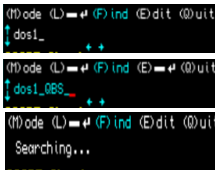
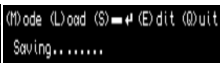
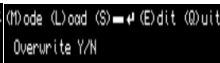
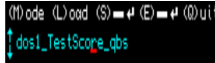


QBITS QLSounds - File Management



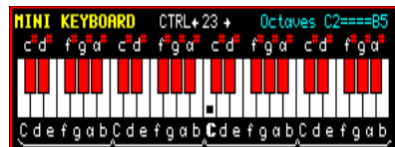
Press 'L' to load a file, Press 'E' to (E)dit **Device/SubDir** Press 'F' to (F)ind **_qbs** Data files. Either 'No Files Found' is displayed or use **Left/Right** Cursors to Select a File. Abort with **Spacebar**, Action with **Enter**. If program recognises **_qbs** File, Loading will commence.

To **Save**, the Loaded or Default Filename is displayed. Press 'E' to (E)dit. **Device/SubDir/Filename**. Delete character [CTRL Left or Right], add new Character(s), Abort **Spacebar**, Action with **Enter**. If not found '**DEVICE ERROR**' is returned. If already exists an **Overwrite Y/N** Prompt is returned.

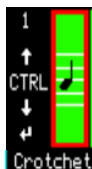


QBITS QLSounds - MINI KEYBOARD

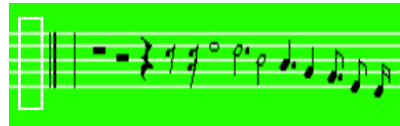
In **SCORE (M)**ode a four Octave Keyboard C2 to B5 is displayed. The Marker ■ shows current Key/Note position, to Play press **Shift+Spacebar** [Spacebar to Cancel]. Use **CTRL ← 23 →** to pan the keys or select Octave **F2 - F5+cdfgab** and use **Shift F2 - F5+cdfga** for #Sharp Keys. Bottom Left of screen shows the Stave Notes position.



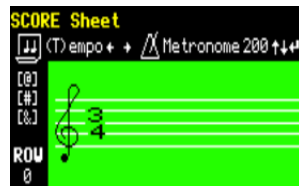
QBITS QLSounds - SCORE Symbols



Use **CTRL ↑ ↓** to Select a Score Symbol (Space through to SemiQuaver). Press **Enter** and Symbol is written to the next Stave position. Where 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 horizontal bar placed above or below the Note head. These are effects that shorten (blend) or extend (detach) the Time between Notes.



QBITS QLSounds - Tempo

Press 'T' and use **← →** Left /Right Cursors to change Rhythm. This ranges from none (Space) then 2/2 through to 6/8. Change Timing with **↑ ↓** Up/Down Cursor Keys. The range is from 30 to 240 Beats per Minute.



Press '@' to Reset default Note Pitches.

Set Notes to Default Y/N



Press '#' to action with **QSound PLAY** command.

Set Notes with QSound Y/N



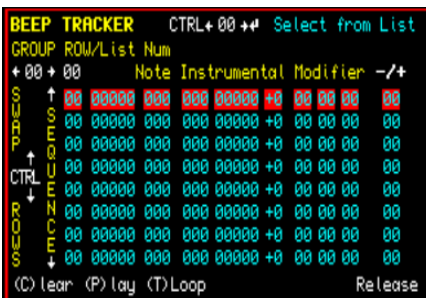
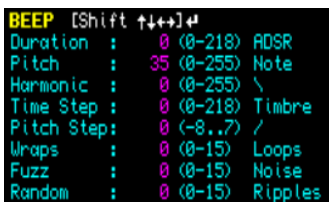
Press '&' displays an **FClef** in bottom Stave Row for where it might be used. [Remove **FClef** by swapping to **BEEP** Mode and back to **SCORE** Mode]



Note: QBITS QLSounds Prog includes a **Play** command for [QPC2] QSound AY-3 system that produces preset Notes for the Four Octave Keyboard.

QBITS QLSounds - BEEP TRACKER

In **BEEP (M)**ode Select an entry, the list holds (1-96) Select with **CTRL ←00→ Left/Right** Cursors 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.



The Tracker helps arrangement of instrumental sounds varying from the musical to just quirky, such as whispery atmospherics to explosions and wailing sirens. 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.

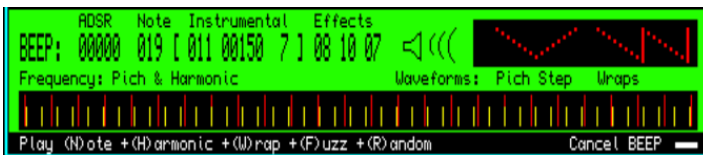
Swap Rows with **CTRL↑↓ Up/Down** Cursors to re-sequence the entries. To Delay the playing of the following Row Set PAUSE **Release** value with **-/+** [00-15]. **(C)**lear - Resets highlighted Row.

Once a number of entries have been set use **(P)** playback all or a selected number of rows set with **(T)** Loop for continuous Playback. To Cancel Playback at any time press **Spacebar**.

QBITS QLSounds - BEEP Attributes

Description labels are to help identify the workings. ADSR is duration of a generated sound, the Attack, Decay, Sustain and Release. A Note has a Primary Pitch, a second Pitch (Harmonic) combined with Time Step and Pitch Step or angle. This creates the Timbre, the characteristic tonal quality that differentiates that of say a guitar from a piano.

Wraps added to the Time Step, create scaling and looping effects such as Wailing or that of a Siren. Fuzz noisiness and Random add further to shape the waveform.

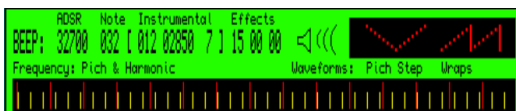


The Graphics are an attempt to show how the Attributes effect the output. **Pitch** and **Harmonics** frequencies are displayed as vertical lines. The first Waveform is Pitch Time/Step created by the **grad_X**, **grad_y** Attributes. The second Waveform is the assumed effects of the **Wrap** Attribute. 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.

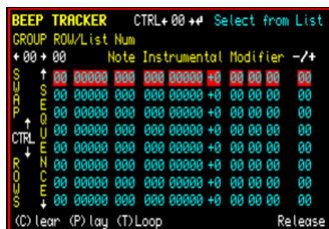
The Pitch and Harmonics vertical lines become wider apart as the frequency of the Pitch lowers. The Pitch Step and Wraps Wave shows Scale and Direction. Play back the various **BEEP** Attributes using 'N' Note, 'H' Harmonic, 'W' Wraps, 'F' Fussiness" 'R' Random (UPPER or lower case).

