

## Introduction

When exploring QL Sound I had to consider my lack of any musical ability. My grandfather in his day played the piano, performed on stage with various song and dance routines and was a bit of an entrepreneur running a small troop of entertainers. Regrettably I admit his musical talents were not passed on and I have been told, mostly in polite terms, that I am tone deaf (a misspent youth listening to loud rock maybe). As to his entrepreneurial skills, I like to think I have some be it limited as with my computer programming.

## QBITS QLSound

So as for musical expertise I start with something of a disadvantage, yet still curious as to what might be achieved experimenting with the **QL SuperBASIC BEEP** command. I began with modifications to the **BEEP exerciser** Prog... from **QL SuperBASIC - The Definitive Handbook** by Jan Jones. Changing the Attribute settings I attempted to create various instrumental sounds. For games this might be the thud of a Dart hitting the board or the whack of a Golf ball driven down the Fairway. Then sounds such as the background purr of an engine running, a sudden Alarm or Siren associated with imminent danger or an emergency. Then Guns, Laser weapons firing with the accompanying sound of a deadly explosion.

The Special Edition has a **TRACKER** Mode for the range of instrumentals that the **BEEP** Attributes can achieve with scaling and wraps. The entries arranged in rows, can be swapped to change their sequence and two or more rows can be set to be played in a continuous loop.

The **SCORE** Mode is for composing with Staves set with Symbols for Notes and Rests etc. and includes additional Labels and Ledgers generated by a Notes octave position. Changeable Settings for the Beat and Metronome speed and Playback of a page or all pages.

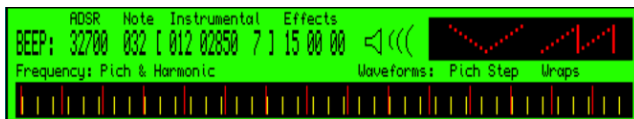
The **QPC2 Emulator** has an implementation of an AY-3-8910 sound generator chip with a number of **QSound** commands. An option to Playback Musical Notes with **QSound** commands has been added as part of the Special Edition.

## QBITS QLSoundSE - BEEP Attributes

In (M)ode **BEEP** the Attributes have description labels to help identify the workings. ADSR for duration of the generated sound, the Attack, Decay, Sustain and Release. Note being the primary Pitch, Harmonic the second Pitch combined with the Time Step and Pitch or angle of Step creates the Timbre, this being the characteristic tonal quality that differentiates between that of say a guitar or piano. Wraps added to the Time step creates scaling and looping effects such as a Wailing or that of a Siren. Fuzz, noisiness and Random add further to shape the waveform.



Graphic display shows how the attributes cause and effects changes to the output Waveform. These display **Pitch** and **Harmonics** frequencies as vertical lines. The first Waveform is Pitch Time/Step created by the **grad\_X**, **grad\_y** parameters. The second Waveform is the assumed effects of the **Wrap** parameter. My hope is in exploring the **BEEP** command Attributes, listening and identifying their effects will lead to a more methodical way of constructing useful sound outputs.



The Pitch and Harmonics vertical lines become wider apart as the frequency of the Pitch lowers. The Pitch Step and Wraps Wave shows Scale and Direction. Play back the various **BEEP** Attributes using 'N' Note, 'H' Harmonic, 'W' Wraps, 'F' Fussiness" Random (Note: UPPER or lower case).

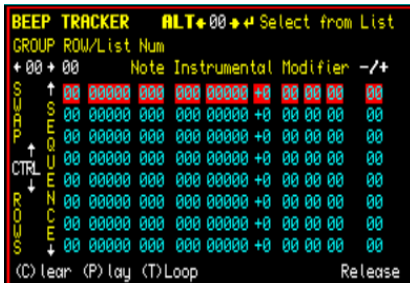
**Note:** Select a BEEP entry from **BEEP TRACKER** and Press Enter. Choose **Attribute** with **Shift Up/Down** cursor Keys and change **values** with **Shift Left/Right** cursor keys. Press Enter to update **Display** and Current Row of **TRACKER** with the changed BEEP Attributes.

## QBITS QLSoundSE - BEEP TRACKER

A Tracker helps arrangement of instrumental sounds varying from the musical to just quirky, such as whispery atmospherics to explosions and wailing sirens. The **BEEP** List holds (1-96) entries. Select with **ALT** **←00→** **Left/Right** Cursors. The **TRACKER** has Groups (0 to 9) with 24 Rows (00 to 23) each entry holding a set of BEEP Attributes. Use **Left/Right** cursors to select Pattern **Group**, and **Up/Down** Cursors to Scroll Row Number, the current Row is shown highlighted.

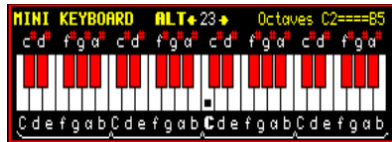
Swap Rows with **CTRL** and **Up/Down** Cursors to re-sequence the entries. Set **PAUSE Release** value **-/+** [00-15] to delay the playing of the following Row. **(C)**ear - Resets selected Row.

Once a number of entries have been set use **(P)**layback for all or **(T)**Loop to select a set number of rows on continuous Playback. To end Playback, press **Spacebar**.

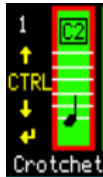


## QBITS QLSoundSE - MINI KEYBOARD

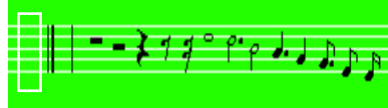
In (M)ode **SCORE** a four Octave Keyboard C2 to B5 is displayed. The Marker ■ shows current Key/Note position, use ALT ← 23 → to change. The bottom Left of screen shows the Notes position on the Stave.



## QBITS QLSoundSE - SCORE Symbols



Use CTRL ↑ ↓ to Select a Symbol (Space through to SemiQuaver). Press Enter and Symbol is written to the next Stave position. When required Ledger lines are added or the Note marked with the relevant Octave Label ie. [C2].



Others variants (1) Staccato with a dot placed above or below the Note head and (2) Tenuto marked with a short line or horizontal bar placed above or below the Note head. These are effects that shorten (blend) or extend (detach) the Time between Notes.

## QBITS QLSoundSE - Beat & Metronome

Press 'B' and use Left /Right Cursors to change displayed Rhythm. This ranges from none (Space) or 2/2 through to 6/8. To change the Metronome Timing use Up/Down Cursor Keys. The range is from 30 to 240 Beats per Minute.



Press '#' Hash to Reset default Pitches.

Set Notes to Default Y/N

Press 'Q' to action with QSound PLAY command.

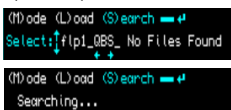
Set Notes with QSound Y/N

Press 'F' displays an FClef in bottom Stave Row for where it might be used. [Remove FClef by swapping to BEEP Mode and back to SCORE Mode]

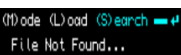
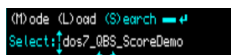


**Note:** QBITS QLSoundSE Prog includes an implementation of the QSound AY-3 system Play command and associated attributes to produce the preset Notes for the Four Octave Keyboard.

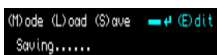
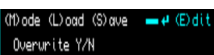
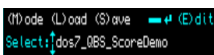
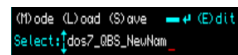
## QBITS QLSoundSE- File Management



For Load change Device with Up/Down Cursor Keys and Press 'S' to Search device for QBS\_ Data files. Either 'No Files Found' is displayed or use Left/Right Cursors to select a File. Action with Enter or abort with Spacebar. If a genuine QBS File, Loading will commence.



For Save the Loaded or Default Filename is displayed. This can be edited by Pressing 'E', Delete with left Arrow, add new Character(s), Return with Enter.

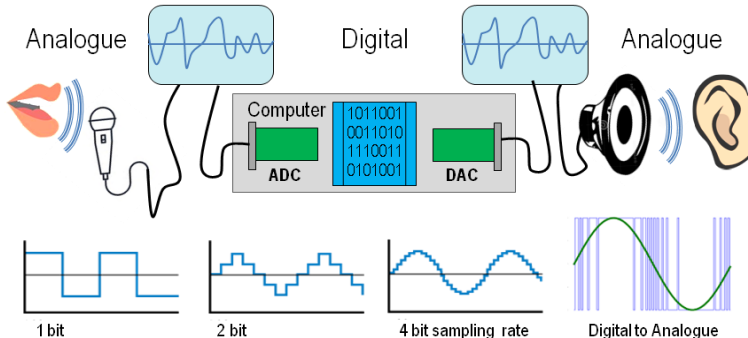


Change Device name? Action with Enter or Abort with Spacebar.

**Note:** More details on background and coding of QBITS QLSoundSE to follow...

## Exploring Sounds - Digital Audio

The human ear registers vibrations, sounds heard that are analogue in nature. In sound recording and reproduction systems, digital audio is the encoded representation of these sounds for processing, storage or transmission. The analogue wave length is sampled in regular time slots and the varying amplitude represented as a series of precise numbers. This allows editing and mixing to be introduced with special effects to simulate reverberation, enhancement of certain frequencies or change of pitch.

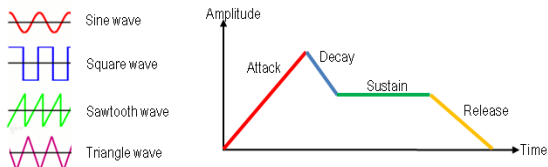


The minimum sample rate of any signal without introducing errors is twice the highest frequency present in the signal and is called the Nyquist frequency or Limit. A Digital representation can be expressed as a number of on/off's, highs and lows or in binary logic of 1's and 0's.

## The Sound Synthesizer

Electronic synthesizers use basic components that work together to reproduce a sound. Oscillators that generate the waveform and change of pitch, filtering that removes certain frequencies in the wave to change the timbre, an amplifier that controls the signal volume, and varying modulation to create effects.

A pure note or pitch will be in the form of a sine wave. Mixed with sympathetic vibrations enriches the tonal blend of a pitch creating differing waveforms. Timbre is perceived as the quality of these differing sounds, the characteristic that represents a pre-conscious identity, based on information gained from frequency transients, noisiness, unsteadiness, perceived pitch and spread of harmonics in a specific time frame. Wow! Really! Simplistically this means they make distinguishable the same pitch from being played on a piano as opposed to a violin. The main pitch is called the fundamental frequency and the related frequencies the harmonics. The wave envelope is the relationship of amplitude to time, this is called the **ADSR** pattern, Attack, Decay, Sustain and Release.



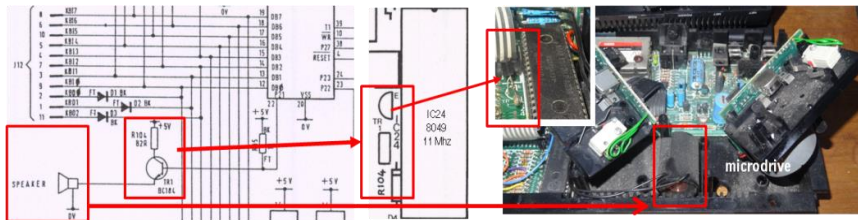
## Summarising

A synthesizer needs to generate sound waves of different shapes. Supply more than one sound tone to produce a fundamental frequency and its harmonics. Then make the volume of the sound change over time to produce different **ADSR** envelope shapes.

## QL Sound Generation

For Sound the QL 68008 CPU (Central-Processor-Unit) communicates with a slave processor 8049 called the IPC (Intelligent-Peripheral-Controller). The **Intel 8049** co-processor chip used by the **Sinclair QL home computer** is designed to work alongside the custom chip ZX8302 ULA (Uncommitted Logic Array), acting as a keyboard buffer / joystick buffer, RS232 receive buffer and as a sound generator directing the output to the internal loudspeaker. The 8049 Chip contains a 2kx8 program memory, a 128x8 RAM, 27 I/O lines, an 8-bit timer/counter in addition to on board oscillator and clock circuits.

Some QLs have an 8749 chip, which is the EPROM version of the Intel 8049. The 8049 can also be replaced by the Hermes Co-Processor manufactured by Tony Firshman, which provides improvements to sound, serial ports and further eliminates problems of keyboard bounce.

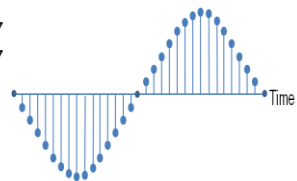


In executing a **SuperBASIC BEEP** instruction, the IPC port 2 P21 switches on/off transistor TR1 with the changing number of 1s and 0s in each output cycle. The emitter follower configuration is used as a voltage buffer amplifier to drive the 600hm 23mm loudspeaker situated between the two microdrives. Due to device inherent latency, it is assumed this produces an output perhaps more recognisable as an analogue wave. The values to generate this digital to analogue conversion are derived from the BEEP parameters sent as part of the IPC instruction.

## QL BEEP - IPC Communication

A command sent to the IPC uses the Manager Trap **MT.IPCOM** in the form of a header describing the command followed by any parameters. For audio output this is: Initiate Sound, 8 parameters.

8 bits	pitch_1	range 0 to 255
8 bits	pitch_2	range 0 to 255
16 bits	interval between steps	range -32768 to +32767
16 bits	duration	range -32768 to +32767
4 bits	step in pitch	range -8 to 7
4 bits	wrap	range 0 to 15
4 bits	randomness of step	range 0 to 15
4bits	fuzziness	range 0 to 15

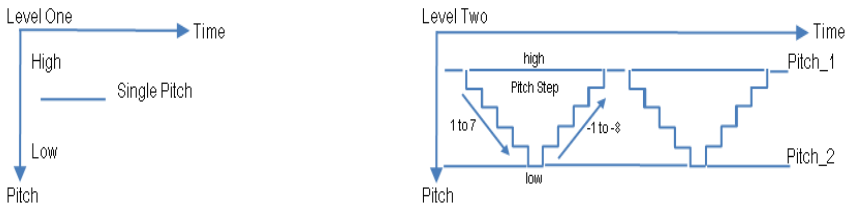


**Frequency and Amplitude:** Two key features of a sound wave are the frequency (how many times the wave vibrates in one second) and is broadly related to the Pitch of the sound we hear. The amplitude (Volume) of a sound is related to the amount of energy that the sound wave carries. As the frequency increase the shorter the wave length, this is represented by the number of changes within a time frame. The higher the energy, the louder it sounds, the higher a waves amplitude. A wave form is therefore generated by the number of ones in the binary record processed by the DAC on each cycle and related to the sample rate. For the QL System the amplitude is Digitally represented in binary 1s and 0s so higher pitches will be heard louder.

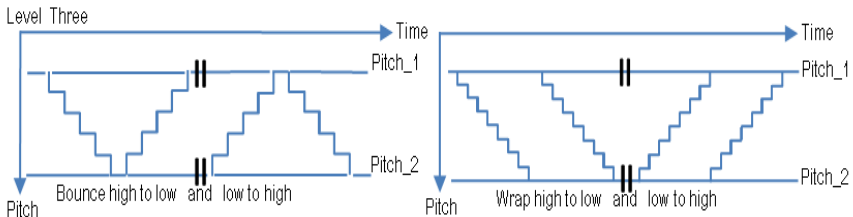
## QL Sound Concepts

The QL Guide describes Sound being generated by the IPC (8049) as controlled by specifying a number of parameters, allowing the stage-by-stage build-up of more complex sounds.

**The first level** is a single Pitch active for a specified time, which is a pure sound at a set frequency. Once the **IPC 8049** has been instructed it will itself carry on for the specified duration. The **BEEP** command with a duration value of zero will run until a following **BEEP** command cancels it out or changes the parameters for a different sound. The **BBQL** duration is allegedly carried out in units of 72 microseconds the range being 1 to 32767 or again from -32768 to -1 (the duration for -1 or 32767 being 2.36 seconds).

