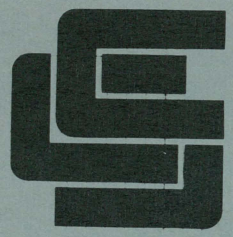


# PETUNIA OPERATING INSTRUCTIONS

by

## CALIFORNIA COMPUTER SYSTEMS

PET is a registered trademark of Commodore Business Machines

COPYRIGHT 1978 BY HUH ELECTRONICS. COPYRIGHT 1979 BY CALIFORNIA COMPUTER SYSTEMS - Santa Clara, California. ALL RIGHTS RESERVED. SECOND PRINTING

IM 108-2

# PETUNIA OPERATING INSTRUCTIONS

## REQUIREMENTS

First you will need a Commodore PET computer and the PETUNIA board. You will also need an amplifier and speaker combination of some kind, as well as a standard RCA to RCA male patch cord. The amplifier and speaker combination can be your home stereo system or anything similar as long as it has auxiliary inputs. The patch cord should be available in varying lengths, from any electronics dealer. Buy a cord long enough to reach from your PET to the amp without the cord clotheslining across the room.

## INSTALLATION

On the back of your PET, you will find three connector edges of the board: one with 6 fingers and two with 12. The six finger connector is the second cassette connector and the one with 12 fingers right next to it is the User Port connector. The PETUNIA plugs onto both connectors simultaneously. To plug the PETUNIA onto the connectors, MAKE SURE THE POWER IS OFF! Then carefully plug the PETUNIA board component side up onto the two connectors. This is all that is necessary to connect the PETUNIA to the PET.

Next take the patchcord and plug it into the RCA jack on the PETUNIA labeled AUDIO. Plug the other end of this patchcord into your amplifier / receiver, making sure the volume is all the way down. It should plug into the jack marked "AUX" or "AUXILIARY". If there is no jack so labeled, you could use ones marked "TUNER" or TAPE IN. Consult the owner's manual for your particular set. Turn up the volume on your amp a tiny bit; the PETUNIA can get pretty loud on some systems.

## OPERATING PROCEDURE

The PETUNIA is actually an 8 bit Digital to Analog Converter (DAC). The software provided is a subroutine that takes one to four note values and a note duration, and plays them by rapidly changing the output of the DAC. Note: it is up to you to write a program to feed notes to the subroutine. The subroutine plays the note for the duration specified then returns to the program. The program can be written either in BASIC or machine language.

You must load the subroutine called PETUNIA into your PET. The subroutine occupies memory from 1000 to 1D77 HEX, or 7168 to 7424 decimal. The cassette provided is actually a BASIC program (also called PETUNIA), that pokes the machine code into those high memory locations.

Load the cassette into your PET as you would any other BASIC program, then type "RUN". The program will poke the routine into memory and then protect it by poking 135,28. This tells BASIC not to write over the subroutine. The program will tell you when it's finished. Delete the program by typing "NEW". Then load in your music playing program.

It is a very good idea to make a back-up copy of the program on a separate cassette. When you're done, you'll want to restore the protected subroutine area back to BASIC, so it may be used by other programs. To do this, poke 135,32.

## HOW TO MAKE MUSIC

As stated earlier, the subroutine provided merely plays notes programmed into it. It is necessary for you to program the notes.

Programming the notes consists of loading 2 numerical values into 2 memory locations for each note. Four notes are possible at a time, requiring a total of eight memory locations. To determine the numerical value which corresponds to each note in the scale, refer to the chart provided in the appendix.

It is also necessary to tell the subroutine how long to play each note. This is called the DURATION value, and it must also be programmed into memory. The decimal value 64 corresponds to a quarter note. Other values can be found by experimenting. Finally two memory locations must be set to determine the overall TEMPO of the piece; these are the TEMPO values. The addresses of the various memory locations are shown in the following table:

LOCATION

| Hex | Decimal | Function |
|---|---|---|
| 1D0C | 7436 | Voice 1 low byte |
| 1D0D | 7437 | Voice 1 high byte |
| 1D0E | 7438 | Voice 2 low byte |
| 1D0F | 7439 | Voice 2 high byte |
| 1D10 | 7440 | Voice 3 low byte |
| 1D11 | 7441 | Voice 3 high byte |
| 1D12 | 7442 | Voice 4 low byte |
| 1D13 | 7443 | Voice 4 high byte |
| 1D14 | 7444 | Note duration |
| 1D15 | 7445 | Tempo low byte |
| 1D16 | 7446 | Tempo high byte |