QBITS QLSounds – Development

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.

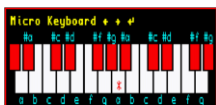


The BEEP Graphics help explain the actions made by the Attributes. The TRACKER replaces an earlier Keyboard. Its function to provide a way to sequence instrumental compositions for **(P)**layback with continuous loops that can be varied with **(T)**Loop.



QBITS Micro Keyboard

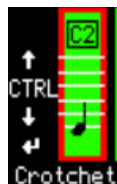
Reviewing early SuperBasic Magazine Progs aimed at using the **QL BEEP** command I hadn't come across any significant graphical representation of a keyboard.



The first QBITS display was two octaves based on a QJUB article which is described later. This is now a four Octave MINI Keyboard.

**QBITS SCORE Sheet**

For Tempo the Beat and Metronome values 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 as 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. as shown on previous page are selected with CTRL and Up/Down Cursors and displayed 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 **Notes** **na%**, **nu%**. For keyboard location on the **Stave** the offset **so%** is added to **nu%**. For the display **ds%** identifies a Symbol Note or Rest etc.

QBITS QLSounds - Program Performance

The Program will require Extended Memory and TK2. Running 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 QLSounds - Compatibility

The Coding was written and tested with the **QPC2** Emulator with **SMSQ/E** which appears to be more tolerant to unassigned variables setting them to '0' instead of '★', which incurs an error that freezes the program in other operating systems.

When LRUN on other QL Platforms a number of problems were identified. For example, Platforms running earlier ROM versions, unacceptable **FOR** loop integers ie a%=1 to 4 or **SELECT ON** a%. Multiple embedded **FOR** loops without **END FOR**'s and as with **END REPEAT** statements missing name then idiosyncrasies such as **CURSOR** graphics coordinates only working with **WINDOWS** channel #1... To improve compatibility of the **QLSounds** Prog across more QL Platforms, where possible code changes have been made to overcome these identified problems.

QBITS QLSounds -Timing

Keyword **PAUSE** and Simon Goodwin's **T_ON**, **T_START**, **T_COUNT**, **T_STOP**, **T_OFF** are linked to the AC supply of 50cycles EU and 60cycles US and were not precise enough for Musical Notes. A somewhat different approach was required, the **DATE** Timer method uses the QL Clock to determine a QL Platforms Timing Cycle per second by utilising a counter that can set a more accurate 20msec delay and potentially shorter Time Delays. This method also provides a level of compatibility across the various QL platforms.

Using Code posted on the QLFORUM Web Site by Steve Poole, we have a Function to determine a QL Platforms cycle rate in msec and a Procedure to execute a calculable Time Delay for *duration* and *delay* between Notes.

QBITS Time Delay derived from **Steve Poole's** – QL Clock **DATE** Timer (part of **fastStave6_bas**)

Embedded code for **QL TIMER** set as part of Program initialisation

```
2000 sd=DATE:REPEAT Chk_Ip:IF sd<>DATE:sd=DATE:EXIT Chk_Ip
```

```
2001 FOR tc=1 TO 1E15:IF sd<>DATE:EXIT tc:END FOR tc      Note: tc tick count
```

```
2002 ms=tc/50      :REMark Set to 20ms
```

```
1000 DEFINE PROCEDURE TDel(wait)
```

Note: 'wait' delay multiplier

```
1001 sd=DATE:FOR td=1 TO ms*wait:IF sd<>DATE:END IF
```

Note: 'sd' start date : td time delay

```
1002 END DEFINE
```


QBITS QLSounds - Playback

The QL Clock **DATE** Timer provided a more accurate Time Delay but following some suggestions by Mark Swift a potential bug was identified to Notes in Playback – this was solved by changes to the **nv** variable and Note Timing values in **SEntry** : **DSymbol** : **PScore**

Note: BEEP 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.

TRACKER /SCORE - Numbering

Euro Keyboard and Tablet problems identified by Andrew. The **AZERTY keyboard**  **chevrons** found to be unwieldy being up/lower case on same key next to Shift. First **ALT Left/Right** cursors was trialled but this created complication with Tablets that used the ALT Cursors for rotation between Portrait /Landscape. The fix was to use **CTRL** with **Left/Right Cursors** and this actually fitted in with the use of CTRL Up/Down Cursors for actions already used in the TRACKER and SCORE Modes.

QBITS QLSounds - DATA Files

A QL SuperBASIC Filename can be 36 ASCII characters in length added to this is the device name of five Characters. For QBITS QLSounds Data files the setup is 5 characters for device followed by up to 28-Alphanumeric characters for Filename.

For example: win1_QLSDS_ScoreDemo_qbs

The last four, for now use the suffix '**_qbs**', but the String can be broken down further by using some characters for a SubDIRectory. As for the future this may change and use of **QSounds** may require a different ending for their Data files. The AY-3-8910 implementation with its interesting past history, will no doubt open the door to more musical innovations.

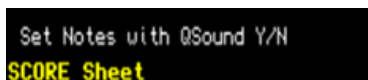
Note: www.QLForum.co.uk

Thanks to various Members who have helped in checking out the QBITS Progs and their offered advice in ways to improve coding. For **QBITS QLSounds** a Special thanks to Members Andrew, Mark and Steve for the QL Clock DATE TIMER.

QBITS_QLSounds – QSound

The **QPC2** emulator now included **QSound**, although the AY-3-8910 implementation offers the potential to extend **QLSounds**, the **QSound PLAY** command is the only one of which short term I have worked on plus use of **SOUND_AY**. Further investigations will continue with new options considered for future Reviews. For now, a Score can be Played back with **QSound** by pressing the '#' key and confirming the action.

The Note ID's are held in an array QNotes(qn)
[qn == C,#C,D,#D,E,F,#F,G, #G, A, #A, H]



The Octave and Note are identified and Read by the QLSounds PROCEDURE **PScore**.

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 software development will utilise both chip outputs simultaneously and extend the range of tonal qualities to accommodate various musical Instruments played back through associated output systems.

QPC2 - QSound Background

An implementation of **ABC-Electronic's QSound** has been added to QPC2 Emulator. Back in the Day the AY-3-8910 sound generator chips had a history of use with a number of Arcade machines and 8bit computers of the 1980's.

This programmable 3 voice generator 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.

