION LETTER
1098 243F B100             LDA   (INBFA),Y     ; GET CURRENT CHARACTER
1099 2441 C92E             CMP   #'.'          ; TEST IF DURATION MODIFIER "."
1100 2443 D004             BNE   NTCOLA        ; JUMP IF NOT
1101 2445 C627             DEC   NTDUR         ; DECREMENT NTDUR IF A .
1102 2447 D006             BNE   NTCOLB
1103 2449 C933     NTCOLA: CMP   #'3'          ; TEST IF DURATION MODIFIER "3" (TRIPLET)
1104 244B D003             BNE   NTCOLC        ; JUMP IF NOT
1105 244D E627             INC   NTDUR         ; INCREMENT NTDUR IF A 3
1106 244F C8       NTCOLB: INY                 ; SKIP OVER THE "." OR "3" IF PRESENT
1107
1108 2450 A627     NTCOLC: LDX   NTDUR         ; CONVERT NTDUR INTO FINAL OBJECT FORM
1109 2452 BD9424           LDA   NTDRT1-2,X
1110 2455 F01D             BEQ   NTCOLE        ; JUMP IF ILLEGAL DURATION
1111 2457 8527             STA   NTDUR
1112 2459 AA               TAX                 ; CONVERT NTDUR TO ABSOLUTE TIME VALUE FOR
1113 245A BDA724           LDA   NTDRT2-1,X    ; COMPILER
1114 245D 8528             STA   NTDURX
1115 245F B100             LDA   (INBFA),Y     ; TEST IF VALID NOTE SPEC TERMINATOR
1116 2461 C920             CMP   #' '          ; BLANK
1117 2463 F008             BEQ   NTCOLD
1118 2465 C93B             CMP   #';'          ; SEMICOLON
1119 2467 F004             BEQ   NTCOLD
1120 2469 C90D             CMP   #X'0D         ; CARRIAGE RETURN
1121 246B D0B3             BNE   NTCOLO        ; SIGNAL ER 6 IF NOT VALID TERMINATOR
```

NOTCMP KIM NOTRAN COMPILER
NOTE COLLECT ROUTINE

```
1122 246D 8417      NTCOLD:  STY    INBFPT       ; UPDATE INBFPT
1123 246F 68                 PLA                  ; RESTORE X AND Y
1124 2470 A8                 TAY
1125 2471 68                 PLA
1126 2472 AA                 TAX
1127 2473 60                 RTS                  ; RETURN
1128
1129 2474 A909      NTCOLE:  LDA    #9           ; SIGNAL ER 9 IF ILLEGAL DURATION
1130 2476 852B               STA    NTERR
1131 2478 D0F3               BNE    NTCOLD       ; GO RETURN
1132
1133                ;        NOTE PITCH TRANSLATE TABLE
1134
1135 247A 090A0B    NTPTCT:  .BYTE  9,10,11      ; A@  A  A#
1136 247D 0B0C01             .BYTE  11,12,1       ; B@  B  B#
1137 2480 0C0102             .BYTE  12,1,2        ; C@  C  C#
1138 2483 020304             .BYTE  2,3,4         ; D@  D  D#
1139 2486 040506             .BYTE  4,5,6         ; E@  E  E#
1140 2489 050607             .BYTE  5,6,7         ; F@  F  F#
1141 248C 070809             .BYTE  7,8,9         ; G@  G  G#
1142
1143                ;        DURATION TRANSLATE TABLES
1144
1145                ;        LIST OF VALID DURATION LETTERS
1146 248F 574851    NTDURT:  .BYTE  'W','H','Q'   ; WHOLE, HALF, QUARTER
1147 2492 455354             .BYTE  'E','S','T'   ; EIGHTH, SIXTEENTH, THIRTY-SECOND
1148 2495 00                 .BYTE  0             ; END OF TABLE MARKER
1149
1150                ;        TRANSLATE DURATION LETTER AND MODIFIER TO DURATION CODE
1151 2496 000100    NTDRT1:  .BYTE  0,1,0         ; W. (ILLEGAL)  W   W3 (ILLEGAL)
1152 2499 020305             .BYTE  2,3,5         ; H.   H   H3
1153 249C 040608             .BYTE  4,6,8         ; Q.   Q   Q3
1154 249F 07090B             .BYTE  7,9,11        ; E.   E   E3
1155 24A2 0A0C0E             .BYTE  10,12,14      ; S.   S   S3
1156 24A5 0D0F00             .BYTE  13,15,0       ; T.   T   T3 (ILLEGAL)
1157
1158                ;        TRANSLATE DURATION CODE TO ABSOLUTE TIME VALUE
1159 24A8 C09060    NTDRT2:  .BYTE  192,144,96    ; W   H.  W3
1160 24AB 484030             .BYTE  72,64,48      ; Q.  H3  Q
1161 24AE 242018             .BYTE  36,32,24      ; E.  Q3  E
1162 24B1 12100C             .BYTE  18,16,12      ; S.  E3  S
1163 24B4 090806             .BYTE  9,8,6         ; T.  S3  T
1164
```

NOTCMP KIM NOTRAN COMPILER
NUMBER COLLECT ROUTINE

```
                            .PAGE  'NUMBER COLLECT ROUTINE'
1165                ;       NCOL   NUMBER COLLECT
1166                ;              ENTER WITH INBFPT POINTING TO A DIGIT.