The PETUNIA is a DAC connected to the PET's User Port. It is necessary to set the user port direction to all outputs. So your music playing program should do this by POKE-ing decimal location 59459 with 255. The BASIC statement would look like:

```
10 POKE 59459,255
```

To summarize, you must write a program which puts note values as well as timing information into memory locations and then jumps to the subroutine to play the notes. When the note is done playing the subroutine will return to your BASIC program which should then set up the next notes ..... on and on until the song is finished.

To call the subroutine, the Basic statement would be:
(line #) SYS(7447).

A sample program to play the note C3 and then the note C4 is shown following. The note values are taken from the table in the appendix.

```
10  REM SAMPLE NOTE PLAYING PROGRAM TO PLAY C3 & C4
20  POKE 59459,255
30  REM FIRST WE SET C3 INTO THE VOICE 1 LOCATIONS
40  POKE 7436,209 : POKE 7437,3
50  REM NEXT WE WILL SET THE DURATION
60  POKE 7444,64
70  REM AND LASTLY THE TEMPO
80  POKE 7445,52 : POKE 7446,0
90  REM  AND THEN WE PLAY THE NOTE
100 SYS(7447)
110 REM  SUBROUTINE RETURNS HERE AND WE SET UP C4
120 POKE 7436,163 : POKE 7437,7
130 REM  WE'LL LET THE TIMING STAY THE SAME
140 SYS(7447)
150 REM  THAT'S IT!!
160 END
```

Here is another program using the Petunia. It will randomly generate all combinations of voices, notes, durations, and tempos. If you want to, you can alter the constants A thru F in order to limit the number of combinations.

```
10  Print "♡": POKE 59459,255
20  REM - RANDOM NOTE GENERATOR
30  REM - BY CALIFORNIA COMPUTER SYSTEMS
40  FOR Z=0 TO 10: POKE(7436+Z),0: NEXT Z
50  F=2*(INT(4*RND(TI)))
60  A= INT(255*RND(TI))
70  B= INT(255*RND(TI))
80  C= INT(255*RND(TI))
90  D= INT(255*RND(TI))
100 E= INT(255*RND(TI))
110 POKE(7436+F),A: POKE(7437+F),B
120 POKE 7444,C
130 POKE 7445,D: POKE 7446,E
140 PRINT "VOICE IS   ", (F/2+1)
150 PRINT "VOICE IS   ", A,B
160 PRINT "DURATION IS  ",C
170 PRINT "TEMPO IS  ", D,E
180 PRINT: PRINT
190 SYS(7447): GO TO 40
```

This is not a very efficient way to play music, but you get the basic idea of what is required. Also this program need not be in BASIC, but could have been a machine language routine as well.

Good luck with the music programming, and we hope you realize many hours of enjoyment from your PETUNIA.