Command	Function
---------	----------

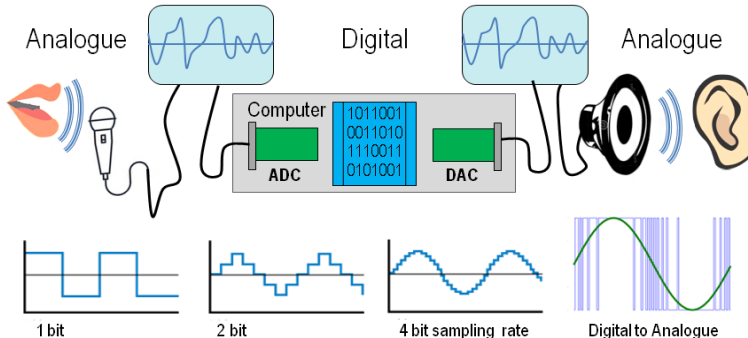
HOLD	pauses the playing of a sound channel
PLAY	sets a sound channel with a sequence of notes
PLAYING	tests if a sound channel is currently playing
RELEASE	starts, or resumes playing of sound channels
SOUND_AY	clears sound channel queues

Construction of the sound string

Function	Variables
NOTES	CDEGFAH [H corresponds to B,HB to B flat]
Sharps	# [C#]
Flats	b [Cb]
RESTS	p [1 unit length]
Octave	o0 ... o7 [Note range C0 to H7 default o2]
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 - Note Max is only 4 digits not 5 - x32767]
Stop	s [causes a sound channel to wait]
Activate Ch	r1 r2 r3 [Activate a waiting Channel]

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.

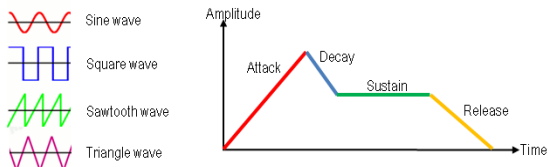


The minimum sample rate of any signal without introducing errors is twice the highest frequency present in the signal and is called the Nyquist frequency or Limit. A Digital representation can be expressed as a number of on/off's, highs and lows or in binary logic of 1's and 0's.

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.



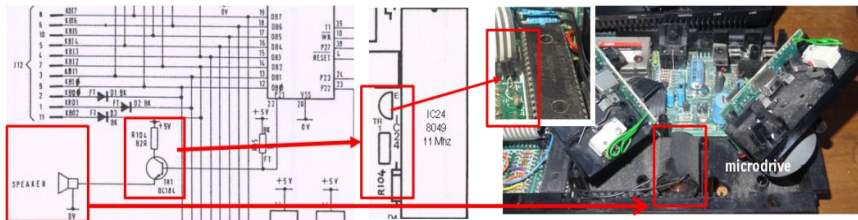
Summarising

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.

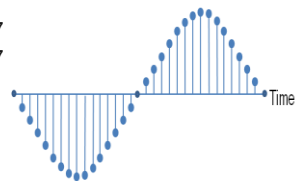


In executing a **SuperBASIC BEEP** instruction, the IPC port 2 P21 switches on/off transistor TR1 with the changing number of 1s and 0s in each output cycle. The emitter follower configuration is used as a voltage buffer amplifier to drive the 600hm 23mm loudspeaker situated between the two microdrives. 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

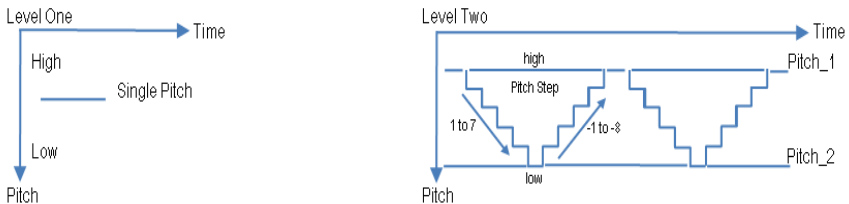


Frequency and Amplitude: Two key features of a sound wave are the frequency (how many times the wave vibrates in one second) and is broadly related to the Pitch of the sound we hear. The amplitude (Volume) of a sound is related to the amount of energy that the sound wave carries. As the frequency increase 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.

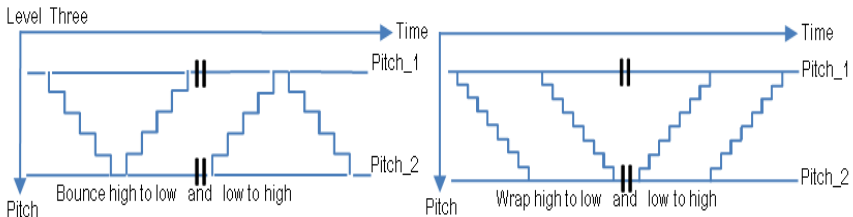
QL Sound Concepts

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).

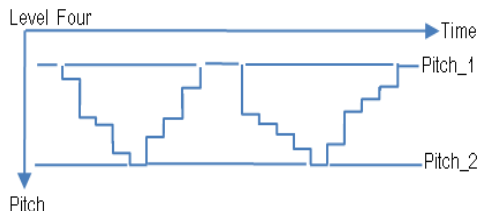


At level two a second pitch is added and the rate at which the sound ramps between the two pitches allegedly can produce semi musical beeps, spiralling or rippling tones, growls, zaps and moans. The number of steps and direction high to low or low to high can be configured.



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 made modifications to the **BEEP Exerciser** 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 - high / 8 to 15 high - 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 ip

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 ip

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 is 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:

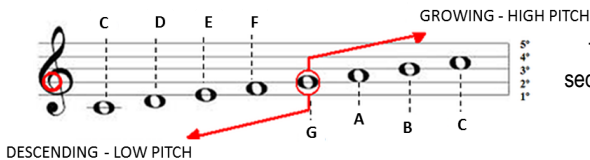
C → do
D → re
E → mi
F → fa
G → so
Separator
A → la
B → ti (H in German)



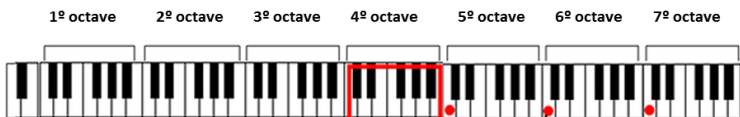
Sheet music registers the harmonic, rhythmic and melodic ideas. The Notes are positioned and written in the form of musical symbols.

Music Stave

The five lines (1st, 2nd, 3rd, 4th and 5th) 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

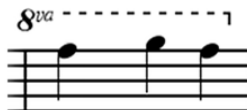
Where the Stave can't handle the representation of the notes for a full range of octaves, ledger lines are used. These lines are nothing more than the continuation of the Stave, they are used to represent notes that surpass the bottom or upper limits.

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.