1167                ;              EXIT WITH NMERR=0, NUMBER VALUE IN NBF, AND INBFPT
1168                ;              POINTING TO TERMINATOR (FIRST NON-DIGIT) IF A VALID
1169                ;              NUMBER FOUND (0-255).
1170                ;              EXIT WITH NMERR=ER1, NBF UNDEFINED, AND INBFPT POINTING
1171                ;              TO TERMINATOR IF OVERFLOW.
1172                ;              NO REGISTERS BOTHERED
1173
1174 24B7 48          NCOL:  PHA                  ; SAVE A
1175 24B8 98                TYA                  ; SAVE Y
1176 24B9 48                PHA
1177 24BA A900              LDA    #0
1178 24BC 8522              STA    NBF           ; CLEAR NUMBER BEING FORMED
1179 24BE 852C              STA    NMERR         ; CLEAR OVERFLOW FLAG
1180 24C0 A417              LDY    INBFPT
1181 24C2 B100      NCOL1:  LDA    (INBFA),Y     ; GET CHARACTER FROM INPUT BUFFER
1182 24C4 C930              CMP    #'0'
1183 24C6 9024              BCC    NCOL3         ; BRANCH IF NOT A DIGIT
1184 24C8 C93A              CMP    #'9'+1
1185 24CA B020              BCS    NCOL3         ; BRANCH IF NOT A DIGIT
1186 24CC A522              LDA    NBF           ; IF A DIGIT, FIRST MULTIPLY NBF BY 10
1187 24CE 0A                ASLA                 ; WITH OVERFLOW CHECK
1188 24CF B021              BCS    NCOL4
1189 24D1 0A                ASLA
1190 24D2 B01E              BCS    NCOL4
1191 24D4 6522              ADC    NBF
1192 24D6 B01A              BCS    NCOL4
1193 24D8 0A                ASLA
1194 24D9 B017              BCS    NCOL4
1195 24DB 8522              STA    NBF
1196 24DD B100              LDA    (INBFA),Y     ; GET NEW DIGIT
1197 24DF 69D0              ADC    #-'0'         ; CONVERT TO BINARY
1198 24E1 18                CLC
1199 24E2 6522              ADC    NBF           ; ADD TO NBF WITH OVERFLOW CHECK
1200 24E4 B00C              BCS    NCOL4
1201 24E6 8522              STA    NBF
1202 24E8 C8        NCOL2:  INY                  ; GO TO NEXT INPUT CHARACTER
1203 24E9 4CC224            JMP    NCOL1         ; GO FOR NEXT DIGIT
1204
1205 24EC 8417      NCOL3:  STY    INBFPT        ; UPDATE INBFPT
1206 24EE 68                PLA                  ; RESTORE REGISTERS
1207 24EF A8                TAY
1208 24F0 68                PLA
1209 24F1 60                RTS                  ; RETURN
1210
1211 24F2 A901      NCOL4:  LDA    #1            ; SET OVERFLOW ERROR CODE
1212 24F4 852C              STA    NMERR
1213 24F6 4CE824            JMP    NCOL2         ; CONTINUE SCANNING FOR END OF NUMBER
1214
```

```
                                .PAGE  'KEYWORD COLLECT AND MATCH ROUTINE'
1215                    ;       KWCOL  KEYWORD COLLECT AND MATCH
1216                    ;              ENTER WITH INBFPT POINTING TO A NON-BLANK CHARACTER.