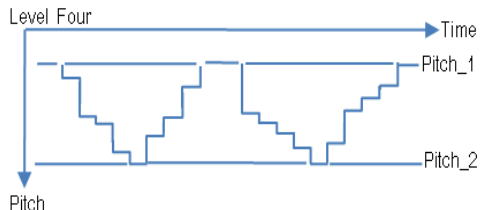


**At level two** a second pitch is added and the rate at which the sound ramps between the two pitches allegedly can produce semi musical beeps, spiralling or rippling tones, growls, zaps and moans. The number of steps and direction high to low or low to high can be configured.



**The third level** controls how the sound behaves after reaching one of the pitches. The sound is left to bounce or wrap a number of times. Depending on what step direction this can be high to low or visa verse.

**Level four** introduces a deviation from the specified step or gradient in moving between pitches with random larger or smaller steps. This random element can generate a wide and unexpected range of sounds.



Fuzzy is level five and is described as a further variation that adds changes to the pitch being generated and tends to make the sound more like a buzz.

**Note:** Directly being able to vary the amplitude of the sound output is unfortunately not an option.

## QL SuperBASIC BEEP Attributes

The eight parameters listed in the QL User Guide are **duration**, **pitch**, **pitch\_2**, **grad\_x**, **grad\_y**, **wraps**, **fuzzy** and **random**. They are grouped as duration + pitch, pitch\_2 + grad\_x + grade\_y, wrap, fuzzy, random and used in different combinations and values to build complex sounds.

### Duration

If the minimum time length used by the **IPC (8049)** processor is 1x72 microseconds, what would be the shortest multiple to perceive a sound? One factor is the Pitch or Frequency and the other the volume of Amplitude. Once a sound wave reaches the human ear the brain can perceive it in around 50 milliseconds. The stapes reflex is where the ear protects itself from very loud noises, here perception can be in as little as 25 milliseconds. In the relationship between hearing a sound and pitch perception this would be in the order of 100ms or slightly less for a higher pitch.

When a program has a number of **BEEP** Instructions to be carried out sequentially there is a concern that the sounds will be overwritten before being fully executed. The length of a sound being an important factor, an alternative to Duration parameter is to use the Keyword **PAUSE** to control when to activate a following **BEEP** instruction to the **IPC 8049** processor. The **PAUSE** command uses multiples of 20 milliseconds for example: - **BEEP 0,3 : PAUSE 100 : BEEP** (will last two seconds).

### Pitch

Hearing the sensation of a vibration is commonly referred to as Pitch, which is the perceptual property of sounds and allows the ordering of their frequency to be judged as higher or lower. When only the **BEEP** duration and first pitch parameters are used it produces a single fundamental frequency. The **QL BEEP Pitch** range is 0 to 255 where the pitch climbs from its lowest at 255.

### Harmonic

The second Pitch (**Pitch\_2**), I will refer to as the **Harmonic**, add a **Time Interval** (**grad\_x**) and a **Pitch Step** (**grad\_y**), they create a sequence of sound variations ordered by the time duration and number of steps between the main pitch and second pitch. The Time Interval again is in multiple units of 72 microseconds for each note in the sequence. The Pitch Step range is -8 to 7 where step 1 to 7 scales downwards high to low pitch and -8 to 0 starts the sequence scaling upwards from low to a higher pitch. From then on the sequence bounces between the two pitches. A **Harmonic** without a **Time Interval** and/or **Pitch Step** has no affect. Adding a **Pitch** step of 1 when **Harmonic** and **Time Interval** are both 0 identifies the pitch as a high zero. **Harmonic** plus a **Pitch** step with **Time Interval** 0 just changes Main **Pitch** to the **Harmonic**.

### Wraps

Wraps repeat a sequence of harmonics produced by the **pitch\_1**, **pitch\_2**, **grad\_x**, **grad\_y** parameters a number of times. Zero continues the bounce effect of the harmonic. Increasing values 1 to 7 creates scaling high too low for the number of Wraps. Scaling 8 to 15 creates Wraps low to high.

### Fuzzy & Random

Fuzzy decreases the purity of the pitch, Random just randomises the steps until little of the original sequence is evident. Both of these have a range 0 to 15, zero has no effect and the active range is more like 8 to 15. Increasing the fuzzy range as said before, blurs the pitch to a buzz.

### BEEPING

This **SuperBASIC** function detects if the QL hardware is producing a sound and simply returns as true or false. **IF BEEPING THEN BEEP**. If true this will cancel any QL Sound output.

## QL Sound Output

As a starting point in coding for a QL Sound output, I added a few modifications to the **BEEP Exerciser** Prog... from the **QL SuperBASIC - The Definitive Handbook** by Jan Jones.

Variable	Parameter	Value	Description
d	duration	0 to 235	[ 0 increments of $d*1e4/72$ ]
p	pitch	0 to 255	[ 0 highest descending to 255 ]
h	harmonic	0 to 255	[ 0 highest descending to 255 ]
t	time interval	0 to 235	[ 0 increments of $t*1e4/72$ ]
s	pitch step	0 to 15	[ effective range 1 to 7 low to high 8 to 15 high to low ]
w	wrap	0 to 15	[ effective range 1 to 15 ]
f	fuzz	0 to 15	[ effective range 8 to 15 ]
r	random	0 to 15	[ effective range 8 to 15 ]

100 REMark **QLBeepv1** (QBITS Exploring QL Sounds 2018)

104 MODE 4:**Blnit** : **BMenu**

108 **DEFine PROCEDURE Blnit**

110 WINDOW 492,200,8,8:PAPER 7:INK 0:CSIZE 0,0:CLS

112 CURSOR 8,6:PRINT 'Play (P)itch +(H)armonic +(W)rap +(F)uzz +(R)andom'

114 CURSOR 8, 50:PRINT 'Duration : (0 to 235)' :REMark d

116 CURSOR 8, 60:PRINT 'Pitch : (0 to 255)' :REMark p

118 CURSOR 8, 70:PRINT 'Harmonic : (0 to 255)' :REMark h


120 CURSOR 8, 80:PRINT 'Time Step : (0 to 235)' :REMark t

122 CURSOR 8, 90:PRINT 'Pitch Step : (-8 to 7)' :REMark s

124 CURSOR 8,100:PRINT 'Wraps : (0 to 15)' :REMark w

126 CURSOR 8,110:PRINT 'Fuzz : (0 to 15)' :REMark f

128 CURSOR 8,120:PRINT 'Random : (0 to 15)' :REMark r

130 CURSOR 8,134:PRINT 'Edit use  Space cancels BEEP <Esc> quit menu'

132 DIM BPm(7):INK 2:FOR ipm=0 TO 7:BRead

134 **END DEFine**

138 **DEFine PROCEDURE BMenu**

140 INK 0:ipm=0:BPrt

142 **REPeat lp**

144 CSIZE 0,1:CURSOR 8,24:PRINT 'BEEP: ';d; 'p; [ 'h;' 't;' 's;' ] 'w;' 'f;' 'r;':CLS 4

146 k=CODE(INKEY\$(-1)) :REMark Read Keyboard

148 **SElect ON k**

150 =208:IF ipm>0:BChange -1 :REMark Up

152 =216:IF ipm<7:BChange 1 :REMark Down

154 =192:BPm(ipm)=BPm(ipm)-1:BRead :REMark Left

156 =200:BPm(ipm)=BPm(ipm)+1:BRead :REMark Right

158 = 80,112:BEEP d,p :REMark (P)itch

160 = 72,104:BEEP d,p,h,t,s :REMark +(H)armonic

162 = 87,119:BEEP d,p,h,t,s,w :REMark +(W)rap

164 = 70,102:BEEP d,p,h,t,s,w,f :REMark +(F)uzz

166 = 82,114:BEEP d,p,h,t,s,w,f,r :REMark +(R)andom

168 = 32:BEEP

170 = 27:BEEP:STOP

172 **END SElect**

174 **END REPeat lp**

176 **END DEFine**



```

180 DEFine PROCedure BPrt
182 CSIZE 0,0:CURSOR 80,50+ipm*10:PRINT BPm(ipm) TO 14:CSIZE 0,1
184 END DEFine

```

```

188 DEFine PROCedure BChange(change)
190 INK 4:BPrt
192 ipm=ipm+change
195 INK 7:BPrt
196 END DEFine

```

```

200 DEFine PROCedure BRead
202 BPrt: d=INT(BPm(0)*10000/72): t=INT(BPm(3)*10000/72)
204 p=BPm(1):h=BPm(2): s=BPm(4):w=BPm(5):f=BPm(6):r=BPm(7)
206 END DEFine

```

Play <P>itch +<H>armonic +<W>rap +<F>uzz +<R>andom

BEEP: 0 1[ 3 10000 7 ]0 0 0

Duration	:0	(0 to 235)
Pitch	:1	(0 TO 255)
Harmonic	:3	(0 TO 255)
Time Step	:72	(0 to 235)
Pitch Step	:7	(-8 to 7)
Warps	:0	(0 to 15)
Fuzz	:0	(0 to 15)
Random	:0	(0 to 15)

Edit use ↑↓→ Space cancels BEEP <Esc> to quit menu

To my untrained ear the example gives a passable representation of a UK Police Panda car siren.

### QBITS QL BEEP Exerciser

Using **BEEP** parameters with the QL internal speaker arrangement is more a trial-and-error process rather than any constructed methodology. However, the program supplied here allows setting the various Attributes and switching them on sequentially to hear the effects that take place.

How does the QL Sound Generator fit the bill? It has two pitches and a method to ramp up and down between the two frequencies producing a range of harmonics. Adding harmonics can build the fundamental frequency from sinusoidal to more that of a square wave. Then there's Wrap which I believe is intended to create an output similar to a sawtooth wave. There are also the parameters for fuzziness and randomness potentially further changing the output waveform. The one thing the IPC doesn't have is any separate control over wave amplitude. This gives a partial explanation as to why higher pitches sound louder than their lower counterparts, the strings of 1s and 0s being closer together are going to add to create more sound.

The next step was to look more deeply into Pitch Frequency and their Harmonics. This relationship between short and long-time intervals of rising or falling pitches and the wave envelope they produce create tonal quality. In music this can be represented by symbols that identify a range of Notes and the way each is to be played. A Notes differing tone are dependent on the instrument being played. Therefore, at this point we take a quick overview of musical representation, Stave, Clefs, Notes, their meaning and relationship.

## Identifying a Music Note

In evaluating Pitch in musical terms let's begin by identifying the Notes and their representation, letters or symbols are used making it easier to write and quicker to read. Pitch classes are represented by letters of the alphabet (A,B,C,D,E,F,G) or more often by the naming convention Do-Re-Me-Fa-Sol-La-Ti.

The letter definition and corresponding notes are:

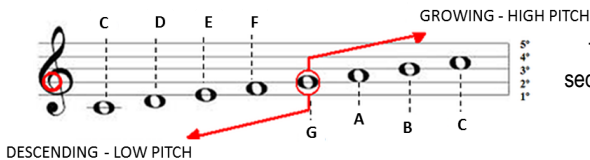
C → do  
D → re  
E → mi  
F → fa  
G → so  
Separator  
A → la  
B → ti (H in German)



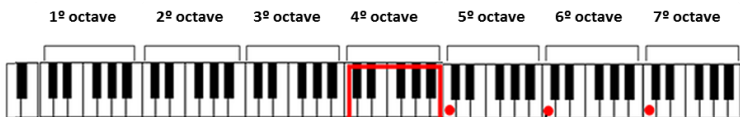
Sheet music registers the harmonic, rhythmic and melodic ideas. The Notes are positioned and written in the form of musical symbols.

## Music Stave

The five lines (1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup> and 5<sup>th</sup>) of a **Stave** are where each line and each space between represent a different note of scale.



To read sheet music - is the sequence of the notes, forwards and backwards!



Middle **C (C4)** located at the centre of a piano keyboard. The highlighted 'C' notes hold different stave positions dependant on the octave in which they are located.

CENTRAL C ( C4 )



## Ledger Lines

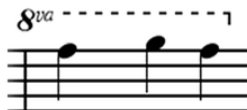
Where the Stave can't handle the representation of the notes for a full range of octaves, ledger lines are used. These lines are nothing more than the continuation of the Stave, they are used to represent notes that surpass the bottom or upper limits.

## Treble Clef

Musicians throughout history have assigned different positions for their notes. **Clefs** were created as symbols serving to sign the note and the line of reference adopted. The most common Clef for guitar, piano and voice is the **Treble Clef** also known as the **G Clef** because the design of the Clef encircles the second Stave line which is G.



The Symbol 8v is followed by the letter "b", which means "below", "8va" would be for octaves above.



Interpretation of the notes (F, G, F) should be played one octave above the position that it is the Stave.

## Accidentals

To show the increase or decrease of a notes pitch by one half step, symbols called Accidentals are used. When these same symbols appear at the very beginning of the music Score they specify a key signature. They stay in effect for all of the notes of the same pitch for the rest of the measure.

Flats lower the pitch of the note by one half step.



Sharps raise the pitch of the note by one half step.



Naturals cancel out any previous Flats or Sharps. The pitch returns to normal.

## Articulations

These effect how the note is played and include the slur, ties, phrase mark, staccato, staccatissimo, accent, sforzando, rinforzando, and legato.

Slurs smoothly connect notes of different pitches. This means to play the notes without breaks.

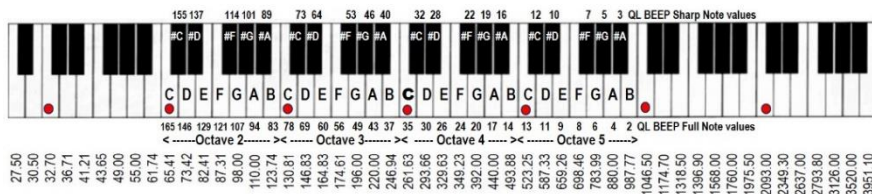


Ties connect notes of the same pitch, forming essentially one longer note.

## Key Notes & BEEP values

At this point I thought it might be useful to review the range of piano keys and associated notes or pitches to their related frequencies. Then with a little help from a **QLUB** Prog and again with my untrained ear I cross referenced **BEEP** Pitch values around the middle **C** as shown on the chart.

BEEP Pitch\_1 = 0 to my untrained ear and equates to a C6 or 1046.50Hz.



## QLUB Music Micro Please!

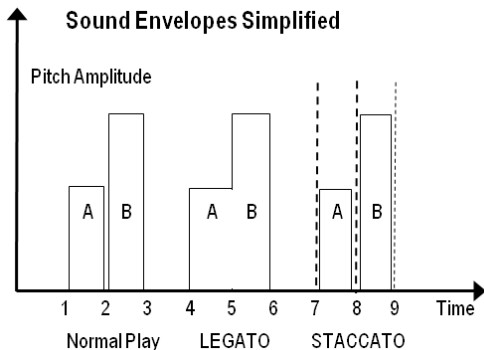
The **QLUB** edition of Mar/April 1985 carried an article with a short program which displayed to screen a **Stave**, a **G-Clef** and added **crotchet** symbols to selected pitches. It described musical notes over two Octaves that could be reproduced and gave the **BEEP pitch\_1** numbered equivalent.

The article displayed the music symbols and their beat values from **Breve** to **hemi-demi-semi-quaver** (8 beats down to 1/16). Also drawn were Simplified **Sound envelopes** showing **Pitch / Amplitude** of Normal Playing time against **Legato** and **Staccato**.