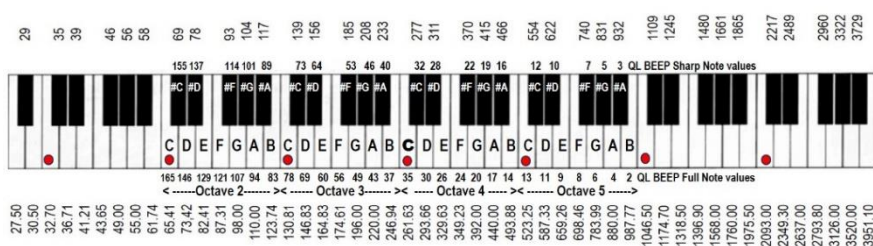


Ties connect notes of the same pitch, forming essentially one longer note.

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!

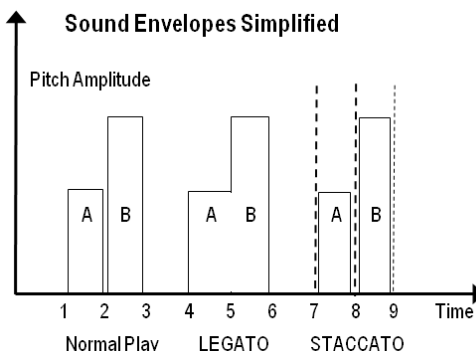
The **QLUB** edition of Mar/April 1985 carried an article with a short program which displayed to screen a **Stave**, a **G-Clef** and added **crotchet** symbols to selected pitches. It described musical notes over two Octaves that could be reproduced and gave the **BEEP pitch_1** numbered equivalent.

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

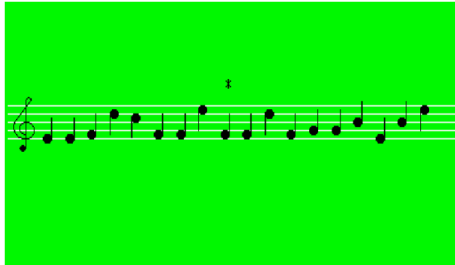
```



```

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**. Music has a regular pulse or rhythm identified as the **beat**. The number of beats in each uniformed section or measure is written after the **Clef** at the beginning of a score. 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 beats 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 to determine its duration.

For practical use with the **QL Sound System** and to provide a distinct separation between **Notes** or **Rests** from any previously played, a **delay** 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 a Time Delay (20ms) 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 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 multiplier of 3 is only 60ms and not quite enough time for the human ear to differentiate the change in pitch.

QBITS Time Delay

For a **Note's** playback's **ADSR** in this arrangement the **Attack**, **Decay**, **Sustain** is covered by the **BEEP** command with its set parameters and Time Delay for its '**duration**' followed by **BEEP** with a second '**delay**' to sustain a **Release** before any next Note is played.

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** shown as a **Dot** after the Note. For modes of play such as **Staccato** or **Tenuto/Legato** with **Augmentations** 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*::**TDel** *duration* : **BEEP** :**TDel** *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 **duration + delay**. The **Space**, **Separation Bar** and **End Bar** are given a zero-time delay.

QBITS Notes and Scores

The **QBITS Notes** chosen range 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, labeled 'NOTES', shows a staff with various note types: 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, labeled 'RESTS', shows a staff with various rest types: Staccato, Normal, Tenuto, Sharp, SemiBreve(4), Minim(2), Crotchet(1), Quaver(0.5), and SemiQuaver(0.25). A legend on the left shows 'FClef' and 'Space'.

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 Exploring QL Sounds

QBSBeepv1
QLUBMicrov1
QBITS_QLSounds_bas

BEEP Exerciser...
QLUB Magazine: Music Micro Please
QBITS QLSounds Main Prog...

TrackDemo_qbs
ScoreDemo_qbs

A Demo Data File for **TRACKER** Mode!
A Demo Data File for **SCORE** Sheet Mode!

Notes:

QBITS PROCedures

Init_win : Init_keys	Init Windows Includes TIME: Load Default KeyBoard Pitches
QLSMenu : QLTimer : QLSHelp	Mode/File Menu, Set Time Delay, Key Instructions
QBITS_QLSounds	Main Menu
TDel : QExit	Time Delay Function Exit Program
MBeep	BEEP Attributes and TRACKER Screen
MScore	SCORE Sheet and MINI KEYBOARD Screen
QBold(ch%,col%,cs%,cx%,xy%,str\$) & Bold Character	
BSelect(chg%) : BPrnt : BSet : BRead : Batt :BWrite(Chg%) : BWave BEEP Attributes	
TPatn(chg%) : TInst(chg%) :	TEntry Select Tracker Pattern / Instrumental : Enter in Table
TDdecay(chg%) : TSwap(chg%)	Set a Delay between rows : Swap rows Up/Down
RowNu(scol,py%,pr%) : RowChg(chg%) : RowUp : RowDn : TClear	Change Tracker Row
TLloop	Select two or more Rows to play in continue loop
PTrack(pt%,rm%,rx%)	Play selected rows
SNew	Clear SCORE & TRACKER for New Entries
Note_Reset	Reset to Default BEEP Note Sounds
Set_QLSound	Play Score with QSound Note Sounds
SEntry	Write Selected Note to next position on Score Sheet
KChg(chg%) : NChg(chg%) : SChg(mp%,chg%) : RChg(chg%)	Change Key /Note /Stave /Row
PScore : QNotes(qn)	Play Score BEEP Notes or QSound
TDel (wait	Time Delay - wait multiplier: Set 20ms Delay Timer for QL Platform
DSymbol	Displays selected Symbols
SSpace	Clears Stave position
Ledger	Add Ledger lines when required to extend Stave
EBar	SBar Draws End Bar : Separation bar
Head, Stem, Hook1, Hook2	Draws the Parts that make up a Note Symbol
Semibreve	SBRest Draws Note Rest Symbol Beat value of 4
Minim	MRest Draws Note Rest Symbol Beat value of 2
Crochet	CRest Draws Note Rest Symbol Beat value of 1
Quaver	QRest Draws Note Rest Symbol Beat value of ½
Semiquaver	SQRest Draws Note Rest Symbol Beat value of ¼
Sharp	Pitch raised by ½ pitch
Dot	Beat played 1½ times normal length (Beat values 3, 1½, ¾)
Staccato	Notes played Delay = 8/10 of Beat (distinct)
Tenuto	Notes played Delay = Whole Beat (lengthy)
GClef	FClef Draws the GClef : FClef
Tempo	TPrnt Select Beat & Metronome values :Prints Tempo Selection
QHash	Sharp for Keyboard Notes
KLoad : KSave	Load Save Filename into/from Prog Data Array Fields
SeIPath:FList:SFiles:FCheck	Select device & Data file Check QBS_ against DIR file List
Ed_Str	Filename Editor Delete / Add Character