1217                    ;              ROUTINE LOOKS AT THIS CHARACTER AND THE NEXT TWO.
1218                    ;              IF THE 3 CHARACTERS MATCH A KEYWORD, RETURN WITH X
1219                    ;              POINTING TO JUMP ADDRESS IN KEYWORD TABLE ENTRY AND C=1
1220                    ;              AND INBFPT POINTING TO CHARACTER FOLLOWING THE KEYWORD.
1221                    ;              IF NO MATCH, RETURN WITH C=0 AND INBFPT UNCHANGED.
1222                    ;              PRESERVES Y
1223
1224 24F9 98     KWCOL:  TYA                    ; SAVE Y
1225 24FA 48             PHA
1226 24FB A200           LDX   #0              ; ZERO POINTER INTO KEYWORD TABLE
1227 24FD 8644   KWCOL1: STX   KWTP
1228 24FF BD2925         LDA   KWTAB,X         ; TEST IF AT END OF KEYWORD TABLE
1229 2502 D003           BNE   KWCOL3          ; JUMP IF NOT
1230 2504 18             CLC                   ; CLEAR CARRY TO INDICATE FAILURE TO FIND
1231 2505 9016           BCC   KWCOL5          ; A MATCH AND RETURN
1232
1233 2507 A903   KWCOL3: LDA   #3              ; INITIALIZE MATCH COUNT
1234 2509 8545           STA   KWMC
1235 250B A417           LDY   INBFPT          ; GET POINTER TO CURRENT INPUT CHARACTER
1236 250D B100   KWCOL4: LDA   (INBFA),Y       ; TEST IF MATCH BETWEEN KEYWORD TABLE ENTRY
1237 250F DD2925         CMP   KWTAB,X         ; AND INPUT BUFFER CHARACTER
1238 2512 D00C           BNE   KWCOL6          ; JUMP IF NO MATCH
1239 2514 E8             INX                   ; IF MATCH, INCREMENT INDICES
1240 2515 C8             INY
1241 2516 C645           DEC   KWMC            ; TEST IF 3 CHARACTERS MATCHED
1242 2518 D0F3           BNE   KWCOL4          ; GO MATCH NEXT IF NOT
1243 251A 8417           STY   INBFPT          ; IF MATCH, UPDATE INBFPT
1244 251C 38             SEC                   ; SET CARRY TO INDICATE MATCH FOUND
1245 251D 68     KWCOL5: PLA                   ; RESTORE Y
1246 251E A8             TAY
1247 251F 60             RTS                   ; RETURN
1248
1249 2520 A544   KWCOL6: LDA   KWTP            ; UPDATE KWTP TO POINT TO NEXT KEYWORD IN
1250 2522 18             CLC                   ; TABLE
1251 2523 6905           ADC   #5
1252 2525 AA             TAX
1253 2526 4CFD24         JMP   KWCOL1          ; GO TRY TO MATCH NEXT KEYWORD
1254
1255                    ;       KEYWORD TABLE
1256
1257 2529 4E5643 KWTAB:  .BYTE  'N','V','C'    ; DEFINE NUMBER OF VOICES
1258 252C DD21           .WORD  NVCP
1259 252E 414354         .BYTE  'A','C','T'    ; ACTIVATE VOICES
1260 2531 F821           .WORD  ACTP
1261 2533 444354         .BYTE  'D','C','T'    ; DEACTIVATE VOICES
1262 2536 FC21           .WORD  DCTP
1263 2538 574156         .BYTE  'W','A','V'    ; ASSIGN WAVEFORM
1264 253B 3122           .WORD  WAVP
1265 253D 4A5352         .BYTE  'J','S','R'    ; JUMP TO SUBROUTINE
1266 2540 5D22           .WORD  JSRP
1267 2542 4A4D50         .BYTE  'J','M','P'    ; JUMP
1268 2545 6122           .WORD  JMPP
```



```
1269 2547 525453         .BYTE  'R','T','S'    ; RETURN FROM SUBROUTINE
1270 254A 9322           .WORD  RTSP
1271 254C 54504F         .BYTE  'T','P','O'    ; SET TEMPO
1272 254F B222           .WORD  TPOP
1273 2551 454E44         .BYTE  'E','N','D'    ; END OF SONG
1274 2554 9B22           .WORD  ENDP
1275 2556 414253         .BYTE  'A','B','S'    ; FORCE ABSOLUTE PITCH UPDATE
1276 2559 C822           .WORD  ABSP
1277 255B 535542         .BYTE  'S','U','B'    ; DEFINE BEGINNING OF SUBROUTINE AREA
1278 255E D422           .WORD  SUBP
1279 2560 455342         .BYTE  'E','S','B'    ; DEFINE END OF SUBROUTINE AREA
1280 2563 FA22           .WORD  ESBP
1281 2565 00             .BYTE  0              ; END OF TABLE MARKER
1282
```

NOTCMP KIM NOTRAN COMPILER
SYMBOL TABLE SEARCH

```
                                .PAGE   'SYMBOL TABLE SEARCH'
1283                    ;       SYMSR   SYMBOL TABLE SEARCH
1284                    ;               ENTER WITH SYMBOL TO SEARCH FOR IN A RANGE OF 1-255.