100 REMark **QLUBMicrov1** (QLUB Music Micro QBITS - 2018)

```
104 DIM pitch(18)
106 MODE 4:WINDOW 448,200,32,16:PAPER 4:CLS
108 WINDOW#0,448,20,32,216:PAPER#0,7:INK#0,0:CLS#0
110 PRINT#0,'auto or manual ? (a/m)'
112 IF INKEY$(-1)='m' THEN yourself=1:ELSE yourself=0
```

```
116 REPEAT loop
118 up=50:across=16:inc=0:CLS
120 STAVE:Draw_Clef
122 FOR note=1 TO 18
124   Pick_Note yourself
126   Display_Note
128   pitch(note)=p
130   across=across+8
132 END FOR note
134 Play_Tune
136 CLS#0:PRINT#0,'Another tune ? (y/n)'
138 again$=INKEY$(-1)
140 IF again$='n' THEN EXIT loop
142 END REPEAT loop
144 STOP
```



```
148 DEFine PROCEDURE Pick_Note(yourself)
150 IF yourself
152 REPEAT check
154   CLS#0:INPUT#0, 'Note number (1 to 9)';choice
156 ELSE
158   choice=RND(1 TO 9)
160 END IF
162 SELECT ON choice
164   =1:p=24:inc=0
166   =2:p=22:inc=1.5
168   =3:p=19:inc=3
170   =4:p=15:inc=4.5
172   =5:p=12:inc=6
174   =6:p=11:inc=7.5
176   =7:p= 9:inc=9
178   =8:p= 7:inc=10.5
180   =9:p= 6:inc=12
182   =REMAINDER :END REPEAT check
184 END SELECT
186 END DEFine
```



```

190 DEfIne PROCedure Display_Note
192 FILL 1:CIRCLE across, up+inc,1.5:FILL 0
194 IF p<12
196   LINE across-1.5,up+inc TO across-1.5,up+inc-8
198 ELSE
200   LINE across+1.5,up+inc TO across+1.5,up+inc+8
202 END IF
204 BEEP-1,p:PAUSE#0:BEEP
206 END DEfIne

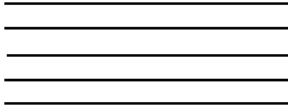
```



```

210 DEfIne PROCedure Stave
212 INK 7
214 FOR ledger=0 TO 12 STEP 3
216   LINE 2,up+ledger TO 165,up+ledger
218 END FOR ledger
220 INK 0
222 END DEfIne

```



```

226 DEfIne PROCedure Draw_Clef
228 LINE 8,up+1.5
230 ARC_R TO 0,4.5,-PI
232 ARC_R TO 0,-6,-PI TO -3,7,-3*PI/4
234 LINE_R TO 5,7:ARC_R TO -2,0,PI
236 LINE_R TO 0,-18
238 FILL 1:CIRCLE_R -1,0,1:FILL 0
240 END DEfIne

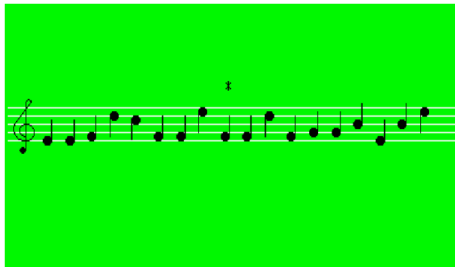
```



```

244 DEfIne PROCedure Play_Tune
246 CLS#0:PRINT#0,'Press any key to Play!'
248 PAUSE
250 x=116:y=72
252 FOR note=1 TO 18
254   Blip x,y
256   BEEP -1,pitch(note)
258   PAUSE 20
260   Blip x,y:x=note*8+16
262   BEEP
264 END FOR note
266 CLS#0:PRINT#0,'Play again(y/n)'
268 IF INKEY$(-1)!='y':Play_Tune
270 END DEfIne

```



```

274 DEfIne PROCedure Blip(x,y)
276 INK 4:OVER -1:CURSOR x,y,0,0 :PRINT '*' :OVER 0
278 END DEfIne

```

The author of the **QLUB** publication was not given, however later that year 1985; the **QL User** magazine published **James Lucy's QL COMPOSER** which would appear to have some connection.

Bearing in mind the quizzical nature of the QL Sound generator it seemed logical to explore and retain sets of BEEP parameters, each key defined either as a musical note or everyday sound or even weird futuristic effects. To create a Score with keyboard notes offering differing sounds it now required a selection of Symbols to identify timing and how they might be played.

## QBITS Music Score

Having the **Staff** and **GClef**, next was determining a Time Signature or **Tempo**. Normal music has a regular pulse or rhythm identified as the **beat**. This is represented by two numbers written after the **Clef** at the beginning of a score to establish the number of beats in each uniformed section or measure. Provided are double **2/2**, **2/4**, **4/4**, and triple **3/4**, **3/8**, **6/8** timing options that can be used with the separator bar. The number on top tells you the beats in a measure; the number at the bottom is an indicator to the note combination for each **beat**. For example, a **4/4** timing would be four beats to the metre and each beat represented by a Crotchet, but potentially other note combinations, two Minims or a single Semibreve.

## QBITS Metronome

The **Beat** timing is calculated against the rate or number to be played per minute. The given range is from 30 to 240, in 10 beat steps. This is then used to calculate an individual **Note** or **Rest** value in determining the duration used with the **PAUSE** Command (*see below*).

For practical use with the **QL Sound System** and to provide a distinct separation between **Notes** or **Rests** from any previously played, a **delay PAUSE** is inserted. For normal playback **80 milliseconds** is chosen as a **standard break**. For **Staccato** (separated) this delay is increase to **120 milliseconds** to make the Note more distinctive and for a **Legato** (lengthened) reduced to **40 milliseconds** so it appears to merge with any following Note. As stated earlier to recognise differing pitches the Human ear requires around 100 milliseconds. Therefore, the shortest duration for normal play based on above assumptions for a Semiquaver (a quarter Note) would need to be in the order of 180ms.

Calculating timing for beats per minute (**bpm**), 60 seconds is multiplied by the **PAUSE** value for one second ie.  $60 \times 50 = 3000$ . This is then multiplied by the **Note** or **Rest** value (4 to 0.25). The result is then divided by the Metronome rate. The **PAUSE** duration for a Crotchet with a max of 240bpm would be  $(60 \times 50 \times 1) / 240 = 12.5$ , for a Semiquaver  $(60 \times 50 \times 0.25) / 240 = 3.125$ . Clearly a duration **PAUSE** of 3 is only 60ms and not nearly enough time for the human ear to differentiate a change in pitch.

For the **ADSR** of a **Note's** playback in this proposed arrangement the **Attack**, **Decay**, **Sustain** is covered by the **BEEP** command with its attributes and set by the following **PAUSE** setting. The second **BEEP** followed by a further **PAUSE** delay creates the **Release** before any following Note.

**Accents** or **Articulations** explain how each **Note** is to be performed, **Staccato** with the note short and detached, **Tenuto** holding the Note for its full value blending into the next as in playing **Legato**. Another way to extending the duration and by half as much time again, is by placing an **Augmentation Dot** after the Note. For modes of play such as **Staccato** or **Tenuto/Legato** and the **Augmentation Dot** the **duration** and **delay** can be adjusted to reflect the change by increasing and decreasing their lengths.

**BEEP** *d,p,h,t,s,w,f,r* : **PAUSE** *duration* : **BEEP** : **PAUSE** *delay*

**Staccato** marked by a dot placed above or below the Note head

**Tenuto** marked by a line placed above or below the Note head.

**Dot** placed after the **note** adds half of the value of the **note** to itself.



As important are the **Rests** where there is no **BEEP** command just a **PAUSE** *duration* + *delay*. The **Space**, **Separation Bar** and **End Bar** are given a zero time.

## QBITS Notes and Scores

The **QBITS Notes** chosen range is from the **Semibreve** down to the **Semiquaver** (a value of 4 beats down to 1/4 of a beat), this includes **Notes** with a **Dot Argumentation**. Then **Rests** to hold equivalent time slots where no music is played. **Sharps** that increase a note by half a pitch step and **Staccato** and **Tenuto** to further emphasise the length of how a note is to be played. Other symbols include a **Separation bar** for the measures or metre and an **End bar** to complete a musical score.

The diagram illustrates the QBITS musical symbols. The top section, titled "NOTES", shows a staff with various note values: Semibreve(4), Minim+Dot(3), Minim(2), Crotchet+Dot(1.5), Crotchet(1), Quaver+Dot(0.75), Quaver(0.5), and SemiQuaver(0.25). A legend on the left shows "Note Parts" (Flag 1, Stem, Head, Flag 2) and "GClef" with "Beat" and "Stave" labels. The bottom section, titled "RESTS", shows symbols for Staccato, Normal, Tenuto, Sharp, SemiBreve(4), Minim(2), Crotchet(1), and Quaver(0.5). It also includes a "Space" box, a "Separation Bar", and an "End Bar".

## QBITS Musical Symbol Generation

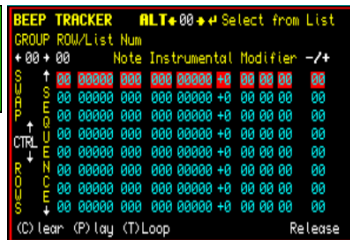
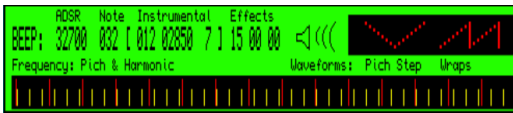
Considerations to take into account are a **Note's** positions either below, between or on a **Stave line**, and if additional **ledgers** are required. The **Stave** and components that make up the Musical Symbols **Notes**, **Rests** etc. use **SuperBASIC** commands that operate with the **Graphics Coordinate System**. Each Symbol of a Score will therefore require progressive positioning along the Stave as well as vertical positioning relative to the scale of a Note being displayed.

The first requirement is the **Space** which clears a position and redraws the **Stave** lines. The **Space** is a **FILLED** rectangle drawn with the **LINE** command. Further use of the **LINE** command then displays the **Stave** lines. For example, the combination of a **Space** and selected **Beat** value after the **GClef** allows the signature **Beat** to be changed or optionally to show **No Beat**.

The **Separation Bar**, **End Bar**, **Semibreve** and **Minim Rests** make further use of the **LINE** and **Fill** commands. The **Crotchet**, **Quaver** and **Semiquaver Rests** required a little more engineering. The actual **Notes** are built up from separate parts, **Head**, **Stem** **Tails** as shown. For **Accidentals** **#Sharps**, **Articulations**, **Staccato/Tenuto Dots - Lines** and following **Augmentation Dots** are added where required.

## QBITS QLSoundSE – Layout

To contain a **BEEP** exerciser, **TRACKER** and **SCORE** Sheet with **KEYBOARD** was not possible in one Screen. The decision was made early in the development stage to separate out the BEEP and SCORE Modes in to different screens.



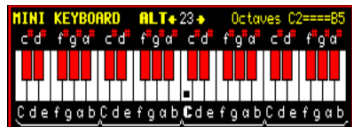
The BEEP Graphics is an attempt to help explain the actions of the Attributes. The Special Edition TRACKER replaces an earlier Keyboard. Its function to provide a way to sequence instrumental compositions for Playback - (P)lay and for selected continuous loops (T)Loop.

## QBITS Micro Keyboard

Reviewing early SuperBasic Magazine Progs aimed at using the **QL BEEP** command I hadn't come across any graphical representation of a keyboard.



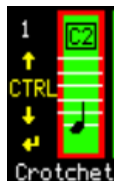
The first QBITS display was two octaves based on QLUB article described earlier. This is now a Four Octave MINI Keyboard.



## QBITS SCORE Sheet



For Beat and Metronome, Press 'B' then use the Cursor keys, followed by Enter.



The Score uses the Keyboard to determine the Notes Pitch, ranging from C2 to B5 this includes the Sharps, [Flats are viewed a step down ie. the Sharp of previous keyboard Note]. In addition, Accents or Articulations are linked by choice of the Note Symbol or actioning (1) Staccato or (2) Tenuto from the bottom Menu. The symbols for NOTES and RESTS etc. shown on previous page are selected with CTRL and Up/Down Cursors and shown Bottom Left of the screen.

## QBITS Program Arrays & Variable Assignment

The **BEEP** List entries are held by **BKey(96,7)** [0 .. 47 used for Octave C2 to B5]. **Score(9,23,4)** holds Data of the Group Pages and Row/Note Sequences. **Mkey(26)** holds the Stave Scale offset.

For the **BEEP Tracker Bkey(kn%,kp%)** **kn% 0 .. 96 : Kp% d,p,h,t,s,w,f,r**  
**d** duration, **p** pitch, **h** harmonic, **t** time interval, **s** step, **w** warp, **f** fuzzy, **r** random.

For the **Score Sheet Score(sl%,sn%,sp%)** **sl% 0 .. 9 : sn% 0 .. 23 : sp% 0 .. 4**  
**sl%** 10 Groups /Stave Rows, **sn%** 24 Notes, **sp%** 0=kn% 1=ds% 2=0 3=ar% 4=nv%  
**ds%** Score Symbol 0-15 : ar% 1-Staccato 2-Tenuto : **nv%** Note/Rest value 4 .. 0.25

## QBITS Screen Coordinates

WINDOW x,y Graphic coordination's previously used **across** and **up**, for the **Mini Keyboard** this became **ka%**, **ku%** and for the **Score Sheet** and for **Notes na%**, **nu%**. For keyboard location on the **Stave** the offset **so%** is added to **nu%**. For the display symbol, **ds%** to identify a Note or Rest etc.



## QBITS QLSoundSE - Data File Management

A QL SuperBASIC Filename can be 36 ASCII characters in length. QBITS Sound Data files are identified with '**QBS\_**' followed by up to 15-Alphanumeric characters. For LOAD first Select a Path, drive device with Up/Down Cursors then press '**S**' to carry out a search of available **QBS\_**Data files.

If none are found, Scroll for more devices with Up/Down Cursor keys. Select from file(s) found with Left/Right Cursors and press Enter. All being well, file will load and display as Instrumentals in Tracker Mode or if saved as a set of Musical Notes, Symbols are displayed on the Stave Rows.

To Save the current entries to the BEEP Tracker or Score Sheets, press '**E**' to Edit an existing or create a New Filename. The program checks the device and returns 'DEVICE ERROR' if not found. If File exists you are prompted to Overwrite Y/N. Choosing to Save, first the BEEP Attributes BKey(kn%,kp) are written to file, then the Notes Score(sl,sn,sp).

## QBITS QLSoundSE - Compatibility

The Coding was written and tested with the **QPC2** Emulator. When LRUN on other QL Platforms a number of problems were identified. For example Platforms running earlier ROM versions, integers not working with **FOR a%=1 to 4** or **SElect ON a% loop**. Single line embedded **FOR** loops without **END FOR**'s as with **END REpeat** statements missing name. The **QPC2 SMSQ/E** appears to be tolerant to unassigned variables setting them to '0' instead of '★' which incurs an error that freezes the program. Idiosyncrasies as the **CURSOR** graphics coordinate only working with **WINDOWS#1**. As mentioned before my programming skills have limits but I strive to improve. So to increase the compatibility of the **QLSoundSE** Prog across more QL Platforms the obvious solution was to reverse engineer the code problems.

## QBITS Program Performance

The Program will require Extended memory and TK2. Runing under the Interpreter on a BBQL the performance will be slow, a tenfold increase would make it more reasonable, Compiling another option. Used with an emulator such as **QPC2** on an up-to-date hardware platform, the speed is likely to be in the order of 1000 times faster.

## QBITS Exploring QL Sounds

<b>QBSBeepv1</b>	BEEP Prog... (shown on pages 8/9)
<b>QLUBMicrov1</b>	QLUB magazine Music Micro Please (shown on pages 12/13)
<b>QBQLSoundSE</b>	QBITS Exploring QL Sounds Main Prog... (see page 20)
<b>QBS_ScoreDemo</b>	A Demo Data File for <b>SCORE</b> Sheet Mode!
<b>QBS_TrackDemo</b>	A Demo Data File for <b>BEEP TRACKER</b> Mode!