QBITS QLSounds – Code

1000 REMark **QBITS_QLSounds_bas** [QBITS QLSounds 2024 QL40th - QPC2] vA40

1002 dev\$='win1_':MODE 4:gx=0:gy=0 :REMark Basic Settings

1004 **WHEN ERROR**:eck=1:**CONTINUE**:**END WHEN**

1006 REMark **Import QBITConfig Settings - QPC2**

1007 DIM Drv\$(15,5):OPEN_IN#9,dev\$&'QBITConfig':

1008 INPUT#9,gx\gy\dn\$\dev\$\dn%\dm%:FOR d=0 TO 15 :INPUT#9,Drv\$(d):END FOR d:CLOSE#9

1010 REMark **Set Arrays**

1011 DIM Bkeys(96,7),Score(9,23,4),SFn\$(20,15),DFn\$(32),str\$(30)

1012 REMark **Mkey(n%) - Stave offsets +16.5 to -3**

1013 DIM MKey(26):**RESTORE 1015**:FOR a=0 TO 26:**READ MKey(a)**

1014 DATA 16.5, 15, 15, 13.5, 13.5, 12, 12, 10.5, 9, 9, 7.5, 7.5, 6

1015 DATA 4.5, 4.5, 3, 3, 1.5, 1.5, 0, -1.5, -1.5, -3, -3, -4.5, -6, -6

1016 REMark **Octave Keys Checks**

1017 DIM OChk\$(21,2):**RESTORE 1019**:FOR a=21 TO 1 STEP -1:**READ OChk\$(a)**

1018 DATA 'C2','C2','D2','D2','E2','F2','F2','G2','G2','A2','A2','B2'

1019 DATA 'C3','C3','D3','D3','E3','F3','F3','G3','G3'

1020 REMark **Beat Tempo Signatures**

1021 DIM TP\$(7,2,1):**RESTORE 1023**:FOR i=2 TO 7:**READ TP\$(i,1),TP\$(i,2)**

1022 DATA '2','2','2','4','4','4','3','4','3','8','6','8'

1024 **DEFine PROCEDURE Init_keys**

1025 **RESTORE 1030** **Note:** Bkey - kn%(0-96) kp%(0-8) d,p,h,t,s,w,f,r

1026 FOR a=1 TO 48

1027 **READ n:** Bkeys(a,1)=n:Bkeys(a,2)=0:Bkeys(a,3)=0:Bkeys(a,4)=0

1028 Bkeys(a,5)=0:Bkeys(a,6)=0:Bkeys(a,7)=0

1029 END FOR a

1030 DATA 2,3,4,5,6,7,8,9,10,11,12,13,14 :REMark B5 to C5

1031 DATA 16,17,19,20,22,24,26,27,28,30,32,35 :REMark B4 to C4

1032 DATA 37,40,43,46,49,53,56,60,64,69,73,78 :REMark B3 to C3

1033 DATA 83,89,94,101,107,114,121,129,137,146,155,165 :REMark B2 to C2

1034 **END DEFine**

1036 REMark **Set Variables**

1037 pr%=1:py%=0:r%=0 :REMark Tracker check/Pattern Row/Screen Row/Release

1038 kn%=23:kp%=0:ds%=12:as%=0:ar%=0 :REMark Notes/Parameter/Symbol/Articulates

1039 bn%=1:mn%=200:m%=0:mp%=2 :REMark Beat/Metrenome/Marker/Row Pointer

1040 sl%=0:sn%=0:sp%=0:chg%=0:ns%=0 :REMark Score Line/Number/Parameter etc.

1041 n%=1:eck%=0:ck%=0 :REMark MKey Num : File num :Checks

1042 f%=1:fm%=0:DFN\$=dev\$:SFn\$(f%)='Demo' :REMark QBITS Sound Filename & Appendix

1044 **Init_win**:m%=0:**QBMode**:**QBITS_QLSounds**

Note: See **Init_win** for setup of QLPlatform **TIMER** code.

```

1046 DEFINE PROCEDURE TDel(wait)          :REMark Time Delay – multiples of 20msec
1047 sd=DATE:FOR td=1 TO ms*wait:IF sd<>DATE:END IF :END FOR td
1048 END DEFINE

1050 DEFINE PROCEDURE QBITS_QLSounds
1051 REPEAT Mlp
1052 k=CODE(INKEY$(-1))
1053 SELECT ON k
1054 =77,109 :IF m%=0:m%=1:ELSE m%=0: END IF:QBSMode      :REMark (M)ode
1055 =76,108 :KLoad:IF m%=1:RChg 0 :ELSE TPatn 0          :REMark (L)oad
1056 =83,115 :IF Score(0,0,0)>0:KSave                     :REMark (S)ave
1057 =81,113 :QExit :BLOCK 18,10,172,22,0               :REMark (Q)uit
1058 =39, 64 :IF m%=1:Note_Reset :QLSMenu                :REMark [Q] Note Reset
1059 =70,102 :IF m%=1:FClef                               :REMark [F] Clef
1060 =35      :IF m%=1:Set_QSound :QLSMenu                :REMark [#] QPLAY
1061 =236,240,244,248:IF m%=1 :KOct                     :REMark F2 - F5
1062 =80,112 :IF m%=1:PScore :ELSE PTrack 0,rm%,rx%     :REMark (P)lay/(p)lay
1063 =84,116 :IF m%=1:Tempo :ELSE TLoop                 :REMark (T)empo/[T]Lp
1064 =192     :IF m%=1:Schg mp%,-1 :ELSE TPatn -1        :REMark Left SCol/Patn
1065 =200     :IF m%=1:Schg mp%,+1 :ELSE TPatn +1         :REMark Right SCol/Patn
1066 =208     :IF m%=1:RChg -1 :ELSE RowChg -1           :REMark Up SRow/PRow
1067 =216     :IF m%=1:RChg +1 :ELSE RowChg +1           :REMark Down SRow/PRow
1068 =210     :IF m%=1 AND ds%> 0 :NChg -1:ELSE TSwap -1 :REMark Ctrl Up Note/Row
1069 =218     :IF m%=1 AND ds%<15 :NChg +1:ELSE TSwap +1 :REMark Ctrl Dn Note/Row
1070 =49      :IF ar%=0:ar%=1 :ELSE ar%=0:END IF :NChg 0 :REMark (1)Staccato
1071 =50      :IF ar%=0:ar%=2 :ELSE ar%=0:END IF :NChg 0 :REMark (2)Tenuto
1072 =194     :IF m%=1 AND kn%<47:KChg kn%+1:ELSE TInst -1 :REMark CTRL ← Kbd/Trk
1073 =202     :IF m%=1 AND kn%> 0:KChg kn% -1:ELSE TInst+1 :REMark CTRL → Kbd/Trk
1074 =9        :IF m%=1 AND mp%=2:mp%=-2:ELSE mp%=2:END IF :Schg mp%,0 :REMark TAB
1075 =252     :IF m%=1:BEEP 0,p,h,t,s,w,f,r             :REMark Shift Space
1076 =32      :BEEP                                       :REMark Space
1077 =10      :IF m%=1:SEntry :ELSE BSet:TEntry         :REMark Score/Tracker
1078 =196     :IF m%=0:BWrite -1                          :REMark Shift Left BEEP
1079 =204     :IF m%=0:BWrite +1                          :REMark Shift Right BEEP
1080 =212     :IF m%=0:BSelect -1                         :REMark Shift Up BEEP
1081 =220     :IF m%=0:BSelect +1                        :REMark Shift Down BEEP
1082 =45, 95 :IF m%=0:TDecay -1                         :REMark - Delay
1083 =43, 61 :IF m%=0:TDecay +1                          :REMark + Delay
1084 =67, 99 :IF m%=0:TClear                             :REMark (C)lear TkEntry
1085 =78,110 :IF m%=0:BEEP 0,p :ELSE SNew:l=1           :REMark (N)ote/(N)ew
1086 =72,104 :IF m%=0:BEEP d,p,h,t,s,0,0,0             :REMark (H)armonic
1087 =87,119 :IF m%=0:BEEP d,p,h,t,s,w,0,0             :REMark (W)raps
1088 =70,102 :IF m%=0:BEEP d,p,h,t,s,w,f,0             :REMark (F)uzz
1089 =82,114 :IF m%=0:BEEP d,p,h,t,s,w,f,r             :REMark (R)andom
1090 END SELECT
1091 END REPEAT Mlp
1092 END DEFINE

1094 DEFINE PROCEDURE QExit
1095 INK 7:CURSOR 172,22:PRINT 'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$:STOP:ELSE RETURN
1096 END DEFINE