## APPENDIX
## STANDARD NOTE TABLE

| NOTE | LOW BYTE | HIGH BYTE | Note values in hertz |
|---|---|---|---|
| C2 | 233 | 1 | 65.405 |
| C2# | 6 | 2 | 69.295 |
| D2 | 37 | 2 | 73.415 |
| D2# | 69 | 2 | 77.783 |
| E2 | 104 | 2 | 82.408 |
| F2 | 140 | 2 | 87.308 |
| F2# | 179 | 2 | 92.498 |
| G2 | 220 | 2 | 97.998 |
| G2# | 8 | 3 | 103.83 |
| A2 | 54 | 3 | 110.00 |
| A2# | 103 | 3 | 116.54 |
| B2 | 154 | 3 | 123.47 |
| C3 | 209 | 3 | 130.81 |
| C3# | 11 | 4 | 138.59 |
| D3 | 73 | 4 | 146.83 |
| D3# | 138 | 4 | 155.57 |
| E3 | 207 | 4 | 164.82 |
| F3 | 25 | 5 | 174.62 |
| F3# | 102 | 5 | 185.00 |
| G3 | 184 | 5 | 196.00 |
| G3# | 15 | 6 | 207.65 |
| A3 | 108 | 6 | 220.00 |
| A3# | 205 | 6 | 233.08 |
| B3 | 53 | 7 | 246.94 |
| C4 | 163 | 7 | 261.62 |
| C4# | 23 | 8 | 277.18 |
| D4 | 146 | 8 | 293.66 |
| D4# | 21 | 9 | 311.13 |
| E4 | 159 | 9 | 329.63 |
| F4 | 49 | 10 | 349.23 |
| F4# | 204 | 10 | 369.99 |
| G4 | 113 | 11 | 391.99 |
| G4# | 31 | 12 | 415.30 |
| A4 | 215 | 12 | 440.00 |
| A4# | 155 | 13 | 466.16 |
| B4 | 106 | 14 | 493.88 |
| C5 | 69 | 15 | 523.24 |
| C5# | 46 | 16 | 554.36 |
| D5 | 36 | 17 | 587.32 |
| D5# | 41 | 18 | 622.26 |
| E5 | 62 | 19 | 659.26 |
| F5 | 98 | 20 | 698.46 |
| F5# | 153 | 21 | 739.98 |
| G5 | 226 | 22 | 783.98 |
| G5# | 62 | 24 | 830.60 |
| A5 | 175 | 25 | 880.00 |
| A5# | 54 | 27 | 932.32 |
| B5 | 212 | 28 | 987.76 |
| C6 | 139 | 30 | 1046.5 |
| SILENCE | 00 | 00 | -(dash) |

## APPENDIX 2

## SOFTWARE LISTING FOR PETUNIA

This program is based on software by Hal Chaberlin which originally appeared in the September 1977 issue of BYTE magazine "A Sampling of Techniques for Computer Performance of Music". We urge you to read this article for a more complete description of how this program works.

The program has of course been modified for use on the PET. A waveform table exists at 1C00 HEX and is 256 bytes long. It ends at 1CFF HEX and contains a complex organ sounding note. You may change the values in this table if you wish, but keep the values below 63 decimal so the sum of the four voices do not overlap. The waveform table is not listed.

```
ADDR    CODE            ASSEMBLY LISTING


                        W1TB=$1C00
                        W2TB=$1C00
                        W3TB=$1C00
                        W4TB=$1C00

                       *ZERO PAGE VARIABLES

0035                    BASE=$0035
0035                    V1PT=BASE
0038                    V2PT=BASE +3
003B                    V3PT=BASE +6
003E                    V4PT=BASE +9
0041                    V1IN=BASE +12
0043                    V2IN=BASE +14
0045                    V3IN=BASE +16
0047                    V4IN=BASE +18
0049                    DUR=BASE  +20
004A                    TEMPO=BASE +21

                        Four Voice Subroutine

1D00    00              TV1PT   BYT 0
1D01    00                      WORD W1TB
1D02    1C                      $1C
1D03    00              TV2PT   BYT 0
1D04    00                      WORD W2TB
1D05    1C                      $1C
1D06    00              TV3PT   BYT 0
1D07    1C                      WORD W3TB
```

```
1D09    00           TV4PT   BYT 0
1D0A    00 1C                WORD W4TB
1D0C    00 00        TV1IN   WORD 0          VOICE 1 Value
1D0E    00 00        TV2IN   WORD 0          VOICE 2 Value
1D10    00 00        TV3IN   WORD 0          VOICE 3 Value
1D12    00 00        TV4IN   WORD 0          VOICE 4 Value
1D14    00           TDUR    BYT 0          Note duration byte
1D15    52 00        TTEMPO  WORD 82         Tempo word
                     DAC=$E841
                     DDR=$E843
1D17    A2 15        PATCH   LDX 21          SYS HERE TO PLAY NOTE
1D19    8D 00 1D     PAT1    LDA TV1PT,X
1D1C    95 35                STA V1PT,X
1D1E    CA                   DEX
1D1F    D0 F8                BNE PAT1
1D21    78                   SEI
1D22    A0 00        PLAY    LDY 0
1D24    A6 4A                LDX TEMPO
1D26    18           PLAY1   CLC
1D27    B1 36                LDA (V1PT+1),Y
1D29    71 39                ADC (V2PT+1),Y
1D2B    71 3C                ADC (V3PT+1),Y
1D2D    71 3F                ADC (V4PT+1),Y
1D2F    8D 41 E8             STA DAC
1D32    A5 35                LDA V1PT
1D34    65 41                ADC V1IN
1D36    85 35                STA VIPT
1D38    A5 36                LDA V1PT+1
1D3A    65 42                ADC V1IN+1
1D3C    85 36                STA V1PT+1
1D3E    A5 38                LDA V2PT
1D40    65 43                ADC V2IN
1D42    85 38                STA V2PT
1D44    A5 39                LDA V2PT+1
1D46    65 44                ADC V2IN+1
1D48    85 39                STA V2PT+1
1D4A    A5 3B                LDA V3PT
1D4C    65 45                ADC V3IN
1D4E    85 3B                STA V3PT
1D50    A5 3C                LDA V3PT+1
1D52    65 46                ADC V3IN+1
1D54    85 3C                STA V3PT+1
1D56    A5 3E                LDA V4PT
1D58    65 47                ADC V4IN
1D5A    85 3E                STA V4PT
1D5C    A5 3F                LDA V4PT+1
ID5E    65 48                ADC V4IN+1
1D60    85 3F                STA V4PT+1
1D62    CA                   DEX
1D63    D0 08                BNE TIMW
1D65    C6 49                DEC DUR
1D67    F0 0C                BEQ ENDN
1D69    A6 4A                LDX TEMPO
1D6B    D0 B9        TIMW    BNE PLAY1
```

```
1D6D    D0 00                   BNE 1D6D+2
1D6F    D0 00                   BNE 1D6F+2
1D71    D0 00                   BNE 1D71+2
1D73    D0 B1                   BNE PLAY1
1D75    58              ENDN    CLI             BACK TO BASIC
1D76    60                      RTS
```