## QBITS Summary

Although the QL BEEP Attributes hold the possibility for producing a large range of sounds and plausible musical Notes, its accompanying electronic components are limited. Nevertheless, I've enjoyed tinkering with the code in putting together this Prog and learning more about the actions of the QL BEEP command.

As to the future and the latent promises of the AY-3-8910 with its interesting past history, this might open the door to more musical innovations so read on!

## QPC2 Emulator - QSound

An implementation of two ABC-Electronic's QSound have been added to QPC2. In the Day the arrangement used AY-3-8910 sound generator chips. This programmable 3 voice generator has a past history of use with in a number of Arcade machines and 8bit computers in the 1980's.

The AY-3-8910 is set up in a series of sixteen 8bit registers, programmed over an 8bit bus that is toggled between Addressing mode, to select a register and Data mode for transfer of contents to that register. Frequencies equivalent to the top Octave of a piano keyboard can be defined with reasonable accuracy versus the accepted note values for an even-tempered scale to nearly 1 Hz precision and even more finely at lower pitches.

The wavelength to generate is held in two 8bit registers dedicated to channels ABC, but the value is limited to 12 bits giving a total of 4095 different Pitches. Despite the high maximum frequency, the ability to divide that figure by 4096 means the lowest directly definable output frequency is 30.6 Hz, roughly equal to **B0**, the third lowest note on an 88-key piano and as good as subsonic with everyday speaker systems. In essence, the chip is able to produce a decent musical output of Pitches found in most compositions. Another register controls the period of a pseudo-random noise generator with a total of 31 different cycle times, while another controls the mixing into the three primary channels. Three more registers control the volume of a channel or turn on/off envelope shapes, their timing cycle controlled by three more registers.

## QBITS QLSoundSE - QSound

Although the AY-3-8910 implementation offers the possibility to extend QLSounds, the **QSound PLAY** command is the only one of which short term I have established some experience.

The attributes for a **QSound PLAY** command:

NOTES	CDEGFAH	
Sharps	#	[#C]
Flats	b	[Cb]
RESTS	p	[1 unit length]
Octave	o0 ... o7	[Note range C0 to H7]
Volume	v0 ... v15	[v16 Switches to Envelope Waveform Control]
Duration	10 ... 1255	[Default 15]
Noise	n0 ... n31	[Default n0]
Warp Curve	w0 ... w15	[Default w0]
Warp length	x0 ... x3276	[default x0 - <b>Note</b> Max is only 4 digits not 5 - x32767]
Stop	s	
Activate Ch	r1 r2 r3	[Awaiting Channel]

**QBITS QLSoundSE** holds Note ID's in an array QNotes(qn) [qn == C,#C,D,#D,E,F,#F,G, #G, A, #A, H] the Octave and Note are identified as a Score is Read. Assigned to **PLAY** the variable **qstr\$** [q string] is constructed with Volume '**v16**' to activate a Warp Curve '**w10**' [a Triangular waveform] with Length '**x8500**'. To this is appended for each Note the Octave '**o**'&**oct**%& Note ID **QNotes(qn)**.

**PLAY 1,qstr\$:PAUSE dur : Sound\_AY : PAUSE del**

**dur** = duration of Note and **del** = delay between Notes [both counted in Multiples of 20msec]

**Note:** I hope that future development will utilise playback of both chip outputs simultaneously and extend the range of tonal qualities to accommodate various musical Instruments. **QSounds** investigations will continue and further options will be considered for future Reviews.

## QBITS PROCedures

<b>Init_win</b> : <b>QLSHelp</b>	Init Windows, Instructions
<b>QBITS_QLSounds</b>	Main Menu
<b>QExit</b>	Leave Program
<b>MBeep</b>	BEEP Attributes and TRACKER Screen
<b>MScore</b> : <b>Init_keys</b>	SCORE Sheet and MINI KEYBOARD Screen : Init Note Values
<b>QHash</b> ; <b>QBold(ch%,col%,cs%,cx%,xy%,str\$)</b>	Sharp for Keyboard Notes & Bold Character
<b>BSelect(chg%)</b> : <b>BPnt</b> : <b>BSet</b> : <b>BRead</b> : <b>Batt</b> : <b>BWrite(Chg%)</b> : <b>BWave</b>	Select BEEP Attributes
<b>TPatn(chg%)</b> : <b>TInst(chg%)</b> : <b>TEntry</b>	Select Tracker Pattern / Instrumental : Enter in Table
<b>TDecay(chg%)</b> : <b>TSwap(chg%)</b>	Set a Delay between rows : Swap rows Up/Down
<b>RowNu(scol,py%,pr%)</b> : <b>RowChg(chg%)</b> : <b>RowUp</b> : <b>RowDn</b> ; <b>TClear</b>	<b>Change</b> Tracker Row
<b>TLLoop</b>	Select two or more Rows to play in continue loop
<b>PTrack(pt%,rm%,rx%)</b>	Play selected rows
<b>SNew</b>	Clear Score for New Entices
<b>Note_Reset</b>	Reset to Default BEEP Note Sounds
<b>Set_QLSound</b>	Play Score with QSound Note Sounds
<b>SEntry</b>	Write Selected Note to next position on Score Sheet
<b>KChg(chg%)</b> : <b>NChg(chg%)</b> : <b>SChg(mp%,chg%)</b> : <b>RChg(chg%)</b>	Change Key Note Stave Row
<b>PScore</b> : <b>QNotes(qn)</b>	Play Score BEEP Notes or QSound
<b>DSymbol</b>	Displays selected Symbols
<b>SSpace</b>	Clears Stave position
<b>Ledger</b>	Add Ledger lines when required to extend Stave
<b>EBar</b> <b>SBar</b>	Draws End Bar : Separation bar
<b>Head, Stem, Hook1, Hook2</b>	Draws the Parts that make up a Note Symbol
<b>Semibreve</b> <b>SBRest</b>	Draws Note Rest Symbol                Beat value of 4
<b>Minim</b> <b>MRest</b>	Draws Note Rest Symbol                Beat value of 2
<b>Crochet</b> <b>CRest</b>	Draws Note Rest Symbol                Beat value of 1
<b>Quaver</b> <b>QRest</b>	Draws Note Rest Symbol                Beat value of ½
<b>Semiquaver</b> <b>SQRest</b>	Draws Note Rest Symbol                Beat value of ¼
<b>Sharp</b>	Pitch raised by ½ pitch
<b>Dot</b>	Beat played 1½ times normal length (Beat values 3, 1½, ¾)
<b>Staccato</b>	Notes played Pause = 8/10 of Beat (distinct)
<b>Tenuto</b>	Notes played Pause = Whole Beat (lengthy)
<b>GClef</b> <b>FClef</b>	Draws the GClef : FClef
<b>Tempo</b> <b>TPnt</b>	Select Beat & Metronome values:Prints Tempo Selection
<b>KLoad</b>	Load from Selected Filename into Prog Data Array Fields
<b>KSave</b>	Save to Selected Filename from Prog Data Array Fields
<b>SelPath</b> : <b>FList</b> : <b>Sfiles</b>	Select device & Data file ?
<b>FCheck</b>	Filename <b>QBS_</b> check against <b>DIR</b> File List
<b>ED_Str</b>	Filename Editor Delete Add Character
<b>QLSMenu</b>	(L)oad (S)ave (E)xit Restore Menu Part of File Management

## QBITS QLSoundSE – Code

1000 REMark **QBITS\_QLSoundSE\_bas** [QBITS QLSounds SE 2024 QL40th - QPC2] vM30

1002 dev\$='win1\_' :MODE 4:gx=0:gy=0 :REMark Basic Settings

1004 **WHEN ER**or :eck=1:**CONTINUE:END WHEN**

1006 REMark **Import QBITSSConfig Settings - QPC2**

1007 OPEN\_IN#9,dev\$&'QBITSConfig':INPUT#9,gx\gy\dn\$\dev\$\dn%\dm%

1008 DIM Drv\$(15,5):FOR d=0 TO 15:INPUT#9,Drv\$(d):END FOR d:CLOSE#9

1010 REMark **Set Arrays**

1011 REMark Bkeys(kn%,kp%) - kn%(0-96) kp%(0-8) d,p,h,t,s,w,f,r

1012 REMark Score(sl%,sn%,sp%) - sl%(0-9) sn%(0-23) sp%(0-4) kn%,ds%,ar%,nv%

1013 REMark Mkey(n%) - Stave offsets +16.5 to -3

1014 DIM Bkeys(96,7),Score(9,23,4),MKey(26)

1015 DIM SFn\$(20,15),OChk\$(21,2),str\$(30),TP\$(7,2,1)

1016 **RESTORE 1017**:FOR a=0 TO 26:**READ MKey(a)**

1017 DATA 16.5, 15, 15, 13.5, 13.5, 12, 12, 10.5, 9, 9, 7.5, 7.5, 6

1018 DATA 4.5, 4.5, 3, 3, 1.5, 1.5, 0, -1.5, -1.5, -3, -3, -4.5, -6, -6

1019 **RESTORE 1020**:FOR a=21 TO 1 STEP -1:**READ OChk\$(a)**

1020 DATA 'C2','C2','D2','D2','E2','F2','G2','G2','A2','A2','B2'

1021 DATA 'C3','C3','D3','D3','E3','F3','F3','G3','G3'

1022 **RESTORE 1023**:FOR i=2 TO 7:**READ TP\$(i,1),TP\$(i,2)**

1023 DATA '2','2','2','4','4','4','3','4','3','8','6','8'

1025 **DEFine PROCedure Init\_keys**

1026 **RESTORE 1027**:FOR a=1 TO 48:**READ Bkeys(a,1)**

1027 DATA 3,4,5,6,7,8,9,10,11,12,13,14

:REMark B5 to C5

1028 DATA 16,17,19,20,22,24,26,27,28,30,32,35

:REMark B4 to C4

1029 DATA 37,40,43,46,49,53,56,60,64,69,73,78

:REMark B3 to C3

1030 DATA 83,89,94,101,107,114,121,129,137,146,155,165

:REMark B2 to C2

1031 **END DEFine**

1033 REMark **Set Variables**

1034 tr%=0:pr%=1:py%=0:r%=0

:REMark Tracker check/Pattern Row/Screen Row/Release

1035 kn%=23:kp%=0:ds%=12:as%=0:ar%=0

:REMark Notes/Parameter/Symbol/Articulates

1036 bn%=1:mn%=200:m%=0:mp%=2

:REMark Beat/Metrenome/Marker/Row Pointer

1037 sl%=0:sn%=0:sp%=0

:REMark Score Line/Number/Parameter

1038 n%=1:f%=1:eck%=0:ck%=0

:REMark MKey Num : File num :Checks

1039 SD\$='QBS\_' :SFn\$(f%)='Demo'

:REMark QBITSSound\_ Sound FileName

1041 **Init\_win:QLSHelp:init\_keys:MBeep:QBITS\_QLSounds**

1043 **DEFine PROCedure QLSHelp**

1044 **QBold 0,6,1,6,4,'Explore QLSounds'**

1045 **CURSOR#0,6,16:PRINT#0,'Select (M)ode for or Sheet'**

1046 **CURSOR#0,258,4:PRINT#0,'Use Cursor keys Spacebar Enter'**

1047 **CURSOR#0,48,16:PRINT#0,'(M)ode':CURSOR#0,326,16:PRINT#0,'low high'**

1048 **QBold 0,6,1,112,16,'BEEP':QBold 0,6,1,160,16,'SCORE'**

1049 **QBold 0,6,1,270,16,'TRACKER':QBold 0,6,1,416,16,'KEYBOARD'**

1050 **QBold 0,6,2,347,16,'< >':QBold 0,6,1,350,4,'←↑↓→'**

1051 **BLOCK#0,14,3,438,8,6 :QBold 0,6,1,488,4,'←':BLOCK#0,2,4,496,6,6**

1052 **END DEFine**

```

1054 DEFine PROCEDURE QBITS_QLSounds
1055 REPEAT Mlp
1056 k=CODE(INKEY$(-1))
1057 SElect ON k
1058 =35 :IF m%=1:Note_Reset :BLOCK 200,10,0,38,0 :REMark [#] Note Reset
1059 =70,102 :IF m%=1:FClef :REMark [F] Clef
1060 =81,113 :IF m%=1:Set_QSound :BLOCK 200,10,0,38,0 :REMark [Q] PLAY
1061 =84,116 :IF m%=0:TLoop :REMark [T]racker Loop
1062 =192 :IF m%=1:SChg mp% ,-1:ELSE IF tr%=1:TPatn -1 :REMark Left SCol/Patn
1063 =200 :IF m%=1:SChg mp%,+1:ELSE IF tr%=1:TPatn +1 :REMark Right SCol/Patn
1064 =208 :IF m%=1:RChg -1 :ELSE RowChg -1 :REMark Up SRow/PRow
1065 =216 :IF m%=1:RChg +1 :ELSE RowChg +1 :REMark Down SRow/PRow
1066 =210 :IF m%=1 AND ds> 0:NChg -1:ELSE TSwap -1 :REMark Ctrl Up Note/Row
1067 =218 :IF m%=1 AND ds<16:NChg +1:ELSE TSwap +1 :REMark Ctrl Dn Note/Row
1068 =196 :IF m%=0:BWrite -1 :REMark Shift Left BEEP
1069 =204 :IF m%=0:BWrite +1 :REMark Shift Right BEEP
1070 =212 :IF m%=0:BSelect -1 :REMark Shift Up BEEP
1071 =220 :IF m%=0:BSelect +1 :REMark Shift Down BEEP
1072 =193 :IF tr%=1:TInst -1:ELSE IF kn%<47:KChg +1 :REMark ALT ← Tk/Kbrd
1073 =201 :IF tr%=1:TInst +1:ELSE IF kn%> 0:KChg -1 :REMark ALT → Tk/Kbrd
1074 =45,95 :IF tr%=1:TDecay -1 :REMark - Delay
1075 =43,61 :IF tr%=1:TDecay +1 :REMark + Delay
1076 =67,99 :IF tr%=1:TClear :REMark (C)lear TkEntry
1077 =77,109 :IF m%=0:m%=1:MScore:ELSE m%=0:MBeep :REMark (M)ode BEEP/SCORE
1078 =76,108 :KLoad:IF m%=1:RChg 0:ELSE IF tr%=1:TPatn 0 :REMark (L)oad
1079 =83,115 :KSave :REMark (S)ave
1080 =66,98 :IF m%=1:tr%=0:Tempo :REMark (B)eat
1081 =69,101 :QExit :BLOCK 18,10,172,24,0 :REMark (E)xit
1082 =78,110 :IF m%=0:BEEP 0,p:ELSE SNew:|=1 :REMark (N)ote/(N)ew
1083 =72,104 :IF m%=0:BEEP d,p,h,t,s,0,0,0 :REMark (H)armonic
1084 =87,119 :IF m%=0:BEEP d,p,h,t,s,w,0,0 :REMark (W)raps
1085 =70,102 :IF m%=0:BEEP d,p,h,t,s,w,f,0 :REMark (F)uzz
1086 =82,114 :IF m%=0:BEEP d,p,h,t,s,w,f,r :REMark (R)andom
1087 =80,112 :IF m%=1:PScore:ELSE PTrack 0,rm%,rx% :REMark (P)lay/(p)lay
1088 =10 :IF m%=1:SEntry:ELSE BSet:IF tr%=1:TEntry :REMark Score/Tracker
1089 =32 :IF m%=0:BEEP :ELSE IF mp%=2:mp%=-2:ELSE mp%=2:END IF :SChg mp%,0
1090 =49 :IF ar%=0:ar%=1 :ELSE ar%=0:END IF :NChg 0 :REMark (1)Staccato
1091 =50 :IF ar%=0:ar%=2 :ELSE ar%=0:END IF :NChg 0 :REMark (2)Tenuto
1092 END SElect
1093 END REPEAT Mlp
1094 END DEFine

1096 DEFine PROCEDURE QExit
1097 INK 7:CURSOR 172,22:PRINT 'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$:STOP:ELSE RETURN
1098 END DEFine