1285                    ;               EXIT WITH SYMTPT POINTING TO LOW BYTE OF SYMBOL VALUE
1286                    ;               AND C=1 IF SYMBOL IS FOUND, SYMTPT POINTING TO END OF
1287                    ;               SYMBOL TABLE AND C=0 IF NOT FOUND.  NO REGISTERS
1288                    ;               AFFECTED.
1289
1290 2566 48            SYMSR:  PHA                     ; SAVE A
1291 2567 8A                    TXA                     ; SAVE X
1292 2568 48                    PHA
1293 2569 98                    TYA                     ; SAVE Y
1294 256A 48                    PHA
1295 256B A504                  LDA     SYMTA           ; COPY SYMTA INTO SYMTPT
1296 256D 851D                  STA     SYMTPT
1297 256F A505                  LDA     SYMTA+1
1298 2571 851E                  STA     SYMTPT+1
1299 2573 A000                  LDY     #0              ; ZERO Y FOR STRAIGHT INDIRECT
1300 2575 B11D          SYMSR0: LDA     (SYMTPT),Y      ; TEST IF AT END OF SYMBOL TABLE
1301 2577 D003                  BNE     SYMSR1          ; BRANCH IF NOT
1302 2579 18                    CLC                     ; CLEAR CARRY TO INDICATE SEARCH FAILURE
1303 257A 900C                  BCC     SYMSR2          ; GO RESTORE REGISTERS AND RETURN
1304 257C BA            SYMSR1: TSX                     ; LOAD STACK POINTER INTO X TO MAKE SAVED
1305                                                    ; SYMBOL ACCESSABLE
1306 257D DD0301                CMP     X'103,X         ; COMPARE WITH SYMBOL BEING SEARCHED FOR
1307 2580 D00C                  BNE     SYMSR3          ; BRANCH IF NO MATCH
1308 2582 A21D                  LDX     #SYMTPT         ; INCREMENT SYMTPT TO POINT TO VALUE IF
1309 2584 20E625                JSR     DINC            ; MATCH
1310 2587 38                    SEC                     ; SET CARRY TO INDICATE SEARCH SUCCESS
1311 2588 68            SYMSR2: PLA                     ; RESTORE REGISTERS
1312 2589 A8                    TAY
1313 258A 68                    PLA
1314 258B AA                    TAX
1315 258C 68                    PLA
1316 258D 60                    RTS                     ; RETURN
1317
1318 258E A21D          SYMSR3: LDX     #SYMTPT         ; INCREMENT SYMTPT BY 3 TO POINT TO NEXT
1319 2590 20E625                JSR     DINC            ; SYMBOL
1320 2593 20E625                JSR     DINC
1321 2596 20E625                JSR     DINC
1322 2599 4C7525                JMP     SYMSR0          ; GO CHECK NEXT SYMBOL
1323
```

NOTCMP KIM NOTRAN COMPILER
SYMBOL TABLE ADD

```
                                .PAGE   'SYMBOL TABLE ADD'
1324                    ;       SYMAD   SYMBOL TABLE ADD ROUTINE
1325                    ;               ENTER WITH SYMBOL TO ADD IN A, VALUE OF SYMBOL IN CODEPT
1326                    ;               SYMTPT POINTING TO END OF SYMBOL TABLE
1327                    ;               RETURN WITH C=1 FOR SUCCESSFUL ADD, C=0 FOR SYMBOL TABLE
1328                    ;               OVERFLOW.