```

1100 REMark BEEP Attributes

```

1102 DEFINE PROCEDURE BSelect(chg%) :REMark Select BEEP Attribute
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 Print BEEP Attribute
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 Attributer change
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 Attribute
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 BWrite(chg%) :REMark Write BEEP Attribute
1120 Bkeys(n%,kp%)=Bkeys(n%,kp%)+chg%:BRead
1121 IF d< 0:Bkeys(n%,0)=218:ELSE IF d>218:Bkeys(n%,0)= 0
1122 IF p< 0:Bkeys(n%,1)=255:ELSE IF p>255:Bkeys(n%,1)= 0
1123 IF h< 0:Bkeys(n%,2)=255:ELSE IF h>255:Bkeys(n%,2)= 0
1124 IF t< 0:Bkeys(n%,3)=218:ELSE IF t>218:Bkeys(n%,3)= 0
1125 IF s<-8:Bkeys(n%,4)= -8:ELSE IF s > 7:Bkeys(n%,4)= 7
1126 IF w<0:Bkeys(n%,5)= 0:ELSE IF w> 15:Bkeys(n%,5)=15
1127 IF f< 0:Bkeys(n%,6)= 0:ELSE IF f > 15:Bkeys(n%,6)=15
1128 IF r< 0:Bkeys(n%,7)= 0:ELSE IF r > 15:Bkeys(n%,7)=15
1129 INK 7 :BPmt
1130 END DEFINE

```

```

BEEP [Shift ↑↓↓↑]↑
Duration : 0 (0-218) ADSR
Pitch : 19 (0-255) Note
Harmonic : 11 (0-255) \
Time Step : 1 (0-218) Timbre
Pitch Step: 7 (-8..7) /
Wraps : 7 (0-15) Loops
Fuzz : 5 (0-15) Noise
Random : 7 (0-15) Ripples

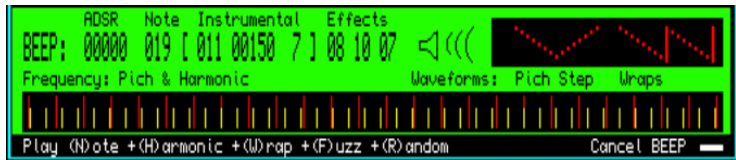
```

Note: Use Shift Cursor Keys 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.


```

1132 DEFINE PROCEDURE BAttr                                     :REMark Show BEEP Attributes
1133 BRead:d=d*150:t=t*150:REMark IF tr=1:RowPt py%,Score(sl,sn,0)
1134 PRINT#4,'BEEP: ';FILL$('0',5-LEN(d))&d;' ' ;FILL$('0',3-LEN(p))&p;
1135 PRINT#4,[' ' ;FILL$('0',3-LEN(h))&h;' ' ;FILL$('0',5-LEN(t))&t;' ' ;
1136 PRINT#4, FILL$(' ',2-LEN(s))&s;']' ;FILL$('0',2-LEN(w))&w;' ' ;
1137 PRINT#4, FILL$('0',2-LEN(f))&f;' ' ;FILL$('0',2-LEN(r))&r:CLS#4,4
1138 END DEFINE

```

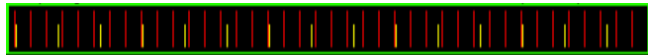


Note: BWave attempts to create a Graphical representation of the BEEP Sound. The Pich and Harmonic are shown as vertical spaced lines. The Pich Step is shown as a Triangular waveform starting with a Rise or Fall. The Wrap shown as a Saw tooth again starting with a Rising or Falling edge.

```

140 DEFine PROCEDURE BWave                                     :REMark Show BEEP Graphics
1411 BLOCK 484,20,8,186,0:BLOCK 160,24,332,150,0
1412 IF p>0:FOR i=0 TO 476 STEP p:BLOCK 1,16,12+i,188,2
1413 IF h>0:FOR i=0 TO 476 STEP h:BLOCK 1,10,13+i,194,6

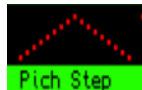
```



```

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

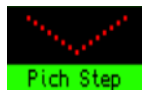
```



```

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

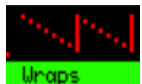
```



```

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 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#3,11,25: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

```

GROUP ROW
+ 00 + 00

```

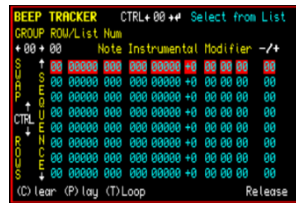
1210 DEFine PROCEDURE RowChg(chg%) :REMark Change Tracker Row
1211 IF m%=1:RETurn :RowNu 0,py%:RowPt 0,py%,pr%:
1212 py%=py%+chg%:pr%=pr%+chg%: RowUp:RowDn:RowNu 7,py%:RowPt 2,py%,pr%
1213 END DEFine