## APPENDIX 3

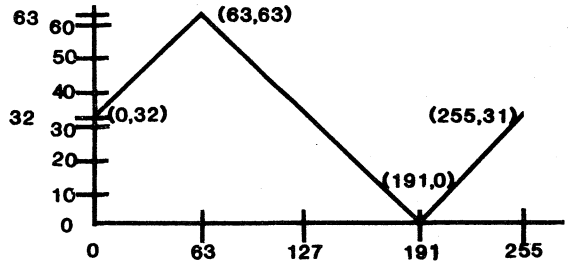
This is a BASIC listing of the software. Data lines 120 through 270 comprise the note table in decimal. The editing features of PET allow you to easily change their values to make any waveform you wish. The main constraint is: do not exceed the value of 63 decimal so the sum of four voices will not overflow.

The data lines 290 - 370 correspond to the assembly listing lines 1D00 through 1D76 HEX.

```
                    **PETUNIA**
20      POKE 135,28
30      FOR A=7168 TO 7542
40      READ B
50      POKE A,B
60      NEXT A
70      PRINT "[redacted]PETUNIA SUBROUTINE IS NOW LOADED"
80      PRINT "AND PROTECTED"
90      PRINT "[redacted]WHEN I SAY 'READY' TYPE 'NEW'"
100     PRINT "TO CLEAR THIS PROGRAM AND THEN ENTER"
110     PRINT     "YOUR MUSIC PLAYING PROGRAM."
120     DATA    51,52,53,54,54,55,56,57,57,58,58
130     DATA    59,59,59,60,60,60,60,60,60,60,60,60,60,60,60
140     DATA    59,59,59,59,59,58,58,58,58,58,58
150     DATA    57,57,57,57,57,57,57,57,57,57,58,58,58,58,58
160     DATA    57,57,57,57,60,60,60,61,61,61,61,62,62,62,62
170     DATA    63,63,63,63,63,63,63,63,63,63,63,63,62,62,62
180     DATA    61,61,60,60,59,59,58,57,56,56,55,54,53,52,51,50
190     DATA    49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34
200     DATA    33,32,31,31,30,30,29,29,29,29,28,28,28,28
210     DATA    29,29,29,29,29,30,30,31,31,32,32,33,33,35,35,36,36
220     DATA    37,38,38,39,40,40,41,41,41,42,42,43,43,43,43,43,43,43,43
230     DATA    42,42,42,41,41,40,39,39,38,37,36,35,34,33,32,31
240     DATA    29,28,27,25,24,23,21,20,19,17,16,15,13,12,11,9,8,7,6,5,4
250     DATA    3,3,2,1,1,0,0,0,0,0,0,0,0,0,1,1,1,2,3,4,5,6,7,8,9
260     DATA    11,12,13,15,16,18,19,21,22,24,26,27,29,31,32,34
270     DATA    35,37,39,40,42,43,44,46,47,48,49
290     DATA     0,0,28,0,0,28,0,0,28,0,0,28,0,0,0,0,0,0,0,0,0
300     DATA    82,0,162,21,189,0,29,149,53,202,208,249
310     DATA      120,160,0,166,74,24,177,54,113,57,113,60
320     DATA      113,63,141,65,232,165,53,101,65,133,53,165
330     DATA      54,101,66,133,54,165,56,101,67,133,56,165,57
340     DATA      101,68,133,57,165,59,101,69,133,59,165,60,101
350     DATA      70,133,60,165,62,101,71,133,62,165,63,101,72
360     DATA      133,63,202,208,,8,198,73,240,12,166,74
370     DATA      208,185,208,0,208,0,208,0,208,177,88,96
```