```

Note: Help screen

Explore QLSounds

Select (M)ode for BEEP or SCORE Sheet

Use Cursor keys ↑↓↔ Spacebar — Enter ↵

TRACKER low < > high KEYBOARD

## 1100 REMark BEEP Attributes

```
1102 DEFINE PROCEDURE BSelect(chg%) :REMark Select BEEP Parameter
1103 INK 3:BPmt:kp%=kp%+chg%:IF kp%<0 OR kp%>7:kp%=0:END IF :INK 7:BPmt
1104 END DEFINE
```

```
1106 DEFINE PROCEDURE BPmt :REMark Draw BEEP Parameter
1107 B$=Bkeys(n%,kp%):CURSOR 72,62+kp%*10:PRINT FILL$(' ',5-LEN(B$))&B$
1108 END DEFINE
```

```
1110 DEFINE PROCEDURE BSet :REMark Set BEEP Parameters
1111 INK 3:FOR i=0 TO 7:kp%=p:BPmt:END FOR i:BRead:BAtr:BWave:kp%=0
1112 END DEFINE
```

```
1114 DEFINE PROCEDURE BRead :REMark Read BEEP Parameters
1115 d=Bkeys(n%,0):p=Bkeys(n%,1):h=Bkeys(n%,2):t=Bkeys(n%,3):s=Bkeys(n%,4)
1116 w=Bkeys(n%,5):f=Bkeys(n%,6):r=Bkeys(n%,7)
1117 END DEFINE
```

```
1119 DEFINE PROCEDURE BAtr :REMark Show BEEP Attributes
1120 BRead:d=d*150:t=t*150:REMark IF tr=1:RowPty%,Score(sl,sn,0)
1121 PRINT#4,'BEEP: ',FILL$('0',5-LEN(d))&d;' ';FILL$('0',3-LEN(p))&p;
1122 PRINT#4,[' ' ;FILL$('0',3-LEN(h))&h;' ' ;FILL$('0',5-LEN(t))&t;' ' ;
1123 PRINT#4, FILL$(' ',2-LEN(s))&s;' ' ;FILL$('0',2-LEN(w))&w;' ' ;
1124 PRINT#4, FILL$('0',2-LEN(f))&f;' ' ;FILL$('0',2-LEN(r))&r:CLS#4,4
1125 END DEFINE
```

**Note:** Use Shift Cursor Key to change the BEEP Attributes.  
Move Up/Down to select Attribute and Left/Right Cursors to change values.

Duration 0 is continuous, the range 0 to 218 set in approximately 12 msec steps [15x12=180sec as a minimum to be heard].  
Mixing pitch with a Harmonic is set by the Time step and Pitch step to create a Timbre or Characteristic quality to the sound.  
Wraps create Scaling up and down making Wailing or Siren sounds. Fuzz and Random add Noise and Ripple effects.

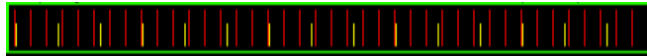
```
BEEP [Shift ↑↓↓+]
Duration : 0 (0-218) ADSR
Pitch : 10 (0-255) Note
Harmonic : 15 (0-255) \
Time Step : 218 (0-218) Timbre
Pitch Step: 7 (-8..7) /
Wraps : 15 (0-15) Loops
Fuzz : 0 (0-15) Noise
Random : 0 (0-15) Ripples
```

```
1127 DEFINE PROCEDURE BWrite(chg%) :REMark Write BEEP Parameter
1128 Bkeys(n%,kp%)=Bkeys(n%,kp%)+chg%:BRead
1129 IF d< 0:Bkeys(n%,0)=218:ELSE IF d>218:Bkeys(n%,0)= 0
1130 IF p< 0:Bkeys(n%,1)=255:ELSE IF p>255:Bkeys(n%,1)= 0
1131 IF h< 0:Bkeys(n%,2)=255:ELSE IF h>255:Bkeys(n%,2)= 0
1132 IF t< 0:Bkeys(n%,3)=218:ELSE IF t>218:Bkeys(n%,3)= 0
1133 IF s<-8:Bkeys(n%,4)= -8:ELSE IF s> 7:Bkeys(n%,4)= 7
1134 IF w<0:Bkeys(n%,5)= 0:ELSE IF w> 15:Bkeys(n%,5)=15
1135 IF f< 0:Bkeys(n%,6)= 0:ELSE IF f> 15:Bkeys(n%,6)=15
1136 IF r< 0:Bkeys(n%,7)= 0:ELSE IF r> 15:Bkeys(n%,7)=15
1137 INK 7 :BPmt
1138 END DEFINE
```

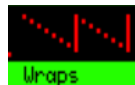
## 140 DEFine PROCEDURE BWave

:REMark Show BEEP Graphics

```
1141 BLOCK 484,20,8,186,0:BLOCK 160,24,332,150,0
1142 IF p>0:FOR i=0 TO 476 STEP p:BLOCK 1,16,12+i,188,2
1143 IF h>0:FOR i=0 TO 476 STEP h:BLOCK 1,10,13+i,194,6
```



```
1144 IF s>0
1145   bs=s:ba=348:bu=154 :REMark Time & Pitch Step /\
1146   FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu+i*2,2
1147   ba=348+bs*4:bu=154+bs*2
1148   FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu-i*2,2
1149 END IF
1150 IF s<0
1151   bs=SQRT(s*s):ba=348:bu=154+bs*2 :REMark Time & Pitch Steps V
1152   FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu-i*2,2
1153   ba=348+bs*4:bu=154
1154   FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu+i*2,2
1155 END IF
1156 IF w<=8 AND w>0
1157   bw=w:wa=bw*4:wu=154 :REMark Wraps /\|
1158   FOR i=0 TO bw:BLOCK 2,2,420+i*4,wu+i*2,2
1159   BLOCK 2,2*bw,420+bw*4,wu,2:BLOCK 2,2*bw,420+bw*8,wu,2
1160   FOR i=0 TO bw:BLOCK 2,2,420+wa+i*4,wu+i*2,2
1161 END IF
1162 IF w>8
1163   bw=w-8:wa=bw*4:wu=154+bw*2 :REMark Wraps \|
1164   FOR i=0 TO bw:BLOCK 2,2,420+i*4,wu-i*2,2
1165   BLOCK 2,2*bw,420+bw*4,156,2:BLOCK 2,2*bw,420+bw*8,156,2
1166   FOR i=0 TO bw:BLOCK 2,2,420+wa+i*4,wu-i*2,2
1167 END IF
1168 END DEFine
```



## 1200 REMark BEEP TRACKER

1202 DEFine PROCEDURE TPatn(chg%) :REMark Change Tracker Pattern

```
1203 IF chg%= -1 AND sl%>0:sl%=sl%+chg%
1204 IF chg%=+1 AND sl%<9:sl%=sl%+chg%
1205 FOR i=0 TO 7:RowPt 0,i,i+rm%
1206 CURSOR#3,36,25:PRINT#3,FILL$('0',2-LEN(sl%))&sl%
1207 RowNu 0,py%:py%=0:pr%=rm%:RowNu 7,py%:RowPt 2,py%,pr%
1208 END DEFine
```



1210 DEFine PROCEDURE TInst(chg%) :REMark Change Tracker Instrumental

```
1211 IF chg%= -1 AND n%> 0 :n%=n%-1
1212 IF chg%=+1 AND n%<95:n%=n%+1
1213 CURSOR#3,142,2:PRINT#3,FILL$('0',2-LEN(n%))&n%:BSet
1214 END DEFine
```

BEEP TRACKER <23> + Select from List

1216 DEFine PROCEDURE TEntry :REMark Update BEEP/Tracker Entry

```
1217 Score(sl%,pr%,0)=n%:Score(sl%,pr%,3)=r%:RowPt 2,py%,pr%
1218 END DEFine
```

```

1220 DEFINE PROCEDURE RowNu(col,py%) :REMark Show Row Number
1221 INK#3,col:CURLSOR#3,37,25:PRINT#3,FILL$('00',2-LEN(pr%))&pr%
1222 END DEFINE

```

```

1224 DEFINE PROCEDURE RowPt(scol,py%,pr%) :REMark Print Tracker Row

```

```

1225 FOR a=0 TO 7

```

```

1226 num%=Score(sl%,pr%,0):str$=BKey(num%,a):STRIP#5,scol

```

```

1227 num$=FILL$'0',2-LEN(num%))&num%:CURLSOR#5,0,py%:PRINT#5,num$

```

```

1227 SELECT ON a

```

```

1228 =0 :CURLSOR#5, a*22 ,py%*11:PRINT#5,FILL$('0',5-LEN(str%))&str%

```

```

1229 =1 :CURLSOR#5,32+a*22 ,py%*11:PRINT#5,FILL$('0',3-LEN(str%))&str%

```

```

1230 =2 :CURLSOR#5,36+a*22 ,py%*11:PRINT#5,FILL$('0',3-LEN(str%))&str%

```

```

1231 =3 :CURLSOR#5,36+a*22 ,py%*11:PRINT#5,FILL$('0',5-LEN(str%))&str%

```

```

1232 =4 :CURLSOR#5,136 ,py%*11:PRINT#5,FILL$('+',2-LEN(str%))&str%

```

```

1233 =5,6,7:CURLSOR#5,76+a*16,py%*11:PRINT#5,FILL$('0',2-LEN(str%))&str%

```

```

1234 END SELECT

```

```

1235 r%=Score(sl%,pr%,3):CURLSOR#5,212,py%*11:PRINT#5,FILL$('0',2-LEN(r%))&r%

```

```

1237 END FOR a

```

```

1238 END DEFINE

```

```

1240 DEFINE PROCEDURE TDecay(chg%) :REMark Change Tracker Decay

```

```

1241 IF chg%=-1 AND r%> 0:r%=r% -1

```

```

1242 IF chg%=+1 AND r%<15:r%=r%+1

```

```

1243 Score(sl%,pr%,3)=r%:RowPt 0,py%,pr%

```

```

1244 END DEFINE

```

Note: Sets Delay between Rows being Played

[0..15]

```

1246 DEFINE PROCEDURE TSwap(chg%) :REMark Swap Rows

```

```

1247 IF chg%=-1 AND pr%= 0 OR chg%=-1 AND py%=0:RETURN

```

```

1248 IF chg%=+1 AND pr%>22 OR chg%=+1 AND py%=7:RETURN

```

```

1249 DIM Tmp(4):RowNu 0,py%:RowPt 0,py%+chg%,pr%

```

```

1250 FOR i=0 TO 4

```

```

1251 Tmp(i)=Score(sl%,pr%+chg%,i)

```

```

1252 Score(sl%,pr%+chg%,i)=Score(sl%,pr%,i)

```

```

1253 Score(sl%,pr%,i)=Tmp(i)

```

```

1254 END FOR i

```

```

1255 py%=py%+chg%:pr%=pr%+chg%:RowNu 7,py%:RowPt 2,py%,pr%

```

```

1256 END DEFINE

```

```

1258 DEFINE PROCEDURE RowChg(chg%) :REMark Change Tracker Row

```

```

1259 IF tr%=0:RETURN :ELSE RowNu 0,py% :RowPt 0,py%,pr%

```

```

1260 py%=py%+chg%:pr%=pr%+chg%:RowUp:RowDn:RowNu 7,py%:RowPt 2,py%,pr%

```

```

1261 END DEFINE

```

```

1263 DEFINE PROCEDURE RowUp

```

```

1264 IF py%<0 AND pr%> 0:SCROLL#5, 11:py%=0:RowPt 0,py%,pr%

```

```

1265 IF pr%<1:pr%=0:py%=0

```

```

1266 END DEFINE

```

```

1268 DEFINE PROCEDURE RowDn

```

```

1269 IF py%>7 AND pr%<23:SCROLL#5,-11:py%=7:RowPt 0,py%,pr%

```

```

1270 IF pr%>=23:pr%=23:py%=7

```

```

1271 END DEFINE

```



```

1273 DEFINE PROCEDURE TClear :REMark Clear Tracker Entry
1274 FOR i=0 TO 7:Bkeys(n%,i)=0
1275 FOR i=0 TO 4:Score(sl%,pr%,i)=0
1276 RowPt 2,py%,pr%
1277 END DEFINE

```



```

1279 DEFINE PROCEDURE TLoop :REMark Play Continuous Loop
1280 IF Bkeys(Score(sl%,rm%,0),1)=0:RETURN
1281 pt%=1:CURSOR#3,136,128:PRINT#3,' ← →0↑ ↓ ← ':BLOCK#3,2,4,202,130,7
1282 REPEAT Row_ip
1283 IF rx%<=rm%:rx%=rm%+1
1284 rm$=FILL$('0',2-LEN(rm%))&rm%:rx$=FILL$('0',2-LEN(rx%))&rx%
1285 CURSOR#3,136,128:PRINT#3,' ← ':rm$;' →0↑ ':rx$;' ↓':k=CODE(INKEY$(-1))
1286 SELECT ON k
1287 =10:PTrack pt%,rm%, rx%:EXIT Row_ip
1288 =192:IF rm%> 0 :rm%=rm%-1
1289 =200:IF rm%<22 :rm%=rm%+1
1290 =208:IF rx% <23 :rx%=rx%+1
1291 =216:IF rx% >rs%+1:rx%=rx%-1
1292 END SELECT
1293 END REPEAT Row_ip
1294 CURSOR#3,136,128:PRINT#3,' ':rm$;' ↑ ':rx$;'
1295 END DEFINE

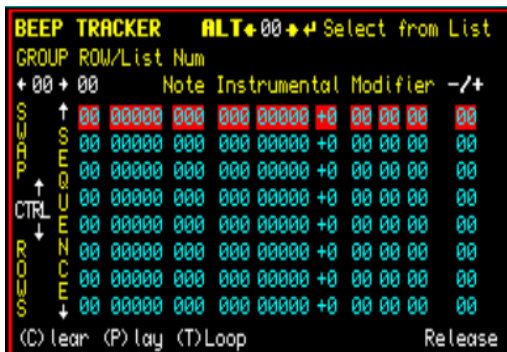
```



```

1297 DEFINE PROCEDURE PTrack(pt%,rm%,rx%) :REMark Play Tracker Pattern
1298 BLOCK#3,20,10,190,128,0: BLOCK#3,12,3,194,132,7
1299 REPEAT Track_ip
1300 TPatn 0:RowChg 0
1301 FOR row=rm% TO rx%
1302 n%=Score(sl%,row,0):BRead:dur=d:del=Score(sl%,row,3)
1303 IF dur=0 AND p>0:dur=25:ELSE NEXT row
1304 BEEP 0,p,h,t,s,w,f,r:PAUSE dur:BEEP:PAUSE del:RowChg 1
1305 IF KEYROW(1)=64:pt%=0:RETURN
1306 END FOR row
1307 END REPEAT Track_ip
1308 END DEFINE

```



**Note** (P)lay will play all rows in Pattern Group 0 to 23 Unless (T)Loop has rearranged for a subset of two or more rows to be played. To play selected rows in a continuous loop they must be actioned with (T)Loop.