```

```

1215 DEFine PROCEDURE RowUp
1216 IF py%<0 AND pr%> 0:SCROLL#5, 11:py%=0:RowPt 0,py%,pr%
1217 IF pr%<1:pr%=0:py%=0
1218 END DEFine

```



```

1220 DEFine PROCEDURE RowDn
1221 IF py%>7 AND pr%<23:SCROLL#5,-11:py%=7:RowPt 0,py%,pr%
1222 IF pr%>=23:pr%=23:py%=7
1223 END DEFine

```

```

1225 DEFine PROCEDURE TInst(chg%) :REMark Change Tracker Instrumental
1226 IF chg%=-1 AND n%>0 :n%=n%-1
1227 IF chg%=+1 AND n%<95:n%=n%+1
1228 IF m%=1:RETurn:ELSE CURSOR#3,143,2:PRINT#3,FILL$('0',2-LEN(n%))&n%:BSet
1229 END DEFine

```

BEEP TRACKER CTRL+00+ Select from List

```

1231 DEFine PROCEDURE TEntry :REMark Update BEEP/Tracker Entry
1232 Score(sl%,pr%,0)=n%:Score(sl%,pr%,3)=r%:RowPt 2,py%,pr%
1233 END DEFine

```

```

1235 DEFine PROCEDURE RowNu(col,py%) :REMark:Show Row Number
1236 INK#3,col:CURSOR#3,37,25:PRINT#3,FILL$('00',2-LEN(pr%))&pr%
1237 END DEFine

```

GROUP ROW/List
+ 00 + 04

```

1239 DEFine PROCEDURE RowPt(scol,py%,pr%) :REMark Print Tracker Row
1240 FOR a=0 TO 7
1241 num%=Score(sl%,pr%,0):str%=Bkeys(num%,a):STRIP#5,scol
1242 num%=FILL$('0',2-LEN(num%))&num%:CURSOR#5,0,py%*11:PRINT#5,num$
1243 SElect ON a

```

31 0000 00 00 0001 +2 00 00 00 00

```

1244 =0 :CURSOR#5,18+a*22,py%*11:PRINT#5,FILL$('0',5-LEN(str%))&str%
1245 =1 :CURSOR#5,32+a*22,py%*11:PRINT#5,FILL$('0',3-LEN(str%))&str%
1246 =2 :CURSOR#5,36+a*22,py%*11:PRINT#5,FILL$('0',3-LEN(str%))&str%
1247 =3 :CURSOR#5,36+a*22,py%*11:PRINT#5,FILL$('0',5-LEN(str%))&str%
1248 =4 :CURSOR#5,136 ,py%*11:PRINT#5,FILL$('+',2-LEN(str%))&str%
1249 =5,6,7:CURSOR#5,76+a*16,py%*11:PRINT#5,FILL$('0',2-LEN(str%))&str%
1250 END SElect
1251 r%=Score(sl%,pr%,3):CURSOR#5,216,py%*11:PRINT#5,FILL$('0',2-LEN(r%))&r%
1252 END FOR a
1253 END DEFine

```

```

1255 DEFINE PROCEDURE TDecay(chg%) :REMark Change Tracker Decay
1256 IF chg%=-1 AND r%> 0:r%=r%-1
1257 IF chg%=+1 AND r%<15:r%=r%+1
1258 Score(sl%,pr%,3)=r%:RowPt 0,py%,pr%
1259 END DEFINE

```

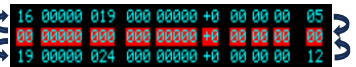
Note: Sets Delay between Rows being Played [0..15]



```

1261 DEFine PROCEDURE TSwap(chg%) :REMark Swap Rows
1262 IF chg%=-1 AND pr%= 0 OR chg%=-1 AND py%=0:RETURN
1263 IF chg%=+1 AND pr%>22 OR chg%=+1 AND py%=7:RETURN
1264 IF m%=1:RETURN:ELSE DIM Tmp(4):RowNu 0,py%+chg%,pr%
1265 FOR i=0 TO 4
1266 Tmp(i)=Score(sl%,pr%+chg%,i)
1267 Score(sl%,pr%+chg%,i)=Score(sl%,pr%,i)
1268 Score(sl%,pr%,i)=Tmp(i)
1269 END FOR i
1270 py%=py%+chg%:pr%=pr%+chg%:RowNu 7,py%:RowPt 2,py%,pr%
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:CUSOR#3,136,128:PRINT#3,' ← →⬆⬆ ⬇ ← ':BLOCK#3,2,4,202,130,7
1282 REPEAT Row_lp
1283 IF rx%<=rm%:rx%=rm%+1
1284 rm%=FILL$(0',2-LEN(rm%))&rm%:rx$=FILL$(0',2-LEN(rx%))&rx%
1285 CUSOR#3,136,128:PRINT#3,' ← ':rm$,' →⬆⬆ ':rx$,' ⬇':k=CODE(INKEY$(-1))
1286 SELEct ON k
1287 =10:PTrack pt%,rm%, rx%:EXIT Row_lp
1288 =192:IF rm%> 0 :rm%=rm%-1
1289 =200:IF rm%<22 :rm%=rm%+1
1290 =208:IF rx% >rs%+1:rx%=rx%-1
1291 =216:IF rx% <23 :rx%=rx%+1
1292 END SElect
1293 END REPEAT Row_lp
1294 CUSOR#3,136,128:PRINT#3,' ':rm$,' ⬆ ':rx$,' '
1295 END DEFINE

```



```

1297 DEFine PROCEDURE PTrack(pt%,rm%,rx%) :REMark Play Tracker Pattern
1298 BLOCK#3,20,10,190,128,2: BLOCK#3,12,3,194,132,7
1299 REPEAT Track_lp
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:TDel dur:BEEP:TDel del:RowChg 1
1305 IF KEYROW(1)=64:pt%=0:BLOCK#3,20,10,190,128,0:RETURN
1305 END FOR row
1307 END REPEAT Track_lp
1308 END DEFINE

```

Note (P)lay will play all rows in a Pattern Group unless a subset of two or more rows has been selected by (T)Loop.