1329
1330 259C 48            SYMAD:  PHA                     ; SAVE A
1331 259D 8A                    TXA                     ; SAVE X
1332 259E 48                    PHA
1333 259F 98                    TYA                     ; SAVE Y
1334 25A0 48                    PHA
1335 25A1 A51F                  LDA     SYMTCT          ; TEST FOR TABLE OVERFLOW
1336 25A3 C50A                  CMP     SYMTLN
1337 25A5 9007                  BCC     SYMAD2          ; JUMP IF OK
1338 25A7 18                    CLC                     ; CLEAR CARRY TO INDICATE UNSUCCESSFUL ADD
1339 25A8 68            SYMAD1: PLA                     ; RESTORE REGISTERS
1340 25A9 A8                    TAY
1341 25AA 68                    PLA
1342 25AB AA                    TAX
1343 25AC 68                    PLA
1344 25AD 60                    RTS                     ; RETURN
1345
1346 25AE E61F          SYMAD2: INC     SYMTCT          ; INCREMENT SYMBOL COUNT
1347 25B0 A000                  LDY     #0              ; ZERO Y FOR STRAIGHT INDIRECT
1348 25B2 BA                    TSX                     ; COPY STACK POINTER TO X TO MAKE SAVED
1349                                                    ; SYMBOL ACCESSABLE
1350 25B3 BD0301                LDA     X'103,X         ; ADD THE SYMBOL TO THE TABLE
1351 25B6 911D                  STA     (SYMTPT),Y
1352 25B8 A21D                  LDX     #SYMTPT         ; INCREMENT SYMBOL TABLE POINTER
1353 25BA 20E625                JSR     DINC
1354 25BD A519                  LDA     CODEPT          ; ADD LOW VALUE TO TABLE
1355 25BF 911D                  STA     (SYMTPT),Y
1356 25C1 20E625                JSR     DINC
1357 25C4 A51A                  LDA     CODEPT+1        ; ADD HIGH VALUE TO TABLE
1358 25C6 911D                  STA     (SYMTPT),Y
1359 25C8 20E625                JSR     DINC
1360 25CB A900                  LDA     #0              ; SET END OF TABLE FLAG
1361 25CD 911D                  STA     (SYMTPT),Y
1362 25CF 38                    SEC                     ; SET CARRY TO INDICATE SUCCESSFUL ADD
1363 25D0 B0D6                  BCS     SYMAD1          ; GO RESTORE REGISTERS AND RETURN
1364
```

```
                                .PAGE  'MISCELLANEOUS ROUTINES'
 1365                   ;       NNB    NEXT NON-BLANK SEARCH
 1366                   ;              ENTER WITH INBFPT POINTING TO FIRST BYTE TO BE SEARCHED.