## 1350 REMark SCORE Sheet

### 1352 DEFine PROCEDURE SNew

```
1353 BLOCK 420,60,76, 76,4:FOR i=0 TO 12 STEP 3:LINE 30,56+i TO 200,56+i
1354 BLOCK 420,60,76,147,4:FOR i=0 TO 12 STEP 3:LINE 32,16+i TO 200,16+i
1355 DIM Score(9,23,4):sl%=0:bn%=1:mn%=200:RChg 0:sn%=0:SChg 1,0
1356 END DEFine
```

### 1358 DEFine PROCEDURE Note\_Reset

```
1359 CURSOR 8,38:PRINT 'Set Notes to Default Y/N'
1360 PAUSE :IF KEYROW(5)=64:CURSOR 170,38:PRINT 'OK ':Init_keys:PAUSE 30
1361 END DEFine
```

Set Notes to Default Y/N

### 1363 DEFine PROCEDURE Set\_QSound

```
1364 CURSOR 8,38:PRINT 'Set Notes with QSound Y/N'
1365 PAUSE :IF KEYROW(5)=64:CURSOR 170,38:PRINT 'OK'
1366 END DEFine
```

Set Notes with QSound Y/N

### 1368 DEFine PROCEDURE SEntry

:REMark Score Entry Settings

```
1369 IF mp%=2:nu%=56:l%=sl%:ELSE nu%=16:l%=sl%+1
1370 Score(l%,sn%,0)=kn% :Score(l%,sn%,1)=ds%
1371 IF ar%=1 OR ar%=2 :Score(l%,sn%,3)=ar%:ELSE Score(l%,sn%,3)=0
1372 na%=36+sn%*7:DSymbol :Score(l%,sn%,4)=nv%:ac%=0:ar%=0:SChg mp%,1
1373 END DEFine
```

### 1375 DEFine PROCEDURE KChg(chg%)

:REMark Change MINI Key

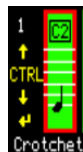
```
1376 IF tr%=0:OVER#3,-1:BLOCK#3,5,4,ka%,ku%,7:OVER#3,0
1377 kn%=kn%+chg%:ka%=276-kn%*5:ku%=50:NChg 0:n%=ns%+kn%+1
1378 IF kn%> 6:ka%=271-kn%*5
1379 IF kn%>11:ka%=266-kn%*5
1380 IF kn%>18:ka%=261-kn%*5
1381 IF kn%>23:ka%=256-kn%*5
1382 IF kn%>30:ka%=251-kn%*5
1383 IF kn%>35:ka%=246-kn%*5
1384 IF kn%>42:ka%=241-kn%*5
1385 SElect ON kn%=1,3,5,8,10,13,15,17,20,22,25,27,29,32,34,37,39,41,44,46:ku%=36
1386 IF m%=0 :BSet :ELSE OVER#3,-1:BLOCK#3,5,4,ka%,ku%,7:OVER#3,0
1387 IF kn%<48:CURSOR#3,142,2:PRINT#3,FILL$(0',2-LEN(kn%))&kn%
1388 END DEFine
```



### 1390 DEFine PROCEDURE NChg(chg%)

:REMark Change Note Symbol

```
1391 IF m%=0:RETurn :ELSE na%=14:nu%=16:BLOCK 100,10,0,208,0:INK 7
1392 ds=ds+chg%:IF ds>15:ds=15:END IF :DSymbol:INK 7:CURSOR 4,208:PRINT N$
1393 END DEFine
```



### 1395 DEFine PROCEDURE SChg(mp%,chg%)

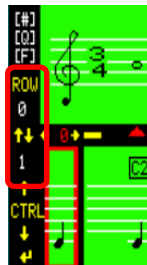
:REMark Change Stave Ponter Up/Dn Row

```
1396 sn%=sn%+chg%:IF sn%<0 OR sn%>23:sn%=0
1397 BLOCK 416,8,80,138,0:INK 2:ma%=36+sn%*7:mu%=42
1398 CURSOR 29,137:PRINT FILL$( ' ',2-LEN(sn%))&sn%
1399 FILL 1:LINE ma%-2,mu% TO ma%,mu%+mp% TO ma%+2,mu% TO ma%-2,mu%:FILL 0:INK 7
1400 END DEFine
```

```

1402 DEFINE PROCEDURE RChg(chg%) :REMark Change Displayed Rows
1403 sl%=sl%+chg%:tn%=kn%
1404 IF sl%< 0:sl%=0
1405 IF sl%> 8:sl%=8
1406 IF sl%<9:CURSOR 6,124:PRINT sl%:CURSOR 6,150:PRINT sl%+1
1407 FOR sn=0 TO 23
1418   na%=36+sn*7
1419   nu%=56:kn%=Score(sl% ,sn,0) :ds=Score(sl% , sn,1)
1410   ar%=Score(sl% , sn,3) :DSymbol
1411   nu%=16:kn%=Score(sl%+1,sn,0) :ds=Score(sl%+1,sn,1)
1412   ar%=Score(sl%+1,sn,3) :DSymbol
1413 END FOR sn
1414 kn%=tn%:ds=11:sn%=0:NChg 1,0 :TPrint bn%
1415 END DEFINE

```



```

1417 DEFINE PROCEDURE PScore :REMark Play Score
1418 LOCAL l,lmin,lmax:mp%=1:nt%=kn% : IF k=81 OR k=113:Qk=1:ELSE QK=0
1419 IF k=112 OR k=113 :lmin=sl%:lmax=sl%+1:ELSE lmin=0:lmax=9:sl%=0:IF tr%=0:RChg 0
1420 REPEAT Play_lp
1421   FOR l=lmin TO lmin+1
1422     IF l=lmin:mp%=3:ELSE mp%=-3
1423     FOR n=0 TO 23
1424       ds =Score(l,n,1):IF ds%<3:NEXT n
1425       n% =Score(l,n,0):sn%=n:Schg mp%,0
1426       nv%=Score(l,n,4):dur=3000*nv%/mn%:del=5
1427       IF Score(l,n,3)=1:del=8:dur=dur-1:REMark Staccato
1428       IF Score(l,n,3)=1:del=8:dur=dur-1:REM Tenuto
1429       IF ds<8:PAUSE dur+del
1430       IF ds>7:BRead:BEEP 0,p,h,t,s,w,f,r:PAUSE dur:BEEP:PAUSE del
1431       IF ds>7 AND Qk=1
1432         qn=n% : oct%=5-(n% DIV 12)
1433         qstr$='v16w10x8500o'&oct%&QNotes(qn)
1434         SOUND_AY :PLAY 1,qstr$:PAUSE dur:SOUND_AY:PAUSE del
1435       END IF
1436     END FOR n
1437   END FOR l
1438   IF lmin<=lmax-2:lmin=lmin+2:RChg 2:ELSE EXIT Play_lp
1439 END REPEAT Play_lp
1440 BEEP :kn%=nt%:sn%=0:ds=12:NChg 0:mp%=2:Schg 1,0
1441 END DEFINE

```

Note: Get Keyboard Note & Octave

```

1443 DEFINE FuNction QNotes(qn)
1444 RESTORE 1447:oct%='o'&oct%:not%=11-qn MOD 12
1445 FOR i=0 TO 11:READ not$:IF not%!=i:str$=oct%&not$:RETURN str$
1446 DATA 'C','C#','D','D#','E','F','F#','G','G#','A','A#','H'
1447 END DEFINE

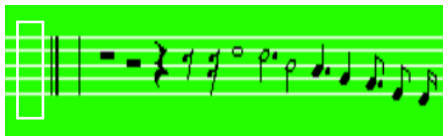
```



```

1450 DEFINE PROCEDURE DSymbol :REMark Select Music Symbol
1451 SSpace:INK 0:as%=0:n%=kn%:sym=ds%:shp=kn%:IF kn%>27:n%=kn% MOD 27+3
1452 IF ds%>7
1453   SELECT ON shp=1,3,5,8,10,13,15,17,20,22,25,27,29,32,34,37,39,41,44,46:as%=1
1454 END IF
1455 IF ds%>7 AND MKey(n%)=-6 :lu%=nu% -3:Ledger:lu%=lu%-3:Ledger
1456 IF ds%>7 AND MKey(n%)=-4.5 :lu%=nu% -3:Ledger
1457 IF ds%>7 AND MKey(n%)=-3 :lu%=nu% -3:Ledger
1458 IF ds%>7 AND MKey(n%)=13.5 :lu%=nu%+12:Ledger
1459 IF ds%>7 AND MKey(n%)>=15 :lu%=nu%+12:Ledger:lu%=lu%+3:Ledger
1460 IF ds%>7:nu%=nu%+MKey(n%) :IF kn%=27:nu%=30:n%=3
1461 SELECT ON sym
1462   =0:nv%=0 :N$='Space'
1463   =1:nv%=0 :EBar :N$='End Bar'
1464   =2:nv%=0 :SBar :N$='Bar Seperator'
1465   =3:nv%=4 :SBRest :N$='Semibreve Rest'
1466   =4:nv%=2 :MRest :N$='Minim Rest'
1467   =5:nv%=1 :CRest :N$='Crotchet Rest'
1468   =6:nv%=.5 :QRest :N$='Quaver Rest'
1469   =7:nv%=.25 :QRest:SRest :N$='SemiQuaver Rest'
1470   =8:nv%=4 :Semibreve :N$='Semibreve'
1471   =9:nv%=3 :Minim :Dot :N$='Minim+Dot'
1472   =10:nv%=2 :Minim :N$='Minim'
1473   =11:nv%=1.5 :Crotchet :Dot :N$='Crotchet+Dot'
1474   =12:nv%=1 :Crotchet :N$='Crotchet'
1475   =13:nv%=.75 :Quaver : Dot :N$='Quaver+Dot'
1476   =14:nv%=.5 :Quaver :N$='Quaver'
1477   =15:nv%=.25 :Semiquaver :N$='Semiquaver'
1478 END SELECT
1479 IF ds%>7 AND kn%>26 :ONote nu%
1480 IF ds%>7 AND as% =1 :Sharp
1481 IF ds%>7 AND ar% =1 :Staccato
1482 IF ds%>7 AND ar% =2 :Tenuto
1483 END DEFINE

```



```

1485 DEFINE PROCEDURE SSpace :REMark Clear Stave Entry
1486 LOCAL x,y,su:x=na%-2.5:y=nu%+21.8:INK 4
1487 FILL 1:LINE x,y TO x+6.5,y TO x+6.5,y-30.8 TO x,y-30.8 TO x,y:FILL 0
1488 INK 7:FOR su=0 TO 12 STEP 3:LINE x,nu%+su TO x+7,nu%+su
1489 END DEFINE

```



```

1491 DEFINE PROCEDURE Ledger :REMark Extra Stave line
1492 INK 7:LINE na%-2.5,lu% TO na%+4.5,lu%:INK 0
1493 END DEFINE

```

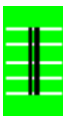


```

1495 DEFINE PROCEDURE ONote(nu) :REMark Note Octave
1496 INK 0:x1=na%-2:x2=na%+3.5:y1=lu%:C$=Ochk$(kn%-26)
1497 SELECT ON nu=13 TO 24:y1=36
1498 SELECT ON nu=25 TO 33:y1=13
1499 SELECT ON nu=53 TO 64:y1=77
1500 SELECT ON nu=65 TO 73:y1=53
1501 STRIP 4:C$=CURSOR x1,y1,1,1:PRINT C$:STRIP 0
1502 LINE x1,y1 TO x2,y1 TO x2,y1-5.5 TO x1,y1-5.5 TO x1,y1
1503 END DEFINE

```





1505 **DEFine PROCEDURE EBar** :REMark End Bar  
 1506 FILL 1:LINE na%+1,nu% TO na%+1,nu%+12 TO na%+1.6,nu%+12  
 1507 LINE TO na%+1.6,nu% TO na%+1,nu%:FILL 0:SBar  
 1508 **END DEFine**



1510 **DEFine PROCEDURE SBar** :REMark Bar  
 1511 LINE na%,nu% TO na%,nu%+12.5  
 1512 **END DEFine**



1514 **DEFine PROCEDURE SBRest** :REMark Semibreve Whole Note  
 1515 FILL 1:LINE na%-1,nu%+7.5:LINE\_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0  
 1516 **END DEFine**



1518 **DEFine PROCEDURE MRest** :REMark Minim Half Note  
 1519 FILL 1:LINE na%-1,nu%+6.2:LINE\_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0  
 1520 **END DEFine**



1522 **DEFine PROCEDURE CRest** :REMark Crotchet Quarter Note  
 1523 LINE na%+2,nu%+4.5:FILL 1:ARC\_R TO -2,-3,3\*PI/4 TO 1,4,-3\*PI/4:LINE\_R TO -2,1  
 1524 ARC\_R TO 0,4,5,3\*PI/4:LINE\_R TO 3,-2:ARC\_R TO 1,-4,5,3\*PI/4:FILL 0  
 1525 **END DEFine**



1527 **DEFine PROCEDURE QRest** :REMark Quaver Eighth Note  
 1528 LINE na%+.8,nu%+3 TO na%+2,nu%+9:LINE\_R TO -2,-2,  
 1529 FILL 1:CIRCLE\_R 0,.6,.6:FILL 0  
 1530 **END DEFine**



1532 **DEFine PROCEDURE SRest** :REMark Semiquaver Sixteenth Note  
 1533 LINE na%,nu% TO na%+1.5,nu%+6 TO na%-.5,nu%+3.5  
 1534 FILL 1:CIRCLE\_R 0,.8,.6:FILL 0  
 1535 **END DEFine**



1537 **DEFine PROCEDURE Head**  
 1538 CIRCLE na%,nu%,1.5,.6,-PI/4  
 1539 **END DEFine**

1541 **DEFine PROCEDURE Stem**  
 1542 IF n%<13  
 1543 LINE na%-1.1,nu%-.5 TO na%-1.1,nu%-6  
 1544 ELSE  
 1545 LINE na%+1.2,nu%+.5 TO na%+1.2,nu%+6  
 1546 END IF  
 1547 **END DEFine**



1549 **DEFine PROCEDURE Flag1**  
 1550 IF n%<13:LINE\_R TO 2,1.5 TO 0,2.5:ELSE LINE\_R TO 2,-1.5 TO 0,-2.5  
 1551 **END DEFine**

1553 **DEFine PROCEDURE Flag2**  
 1554 IF n%<13:LINE\_R TO 0,-1 TO -2,-1:ELSE LINE\_R TO 0,1 TO -2,1  
 1555 **END DEFine**



1557 **DEFine PROCEDURE Semibreve**  
1558 CIRCLE na%,nu%,1.4,.7,PI/2  
1559 **END DEFine**

1561 **DEFine PROCEDURE Minim**  
1562 **Head:Stem**  
1563 **END DEFine**



1565 **DEFine PROCEDURE Crotchet**  
1566 FILL 1:Head:FILL 0:Stem  
1567 **END DEFine**

1569 **DEFine PROCEDURE Quaver**  
1570 **Crotchet na%,nu%:Flag1**  
1571 **END DEFine**



1573 **DEFine PROCEDURE Semiquaver**  
1574 **Quaver na,nu:Flag2**  
1575 **END DEFine**



1577 **DEFine PROCEDURE Sharp**  
1578 LINE na%-2.5,nu%+4 TO na%,nu%+4.5:LINE na%-2.5,nu%+3 TO na%,nu%+3.5  
1579 LINE na%-2,nu%+4.5 TO na%-2,nu%+2:LINE na%-1,nu%+5 TO na%-1,nu%+2.5  
1580 **END DEFine**

1582 **DEFine PROCEDURE Dot**  
1583 FILL 1:CIRCLE na%+2.5,nu%,6:FILL 0  
1584 **END DEFine**



1586 **DEFine PROCEDURE Staccato**  
1587 INK 0:FILL 1  
1588 IF n%<13:CIRCLE na%+1,nu%+2.8,.6:ELSE CIRCLE na%+.3,nu%-2.8,.6  
1589 FILL 0:INK 7  
1590 **END DEFine**