To generate a special waveform of particular interest to you, proceed as follows -

1). Draw your waveform on graph paper, scaled as in the example.



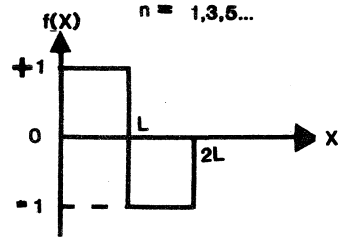

(X,Y) denotes x spaces to right of origin
y spaces above the origin

Now, for X=0 to 255, determine the value of INT (Y). You should now have 256 sets of coordinates. The Y coordinates are the DATA to put into lines 120 thru 270 of the BASIC program.

CAUTION: DATA TABLE MUST HAVE EXACTLY 256 BITS OF DATA OR THE PROGRAM WILL BOMB.

Another way to generate DATA would be to solve an equation, such as a FOURIER EXPANSION.

$$f(x) = \frac{4}{\pi} \sum_{n=1,3,5...} \frac{1}{n} \text{Sin} \frac{n \pi x}{L}$$




For a square wave, this can be written as:

```
10      ?"♡":POKE 135,28
20      FOR X=1 TO 256
30      Z=(360/256)*(pi/180)*X
40      FOR N=1 TO 19 STEP 2
50      A=(I/N)*SIN(N*Z)
60      F=F+A: NEXT
70      S=4*F/pi
80      T=INT(31.5+24*S)
90      POKE(7167+X),T
100     PRINT X,T: A=0:S=0:T=0:F=0
120     NEXT
130     FOR A=7424 TO 7542
```

```
140     READ B
150     POKE A,B
160     NEXT A

290     DATA
300     DATA
310     DATA
320     DATA
330     DATA    SAME AS GIVEN PREVIOUSLY ON PAGE 8.
340     DATA
350     DATA
360     DATA
370     DATA
```

Lines 20 through 120 from a loop to generate exactly 256 bits of data, as required. Lines 40 thru 60 generate the harmonics to make a good square wave, N being the harmonic number. The larger the N, the longer the run time.

Line 80 assures us that only integers are poked, as required by PET. Line 90 pokes the data. Line 100 gives us a print out of each bit and what its data is. This keeps us occupied, as the program takes time to run.

We may add:

```
18 Q = TI
165 PRINT (TI-Q)
```

and when the program is finished running, a print out shows the number of jiffies required to run.

If we write: 40 FOR N=1 TO 1 STEP 2 we will get a sine wave.
If we write: 40 FOR N=1 TO 9 (STEP 1), we will get a sawtooth and the larger the N goes, the more harmonics are calculated and the more ideal our waveform will be.

## PARTS LIST
## PETUNIA

| | |
|---|---|
| 1 | PC board |
| 2 | 16 pin sockets |
| 2 | CD4050B hex buffers |
| 2 | Stackpole 47k resistor packs (7 en. isolated resistors) |
| 1 | 390K ohm 1/4W 5% resistor |
| 1 | 820K ohm 1/4W 5% resistor |
| 1 | 1.6M ohm 1/4W 5% resistor |
| 1 | Female F connector PC mount |
| 1 | 470uf 35 volt electrolytic vertical mounting |
| 1 | Socket TRW 251-06-30-160 / 50 12A30 |
| 1 | Socket 251-12-30-160 / 50 24B10 |

APPENDIX 3

PETUNIA SCHEMATIC