 1367                   ;              EXIT WITH INBFPT POINTING TO FIRST NON-BLANK CHARACTER
 1368                   ;              ALL REGISTERS PRESERVED
 1369
 1370 25D2 48           NNB:   PHA                    ; SAVE A AND Y
 1371 25D3 98                  TYA
 1372 25D4 48                  PHA
 1373 25D5 A417                LDY    INBFPT         ; GET CURRENT CHARACTER POINTER
 1374 25D7 B100         NNB1:  LDA    (INBFA),Y      ; GET CURRENT CHARACTER
 1375 25D9 C920                CMP    #' '           ; TEST IF A BLANK
 1376 25DB D003                BNE    NNB2           ; JUMP IF NOT
 1377 25DD C8                  INY                   ; IF A BLANK, GO TO NEXT CHARACTER
 1378 25DE D0F7                BNE    NNB1           ; GO TEST NEXT CHARACTER
 1379 25E0 8417         NNB2:  STY    INBFPT         ; RETURN WITH CHARACTER IN A WHEN NON-BLANK
 1380 25E2 68                  PLA                   ; FOUND
 1381 25E3 A8                  TAY                   ; RESTORE Y AND A
 1382 25E4 68                  PLA
 1383 25E5 60                  RTS                   ; RETURN
 1384
 1385
 1386                   ;       DINC   DOUBLE INCREMENT
 1387                   ;              ENTER WITH X POINTING TO THE LOW BYTE OF THE WORD TO
 1388                   ;              INCREMENT, HIGH BYTE IS AT X+1 (WORD IN BASE PAGE)
 1389                   ;              NO REGISTERS AFFECTED
 1390
 1391 25E6 F600         DINC:  INC    0,X            ; INCREMENT LOW BYTE
 1392 25E8 D002                BNE    DINC1          ; SKIP AHEAD IF NO CARRY OUT
 1393 25EA F601                INC    1,X
 1394 25EC 60           DINC1: RTS                    ; RETURN
 1395
 1396
 1397
 1398                   ;       LINKS TO USER SUPPLIED INPUT AND OUTPUT ROUTINES
 1399
 1400 25ED 6C0B00       IN:    JMP    (INP)          ; INPUT A LINE TO INPUT BUFFER
 1401 25F0 6C0F00       OU:    JMP    (OUT)          ; OUTPUT A LINE FROM OUTPUT BUFFER
 1402 25F3 6C0D00       INIT:  JMP    (INPINT)       ; INITIALIZE INPUT ROUTINE IN PREP FOR
 1403                                                ; FIRST LINE
 1404 25F6 6C1100       OUIT:  JMP    (OUTINT)       ; INITIALIZE OUTPUT ROUTINE IN PREP FOR
 1405                                                ; FIRST LINE
 1406
 1407                   ;       ER     ERROR ROUTINE
 1408                   ;              IF LISTING IS ENABLED, A MESSAGE OF THE FORM "ER X" IS
 1409                   ;              INSERTED IN THE OBJECT CODE LISTING.
 1410                   ;              IF LISTING IW DISABLED, A CALL TO A USER SUPPLIED ERROR
 1411                   ;              ROUTINE IS PERFORMED WITH THE ERROR CODE IN A
 1412                   ;              NO REGISTERS ARE BOTHERED
 1413
 1414 25F9 48           ER:    PHA                    ; SAVE THE ERROR CODE
 1415 25FA 8A                  TXA                    ; SAVE OTHER REGISTERS IN CASE USER ERROR
 1416 25FB 48                  PHA                    ; ROUTINE USES THEM
 1417 25FC 98                  TYA
 1418 25FD 48                  PHA
```



```
 1419 25FE BA                  TSX                    ; COPY STACK POINTER INTO X SO THAT ERROR
 1420                                                 ; CODE MAY BE RETRIEVED
 1421 25FF A515                LDA    LIST            ; TEST IF LISTING ENABLED
 1422 2601 F020                BEQ    ER2             ; JUMP AHEAD TO USER ERROR ROUTINE IF NOT
 1423 2603 A945                LDA    #'E'            ; PUT ER- IN LISTING
 1424 2605 205D23              JSR    OUCH
 1425 2608 A952                LDA    #'R'
 1426 260A 205D23              JSR    OUCH
 1427 260D A92D                LDA    #'-'
 1428 260F 205D23              JSR    OUCH
 1429 2612 BD0301              LDA    X'103,X         ; RESTORE ERROR CODE
 1430 2615 203F23              JSR    OUHX            ; OUTPUT AS TWO HEX DIGITS
 1431 2618 A920                LDA    #' '            ; FOLLOWED BY A BLANK
 1432 261A 205D23              JSR    OUCH
 1433 261D 68           ER1:   PLA                    ; RESTORE REGISTERS
 1434 261E A8                  TAY
 1435 261F 68                  PLA
 1436 2620 AA                  TAX
 1437 2621 68                  PLA
 1438 2622 60                  RTS                    ; RETURN
 1439
 1440 2623 BD0301       ER2:   LDA    X'103,X         ; GET ERROR CODE BACK
 1441 2626 202C26              JSR    ER3
 1442 2629 4C1D26              JMP    ER1             ; GO RESTORE REGISTERS AND RETURN
 1443
 1444 262C 6C1300       ER3:   JMP    (ERR)           ; GO TO USER SUPPLIED ERROR ROUTINE
 1445
 1446
 1447 0000              CMPEND: .END
NO ERROR LINES
```