1592 **DEFine PROCEDURE Tenuto**  
1593 IF n%<13:LINE na%,nu%+4:ELSE LINE na%,nu%-3  
1594 INK 0:FILL 1:LINE\_R TO 2,0 TO 0,-.5 TO -2,0 TO 0,.5:FILL 0:INK 7  
1595 **END DEFine**



```

1597 DEFine PROCEDURE GClef
1598 INK 0:LINE 16,58:ARC_R TO 1,3,-PI
1599 ARC_R TO 0,-5,-PI TO -3,6,-3*PI/4:LINE_R TO 5,7:ARC_R TO -2,0,PI
1600 LINE_R TO 0,-18:FILL 1:CIRCLE_R -1,0,.8:FILL 0:INK 7
1601 END DEFine

```



```

1603 DEFine PROCEDURE FClef
1604 INK 0:FILL 1:CIRCLE 24,25,1.2:FILL 0
1605 ARC 23,26 TO 27.5,26,-PI:LINE_R TO 0,-2:ARC_R TO -5.5,-6,-PI/4
1606 FILL 1:CIRCLE 29,26,.6:FILL 0:FILL 1:CIRCLE 29,23,.6:FILL 0
1607 END DEFine

```



```

1609 DEFine PROCEDURE Metronome(x,y)
1610 LINE x,y TO x+1,y TO x+3,y-6 TO x-2,y-6 TO x,y:LINE x+.5,y-4 TO x+2.5,y+1
1611 END DEFine

```



```

1613 DEFine PROCEDURE TimeSig(x,y)
1614 LINE x,y-5 TO x,y TO x+6,y TO x+6,y-6:ARC TO x+4,y-6,-PI/2
1615 LINE TO x-1,y-6:ARC TO x,y-7,PI/4:LINE TO x+5,y-7
1616 CIRCLE x+2,y-4,.8,-PI/3:LINE x+2.5,y-4 TO x+2.5,y-1
1617 CIRCLE x+4,y-4,.8,-PI/3:LINE x+4.5,y-4 TO x+4.5,y-1
1618 END DEFine

```

```

1620 DEFine PROCEDURE Tempo :REMark Set Beat & Metronome
1621 CURSOR 68,64:PRINT ' ← → :CURSOR 184,64:PRINT ' ↑ ↓ ← → :BLOCK 2,4,202,66,7

```

```

1622 REPEAT Tip

```

```

1623 TPmt bn%:k=CODE(INKEY$(-1))

```

```

1624 SElect ON k

```

```

1625 =192:IF bn%> 1:bn%=bn% -1 :TPrt

```

```

1626 =200:IF bn%< 7:bn%=bn%+1 :TPrt

```

```

1627 =208:IF mn%<240:mn%=mn%+10 :TPrt

```

```

1628 =216:IF mn%> 30:mn%=mn%-10 :TPrt

```

```

1629 = 10:EXIT Tip

```

```

1630 END SElect

```

```

1631 END REPEAT Tip

```

```

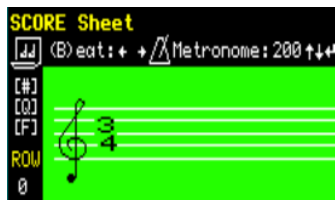
1632 BLOCK 18,10,68,64,0:BLOCK 20,10,184,64,0

```

```

1633 END DEFine

```



```

1635 DEFine PROCEDURE TPrt :REMark Print Tempo

```

```

1636 INK 7:CURSOR 164,64:PRINT FILL$(' ',3-LEN(mn%))&mn%

```

```

1637 IF bn%=1:na%=24:nu%=56:SSpace:RETurn

```

```

1638 CSIZE 2,0:INK 0:STRIP 4:CURSOR 54,97:PRINT TP$(bn%,1)

```

```

1639 CURSOR 56,105:PRINT TP$(bn%,2):CSIZE 0,0:INK 7:STRIP 0

```

```

1640 END DEFine

```



## 1650 REMark File System

### 1652 DEFINE PROCEDURE KLoad

```
1653 INK 5:CURSOR 86,22:PRINT '(S)earch 'QBOLD 1,5,0,80,44,' ← →'
1654 f%=0:SelPath:IF ck%=0:QLSMENU:RETURN:ELSE FCheck
1655 IF ck%=0 OR eck%=1
1656 CURSOR 12,36:PRINT 'File Not Found...':PAUSE 50:ek%=0:QLSMENU:RETURN
1657 END IF
1658 OPEN_IN#99,SFile$:CURSOR 12,36:PRINT 'Loading...'
1659 FOR kn=1 TO 96:FOR kp=0 TO 7:INPUT#99,Bkeys(kn,kp):END FOR kp:END FOR kn
1660 FOR sl=0 TO 9
1661 CURSOR 66+sl*6,36:PRINT '':PAUSE 5
1662 FOR sn=0 TO 23:FOR sp=0 TO 4:INPUT#99,Score(sl,sn,sp):END FOR sp:END FOR sn
1663 END FOR sl
1664 INPUT#99,bn%\mn%:CLOSE#99
1665 sl%=0:sn%=0:sp%=0:ck%=0:KChg 0:PAUSE 50:INK 7:QLSMENU
1666 END DEFINE
```

```
(M)ode (L)oad (S)earch → ←
Select:↓dos7_0BS_ScoreDemo
```

```
(M)ode (L)oad (S)earch → ←
File Not Found...
```

```
(M)ode (L)oad (S)earch → ←
Loading.....
```

### 1668 DEFINE PROCEDURE KSave

```
1669 INK 5:CURSOR 124,22:PRINT ' (E)dit'
1670 SelPath:IF ck%=0:QLSMENU:RETURN:ELSE FCheck
1671 IF eck%=1
1672 CURSOR 12,36:PRINT 'DEVICE ERROR...'
1673 PAUSE 50:ek%=0:QLSMENU:RETURN
1674 END IF
1675 IF ck%=1
1676 CURSOR 12,36:PRINT 'Overwrite Y/N':PAUSE
1677 IF KEYROW(5)<>64ck%=0:QLSMENU:RETURN
1678 END IF
1679 DELETE SFile$:OPEN_NEW#99,SFile$:CURSOR 12,36:PRINT 'Saving... '
1680 FOR kn=1 TO 96:FOR kp=0 TO 7:PRINT#99,Bkeys(kn,kp):END FOR kp
1681 FOR sl=0 TO 9
1682 CURSOR 60+sl*6,36:PRINT '':PAUSE 5
1683 FOR sn=0 TO 23:FOR sp=0 TO 4:PRINT#99,Score(sl,sn,sp):END FOR sp:END FOR sn
1684 END FOR sl
1685 PRINT#99,bn%\mn%:CLOSE#99
1686 sl%=0:sn%=0:sp%=0:ck%=0:PAUSE 50:QLSMENU
1687 END DEFINE
```

```
(M)ode (L)oad (S)ave → ← (E)dit
Select:↓dos7_0BS_ScoreDemo
```

```
(M)ode (L)oad (S)ave → ← (E)dit
DEVICE ERROR...
```

```
(M)ode (L)oad (S)ave → ← (E)dit
Overwrite Y/N
```

```
(M)ode (L)oad (S)ave → ← (E)dit
Saving.....
```

### 1689 DEFINE PROCEDURE QLSMENU

```
1690 CURSOR 2,22:PRINT '(M)ode (L)oad (S)ave (E)xit 'BLOCK 208,18,0,34,0
1691 END DEFINE
```

### 1693 DEFINE PROCEDURE FCheck

:REMark Checks access of File

```
1694 BLOCK 208,18,0,34,0:CURSOR 12,36:PRINT 'Searching...':PAUSE 30
1695 FList:OPEN_IN#99,Drv$(dn%)&'FList'
1696 REPEAT dir_lp
1697 IF EOF(#99):CLOSE#99:BLOCK 204,10,0,36,0:ck%=0:EXIT dir_lp
1698 INPUT#99,RFn$ :IF RFn$==GFN$:CLOSE#99:ck%=1:EXIT dir_lp
1699 END REPEAT dir_lp
1700 END DEFINE
```

```
(M)ode (L)oad (S)earch → ←
Searching...
```

### 1702 DEFINE PROCEDURE FList

:REMark Create File List

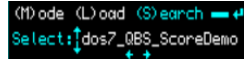
```
1703 DELETE Drv$(dn%)&'FList':OPEN_NEW#9,Drv$(dn%)&'FList':DIR#9,Drv$(dn%):CLOSE#9
1704 END DEFINE
```



```

1706 DEfINE PROCEDURE SelPath :REMark Select Device_FileName
1707 INK 5 :CURSOR 2,36:PRINT 'Select:':CURSOR 154,22:PRINT' ←'
1708 BLOCK 12,3,140,26,5:BLOCK 2,4,160,24,5
1709 QBold 1,5,0,44,32,'↑':QBold 1,5,0,44,40,'↓':INK 7
1710 REPeat FSel
1711 CURSOR 50,36:PRINT Drv$(dn%)&SD$&SFn$(f%),FILL$(' ',16-LEN(SFn$(f%)))
1712 k=CODE(INKEY$(5))
1713 SElect ON k
1714 =192:IF f%> 1 :f%=f%-1 :REMark Change File
1715 =200:IF f%<fm% :f%=f%+1 :REMark Change File
1716 =208:dn%=dn%+1:IF dn%>15:dn%=0 :REMark Chnage Device
1717 =216:dn%=dn% -1:IF dn%< 0:dn%=15 :REMark Change Device
1718 =69,101:Ed_Str :IF lstr%>0:fm%=1 :REMark (E)dit Filename
1719 =83,115:Sfiles :f%=1 :REMark (S)earch Filenames
1720 = 10:IF fm%>0 :ck%=1:EXIT FSel :REMark Action Load/Save
1721 = 32: ck%=0:EXIT FSel :REMark Abort Load/Save
1722 END SElect
1723 END REPeat FSel
1724 SFile$=Drv$(dn%)&SD$&SFn$(f%):GFn$=SD$&SFn$(f%)
1725 END DEfINE

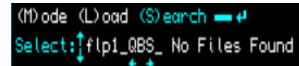
```



```

1727 DEfINE PROCEDURE SFiles :REMark Search For QBS_Files
1728 DIM SFn$(20,30):f%=1:FList:OPEN_IN#9,Drv$(dn%)&'FList'
1729 REPeat dir_lp:
1730 IF EOF(#9) OR f%>20:fm%=f%-1:CLOSE#9:EXIT dir_lp
1731 INPUT#9,QFn$:IF 'QBS_' INSTR QFn$<>0:SFn$(f%)=QFn$(5 TO):f%=f%+1
1732 END REPeat dir_lp
1733 IF fm%<1:CURSOR 110,36:PRINT 'No Files Found':PAUSE 30
1734 END DEfINE

```

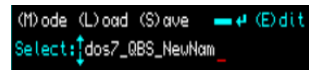


1736 **REMark Filename Editor**

```

1738 DEfINE PROCEDURE Ed_Str :REMark Edit Text String
1739 REPeat str_lp
1740 lstr%=LEN(SFn$(f%)) :BLOCK 6,1,106+6*lstr%,46,2
1741 CURSOR 104,36:PRINT SFn$(f%)&FILL$(' ',15-LEN(SFn$(f%)))
1742 k$=INKEY$(-1):k=CODE(k$):BLOCK 6,1,106+6*lstr%,46,0
1743 SElect ON k
1744 =48 TO 57,65 TO 90,95,97 TO 122:IF lstr%<15:cp%=cp%+1:SFn$(f%)=SFn$(f%)&k$
1745 =192,194,202:IF lstr%>0:lstr%=lstr%-1:SFn$(f%)=SFn$(f%,1 TO lstr%)
1746 = 10:EXIT str_lp
1747 END SElect
1748 END REPeat str_lp
1749 END DEfINE

```



**Note:** Simple Filename Editor Delete character with Left Cursor, Add characters and Enter when complete.

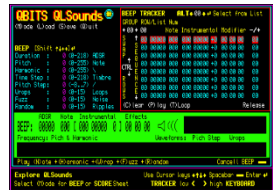
## 1751 REMark Init Setup

### 1753 DEFINE PROCEDURE Init\_win

```

1754 OPEN#6,scr_:WINDOW#6,512,256,gx,gy:PAPER#6,0:BORDER#6,1,3:CLS#6
1755 OPEN#5,scr_:WINDOW#5,232,94,gx+272,gy+36:PAPER#5,0:CSIZE#5,0,0:INK#5,5
1756 OPEN#4,scr_:WINDOW#4,284,20,gx+14,gy+158:PAPER#4,4:CSIZE#4,0,1:INK#4,0
1757 OPEN#3,scr_:WINDOW#3,10,10,10,10
1758 WINDOW#2,504,220,gx+4,gy+4:PAPER#2,0
1759 WINDOW#1,504,220,gx+4,gy+2:PAPER 0:BORDER 1,5
1760 WINDOW#0,512,32,gx,gy+224:PAPER#0,0:BORDER#0,1,3:INK#0,7:CSIZE#0,0,0
1761 CSIZE 2,1:OVER 1
1762 INK 2:FOR i=4 TO 6:CUSOR i,2:PRINT 'QBITS QLSounds'
1763 INK 6:FOR i=6 TO 7:CUSOR i,3:PRINT 'QBITS QLSounds'
1764 CSIZE 2,0:OVER 0:QBold 1,6,1,178,5,'#':CSIZE 0,0
1765 SCALE 120,0,0:INK 5:CIRCLE 76,115,2,8:CIRCLE 76,115,3,4:INK 7:QLSMenu
1766 END DEFINE

```

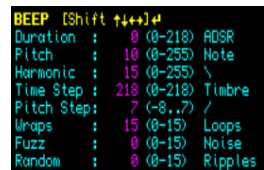


### 1768 DEFINE PROCEDURE MBEEP

```

1769 BLOCK 200,26,0,50,0 :BLOCK 500,143,0,75,0
1770 BLOCK 496,62,2,145,4:INK 7:STRIP 4:INK 0
1771 CURSOR 50,146:PRINT 'ADSR Note Instrumental Effects'
1772 CURSOR 8,175:PRINT 'Frequency: Pitch & Harmonic'
1773 CURSOR 270,175:PRINT 'Wave Forms: Pitch Step Wraps' :STRIP 0
1774 QBold 1,6,1,2,52,'BEEP':QBold 1,6,0,40,52,['Shift ↑↓←→'] ←:BLOCK 2,4,114,54,6
1775 CURSOR 8,208:PRINT 'Play (N)ote +(H)armonic +(W)rap +(F)uzz +(R)andom'
1776 CURSOR 396,208:PRINT 'Cancel BEEP':BLOCK 16,3,476,212,6:INK 5
1777 CURSOR 4, 62:PRINT 'Duration : (0-218) ADSR' :REMark d
1778 CURSOR 4, 72:PRINT 'Pitch : (0-255) Note' :REMark p
1779 CURSOR 4, 82:PRINT 'Harmonic : (0-255) \' :REMark h
1780 CURSOR 4, 92:PRINT 'Time Step : (0-218) Timbre' :REMark t
1781 CURSOR 4,102:PRINT 'Pitch Step : (-8..7) /' :REMark s
1782 CURSOR 4,112:PRINT 'Wraps : (0-15) Loops' :REMark w
1783 CURSOR 4,122:PRINT 'Fuzz : (0-15) Noise' :REMark f
1784 CURSOR 4,132:PRINT 'Random : (0-15) Ripples' :REMark r
1785 INK 0:x=120:y=30:ARC x+5,y+2 TO x+5,y-2,3,PI/2
1786 ARC x+7,y+3 TO x+7,y-3,2,PI/2:ARC x+9,y+4 TO x+9,y-4,PI/2
1787 LINE x+3,y+2.6 TO x,y+1 TO x,y-1 TO x+3,y-2.6 TO x+3,y+2.6
1788 n%=0:BRead:BAtr:BWave:BSet
1789 REMark *** TRACKER ***

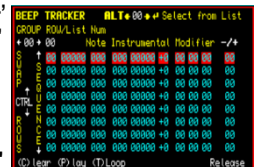
```



```

1790 WINDOW#3,292,142,gx+214,gy+4:PAPER#3,0:BORDER#3,1,2:CLS#3
1791 QBold 3,6,1,0,2,'BEEP TRACKER':QBold 3,6,2,106,2,'ALT ← →'
1792 CURSOR#3,148,2:PRINT#3, '← Select from List':BLOCK#3,2,4,154,4,6
1793 QBold 3,6,0,86,25,'Note Instrumental Modifier':QBold 3,7,1,248,25,'←+'
1794 QBold 3,6,0,1,14,'GROUP ROW/List Num':QBold 3,7,0, 2,24,'← →'
1795 QBold 3,7,0,26,31,'↑':QBold 3,7,0,26,114,'↓': INK#3,6
1796 str$='SEQUENCE':FOR i=0 TO 8:CUSOR#3,26,33+i*9:PRINT#3,str$(i)
1797 str$='SWAP':FOR i=0 TO 4:CUSOR#3,2,26+i*8:PRINT#3,str$(i)
1798 str$='ROWS':FOR i=0 TO 4:CUSOR#3,2,82+i*8:PRINT#3,str$(i)
1799 QBold 3,7,0,12,64,'↑':QBold 3,7,-1,2,74,'CTRL':QBold 3,7,0,12,82,'↓'
1800 CURSOR#3,2,128:PRINT#3,(C)lear (P)lay (T)Loop Release'
1801 OVER#3,0:INK#3,7:rm%=0:rx%=23 tr%=1:TPatn 0:Tinst 0:RowNu 7,py%
1802 END DEFINE

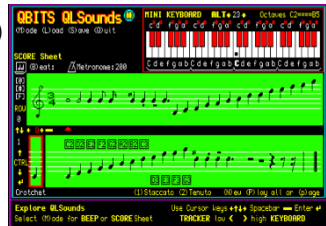
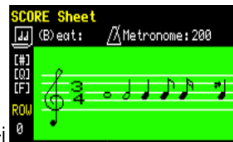
```



```

1804 DEFine PROCEDURE MScore :REmark SCORE Sheet Mode
1805 BLOCK 240,26,0,50,0:BLOCK 500,143,0,75,0 :BLOCK 23,60,25,147,2
1806 BLOCK 476,60,22,76,4:BLOCK 446,60,52,147,4:INK 7:TimeSig 2,85
1807 CURSOR 24,64:PRINT '(B)eat: Metronome: 'Metronome 38,85
1808 FOR i=0 TO 12 STEP 3:LINE 12,56+i TO 200,56+i
1809 FOR i=0 TO 12 STEP 3:LINE 32,16+i TO 200,16+i:LINE 10,16+i TO 19,16+i
1810 QBold 1,6,1,0,52,'SCORE Sheet':QBold 1,7,0,2,80,['#']:
1811 QBold 1,7,0,2,89,['Q'] : QBold 1,7,0,2,98,['Q'] : QBold 1,6,0,2,112,'ROW'
1812 RESTORE 1743:FOR i=1 TO 5:READ x,y,str$:QBold 1,6,1,x,y,str$
1813 DATA 2,136,'↑↓',20,136,'←' '6,162,'↑',6,196,'←',6,184,'↓'
1814 BLOCK 12,3,52,140,6:BLOCK 2,4,14,198,6:BLOCK 498,10,0,208,0
1815 CURSOR 192,208:PRINT '(1)Staccato (2)Tenuto (N)ew (P)lay all or (p)age'
1816 QBold 1,6,0,1,174,'CTRL':GClaf:sn%=0:SCHg 2,0:kn%=23:tr%=0
1817 REMark *** MINI KEYBOARD ***
1818 WINDOW#3,292,74,gx+214,gy+4:PAPER#3,0:BORDER#3,1,2:CLS#3
1819 QBold 3,6,1,0,2,'MINI KEYBOARD': QBold 3,6,2,106,2,'ALT ← →'
1820 CURSOR#3,186,2:PRINT#3,'Octaves C2====B5':kn%=23
1821 FOR i=0 TO 27:BLOCK#3,9,30,4+i*10,26,7
1822 FOR i=0 TO 27
1823 IF i=0 OR i=3 OR i=7 OR i=10 OR i=14 OR i=17 OR i=21 OR i=24:NEXT i
1824 BLOCK#3,9,18,(i*10)-1,26,0:BLOCK#3,7,17,(i*10),26,2
1825 END FOR i
1826 RESTORE 1758:str$='Cdefgab':INK#3,7
1827 FOR i=1 TO 20:READ sx$,sx$:CURSOR#3,sx%-6,14:PRINT#3,sx$:QHash
1828 FOR a=0 TO 3
1829 FOR b=1 TO 7:CURSOR#3,-5+a*70+b*10,58:PRINT#3,str$(b)
1830 END FOR a
1831 DATA 'c',14,'d',25,'f',44,'g',55,'a',66
1832 DATA 'c',85,'d',96,'f',114,'g',125,'a',136
1833 DATA 'c',155,'d',166,'f',184,'g',195,'a',206
1834 DATA 'c',225,'d',236,'f',254,'g',265,'a',275
1835 LINE#3,4,6 TO 8,2 TO 76,2 TO 78,6
1836 LINE#3,78,6 TO 80,2 TO 148,2 TO 150,6
1837 LINE#3,150,6 TO 152,2 TO 220,2 TO 222,6
1838 LINE#3,222,6 TO 224,2 TO 292,2 TO 296,6
1839 QBold 3,7,1,145,58,'C':ka%=146:ku%=50:BLOCK#3,5,4,ka%,ku%,0:RChg 0
1840 END DEFine

```



```

1842 DEFine PROCEDURE QHash
1843 BLOCK#3,5,1,sx%,14,2 :BLOCK#3,5,1,sx%,16,2
1844 BLOCK#3,1,5,sx%+1,13,2:BLOCK#3,1,5,sx%+3,13,2
1845 END DEFine

```

```

1847 DEFine PROCEDURE QBold(ch%,col%,cs%,cx%,cy%,str$)
1848 INK#ch%,col%:OVER#ch%,1:IF cs%<0:cm%=0:ELSE cm%=cs%
1849 FOR a=1 TO LEN(str$)
1850 FOR b=0 TO cm%:CURSOR#ch%,cx%-6+b+a*(6+cs%),cy%:PRINT#ch%,str$(a)
1851 END FOR a:OVER#ch%,0
1852 END DEFine

```

**Note:** Table for own use...

	Duration	Pitch	Harmonic	Time	Step	Wrap	Fuzzy	Random	Remark
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									

### Notes on BEEP Parameters

For Sound construction, **duration** should be considered in steps of 200 milliseconds. For **pitch\_1** & **pitch\_2**, **Fundamental** and **Harmonic**, the following pages identify the QL Sound range of frequencies. Typically make **pitch\_2** a multiple of **pitch\_1** then by increasing the **Time** and/or **Step** (**grad\_x**, **grad\_y**) alters the composite sound output by the number of interim frequencies. **Wrap** creates a changing pattern of a rising or falling scale.

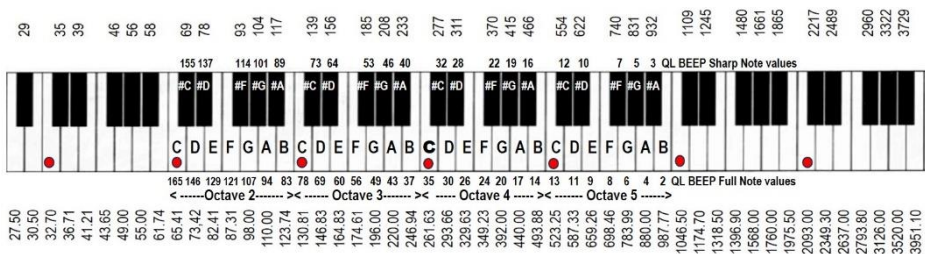
Pitch vs Frequency on the Sinclair QL by Marq

If the QL BEEP highest pitch 0 = 1313Hz and lowest pitch 255 is a frequency of 43Hz. Assuming that the formula is of the form  $a/(x+c)$ , we get approximately the following relationship between frequency (f) and pitch (p):  $f = 11336.256 / (p + 8.634)$  and  $p = 11336.256 / f - 8.634$

Playable Notes and their BEEP value:

F1	251	(deviations have grown to 1 Hz+)	
F#1	236	C4	35
G1	222	C#4	32
G#1	210	D4	30
A1	197	D#4	28
A#1	187	E4	26
B1	175	F4	24
		F#4	22
		G4	20
C2	165	G#4	19
C#2	155	A4	17
D2	146	A#4	16
D#2	137	B4	14
E2	129		
F2	121	(at this point the notes have very little to do with	
F#2	114	the periods, but let's keep going for the sake of	
G2	107	completeness...)	
G#2	101		
A2	94	C5	13
A#2	89	C#5	12
B2	83	D5	11
		D#5	10
(from here; off about 0.5 Hz max.)		E5	9
C3	78	F5	8
C#3	73	F#5	7
D3	69	G5	6
D#3	64	G#5	5
E3	60	A5	4 ?
F3	56	A#5	3 ?
F#3	53	B5	2 ?
G3	49		
G#3	46	C6	2
A3	43		
A#3	40		
B3	37		

QBITS Keyboard Chart



Beep Pitch frequencies supplied by Marq:  
Notes

QBITS highlighted Octave

0	1313.00	46	<b>207.50 G3#</b>	92	112.65
1	1176.71	47	203.77	93	111.54
2	<b>1066.05 C6</b>	48	200.17	<b>94</b>	<b>110.45 A3</b>
3	974.42 4	49	<b>196.69 G3</b>	95	109.39
	897.29	50	193.34	96	108.34
5	<b>831.48 G5#</b>	51	190.10	97	107.32
6	<b>774.66 G5</b>	52	186.96	98	106.31
7	<b>725.11 F5#</b>	53	183.93	99	105.32
8	<b>681.52 F5</b>	54	180.99	<b>100</b>	<b>104.35 G2#</b>
9	<b>642.87 E5</b>	55	178.15	101	103.40
10	<b>608.37 D5#</b>	56	175.39	102	102.47
11	<b>577.38 D5</b>	57	172.72	103	101.55
12	<b>549.40 C5#</b>	58	170.13	104	100.65
13	<b>524.01 C5</b>	59	167.61	105	99.76
14	<b>500.85 B5</b>	60	165.17	106	98.89
15	479.66	61	162.80	<b>107</b>	<b>98.04 G2</b>
16	<b>460.19 A5#</b>	62	160.49	108	97.20
17	<b>442.24 A5</b>	63	158.25	109	96.37
18	425.63	64	156.07	110	95.56
19	<b>410.23 G4#</b>	65	153.95	111	94.76
20	<b>395.90 G4</b>	66	151.89	112	93.97
21	382.54	67	149.88	113	93.20
22	<b>370.06 F4#</b>	68	147.93	114	92.44
23	<b>358.36 F4</b>	69	146.02	115	91.69
24	347.38	70	144.17	116	90.96
25	337.05	71	142.35	117	90.23
26	<b>327.32 E4</b>	72	140.59	118	89.52
27	318.13	73	138.87	119	88.82
28	<b>309.45 D4#</b>	74	137.19	120	88.13
29	301.22	75	135.55	121	87.45
30	<b>293.43 D4</b>	76	133.94	122	86.78
31	286.02	77	132.38	123	86.12
32	<b>278.99 C4#</b>	<b>78</b>	<b>130.85 C3</b>	124	85.47
33	272.28	79	129.36	125	84.83
34	265.90	80	127.90	126	84.20
35	<b>259.80 C4</b>	81	126.47	127	83.58
36	253.98	82	125.08	128	82.97
37	<b>248.42 B4</b>	83	123.71	129	82.37
38	243.09	84	122.38	130	81.77
39	237.99	85	121.07	131	81.19
40	<b>233.09 A4#</b>	86	119.79	132	80.61
41	228.40	87	118.54	133	80.04
42	223.89	<b>88</b>	<b>117.31 A3#</b>	134	79.48
43	<b>219.55 A4</b>	89	116.11	135	78.92
44	215.38	90	114.93	136	78.38
45	211.36	91	113.78	137	77.84

138	77.31	179	60.42	220	49.58
139	76.79	180	60.10	221	49.37
140	76.27	181	59.78	<b>222</b>	<b>49.15 G1</b>
141	75.76	182	59.47	223	48.94
142	75.26	183	59.16	224	48.73
143	74.76	184	58.85	225	48.52
144	74.27	185	58.54	226	48.31
145	73.79	<b>186</b>	<b>58.24 A2#</b>	227	48.11
146	73.31	187	57.95	228	47.91
147	72.84	188	57.65	229	47.70
148	72.37	189	57.36	230	47.50
149	71.92	190	57.07	231	47.31
150	71.46	191	56.79	232	47.11
151	71.01	192	56.50	233	46.92
152	70.57	193	56.22	234	46.72
153	70.14	194	55.94	235	46.53
154	69.70	195	55.67	236	46.34
155	69.28	196	55.40	237	46.15
156	68.86	<b>197</b>	<b>55.13 A2</b>	238	45.96
157	68.44	198	54.86	239	45.78
158	68.03	199	54.60	240	45.59
159	67.63	200	54.34	241	45.41
160	67.22	201	54.08	242	45.23
161	66.83	202	53.82	243	45.05
162	66.44	203	53.57	244	44.87
163	66.05	204	53.31	245	44.70
164	65.67	205	53.06	246	44.52
<b>165</b>	<b>65.29 C2</b>	206	52.82	247	44.35
166	64.91	207	52.57	248	44.17
167	64.54	208	52.33	249	44.00
168	64.18	209	52.09	250	43.83
169	63.82	210	51.85	251	43.66
170	63.46	211	51.61	252	43.49
171	63.11	212	51.38	253	43.33
172	62.76	213	51.15	254	43.16
173	62.41	214	50.92	<b>255</b>	<b>43.00 F1</b>
174	62.07	215	50.69		
175	61.73	216	50.47		
176	61.40	217	50.24		<b>32.70 C1</b>
177	61.07	218	50.02		
178	60.74	219	49.80		

### Listening to Musical Notes

The frequency range of the human ear can be as low as 20 cycles per second or as high as 20,000 cycles per second (20Hz to 20kHz). The higher the frequency, the higher the pitch. Double the frequency and the pitch goes an octave higher. For example, 260Hz is approximately middle C on a piano keyboard 720Hz is C an octave higher, 4186Hz is the highest C8 and A0 the lowest key is 27.5Hz. The AC mains hum of 50Hz in Europe is close to the pitch of G1 = 48.99Hz.

QSound

Command	Function
<b>HOLD</b>	pauses the playing of a sound channel
<b>PLAY</b>	sets a sound channel with a sequence of notes
<b>PLAYING</b>	tests if a sound channel is currently playing
<b>RELEASE</b>	starts, or resumes playing of sound channels
<b>SOUND_AY</b>	clears sound channel queues

Construction of the sound string

Function	Values
Notes	C D E F G A H (H corresponds to B, HB to B flat)
Sharps	#
Flats	b
Rests	p (one length unit)
Change in octave	o0 o1 .. o7 (default o2)
Change in volume	v0 v1 .. v15 V16 switches to envelope control
Duration of note in 1/50 sec	10 .. 1255 (default: 15)
Change of noise frequency	n0 n1 .. 31 (default n0)
Determine warp curve	w0 w1 .. w15 (default w0)
Change length of warp	x0 x1 .. x32767 (default is x0)
Synchronisation stop	s causes a sound channel to wait
Activate a waiting channel	r1 r2 r3

QBITS QLSoundSE – 2024 Change Notes

Changes to **Introduction** and **KEY** instructions for more clarity of use

**TRACKER /SCORE** Note number- **AZERTY keyboard**  **chevrons** unwieldy being up/lower case on same key next to Shift. Changed to use  **ALT Left/Right** cursors.