1352 DEFine PROCEDURE SNew

1353 DIM Score(9,23,4):RChg 0:bn%=0:mn%=120:sl%=0:sn%=0:Schg 2,0

1354 QBSFile\$=":QLSMenu

1355 END DEFine



1357 DEFine PROCEDURE Note_Reset

Set Notes to Default Y/N

1358 CURSOR 8,36:PRINT 'Set Notes to Default Y/N': PAUSE

1359 IF KEYROW(5)=64:CURSOR 170,36:PRINT 'OK ':Init_keys:PAUSE 30

1360 END DEFine



1362 DEFine PROCEDURE Set_QSound

Set Notes with QSound Y/N

1363 CURSOR 8,36:PRINT 'Set Notes with QSound Y/N':PAUSE

1364 IF KEYROW(5)=64:CURSOR 170,36:PRINT 'OK' Qk=1:PScore:ELSE QK=0

1365 END DEFine

1367 DEFine PROCEDURE SEntry

:REMark Score Entry Settings

1368 IF mp%=2:nu%=56:l%=sl%:ELSE nu%=16:l%=sl%+1

1369 Score(l%,sn%,0)=kn% :Score(l%,sn%,1)=ds%

1370 IF ar%=1 OR ar%=2 :Score(l%,sn%,3)=ar%:ELSE Score(l%,sn%,3)=0

1371 na%=36+sn%*7:DSymbol :Score(l%,sn%,4)=nv:ac%=0:ar%=0:Schg mp%,1

1372 END DEFine

1374 DEFine PROCEDURE KOct

:REMark Select Note a to g

1375 LOCAl key,k2,ko%:k1=CODE(INKEY\$(30)):RESTORE 1377

1376 FOR key=97 TO 103:READ ko%:IF k1=key:k2=ko%:EXIT key

1377 DATA 9,11,0,2,4,5,7

1378 SElect on k

1379 =236,240,244,248:k%=47-(k-236)*3)-k2

1380 =238,242,246,250:k%=47-(k-238)*3)-k2:SElect on k2=0,2,5,7,9:k%=k%-1

1381 END SElect : kchg k%

1382 END DEFine

1384 DEFine PROCEDURE KChg(chg%)

:REMark Change MINI Key

1385 OVER#3,-1:BLOCK#3,5,4,ka%,ku%,7:OVER#3,0

1386 kn%=chg%:ka%=276-kn%*5:ku%=50:NChg 0:n%=ns%+kn%+1

1387 IF kn%> 6:ka%=271-kn%*5

1388 IF kn%>11:ka%=266-kn%*5

1389 IF kn%>18:ka%=261-kn%*5

1390 IF kn%>23:ka%=256-kn%*5

1391 IF kn%>30:ka%=251-kn%*5

1392 IF kn%>35:ka%=246-kn%*5

1393 IF kn%>42:ka%=241-kn%*5

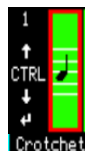
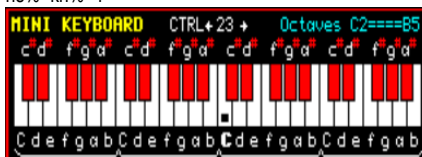
1394 SElect ON kn%=1,3,5,8,10,13,15,17,20,22,25,27,29,32,34,37,39,41,44,46:ku%=36

1395 OVER#3,-1:BLOCK#3,5,4,ka%,ku%,7:OVER#3,0

1396 CURSOR#3,142,2:PRINT#3,FILL\$('0',2-LEN(kn%))&kn%

1397 n%=kn%:Bread:BEEP 0,p,h,t,s,w,f,r :TDel 4:BEEP :Tdel 1

1398 END DEFine



1400 DEFine PROCEDURE NChg(chg%)

:REMark Change Note Symbol display

1401 na%=16:nu%=16:ds%=ds%+chg%: DSymbol

1402 BLOCK 100,10,0,208,0:INK 7:CURSOR 4,208:PRINT N\$

1403 END DEFine

```

1405 DEFINE PROCEDURE SChg(mp%,chg%) :REMark Change Stave Ponter Up/Dn Row
1406 IF m%=0:RETurn:ELSE BLOCK 416,9,80,137,0:sn%=sn%+chg%:IF sn%<0 OR sn%>23:sn%=0
1407 INK 2:CUSOR 33,137:PRINT FILL$( ' ',2-LEN(sn%))&sn%.ma%=36+sn%*7:mu%=42
1408 FILL 1:LINE ma%-2,mu% TO ma%,mu%+mp% TO ma%+2,mu% TO ma%-2,mu%:FILL 0:INK 7
1409 END DEFINE

```

```

1411 DEFINE PROCEDURE RChg(chg%) :REMark Change Displayed Rows
1412 sl%=sl%+chg%:tn%=kn%:INK 7
1413 IF sl%< 0:sl%=0
1414 IF sl%> 8:sl%=8
1415 IF sl%<9:CUSOR 6,126:PRINT sl%:CURSOR 6,147:PRINT sl%+1
1416 FOR sn=0 TO 23
1417   na%=36+sn*7
1418   nu%=56:kn%=Score(sl% ,sn,0) :ds=Score(sl% , sn,1)
1419   ar%=Score(sl% , sn,3) :DSymbol
1420   nu%=16:kn%=Score(sl%+1,sn,0) :ds=Score(sl%+1,sn,1)
1421   ar%=Score(sl%+1,sn,3) :DSymbol
1422 END FOR sn
1423 kn%=tn%:ds=11:sn%=0 :NChg 0 :TPRnt bn%
1424 END DEFINE

```



```

1426 DEFine PROCEDURE PScore :REMark Play Score
1427 LOCAl l,lmin,lmax:mp%=1:nt%=kn% : IF k=81 OR k=113:Qk=1:ELSE QK=0
1428 IF k=112 OR k=113 :lmin=sl%:lmax=sl%+1:ELSE lmin=0:lmax=9:sl%=0
1429 REPEAT Play_lp
1430   FOR l=lmin TO lmin+1
1431     IF l=lmin:mp%=3:ELSE mp%=-3
1432     FOR n=0 TO 23
1433       ds =Score(l,n,1):IF ds<3:NEXT n
1434       n% =Score(l,n,0):sn%=n:SCHg mp%,0:IF KEYROW(1)=64:EXIT Play_lp
1435       nv =Score(l,n,4):dur=INT(3000*nv/mn%):del=4
1436       IF Score(l,n,3)=1:del=8:dur=dur-1 :REMark Staccato
1437       IF Score(l,n,3)>1:del=2:dur=dur+2 :REMark Tenuto
1438       IF ds<8:TDel dur+del
1439       IF ds>7:BRead:BEEP 0,p,h,t,s,w,f,r:TDel dur:BEEP:TDel del
1440       IF ds>7 AND Qk=1
1441         qn=n% : oct%=5-(n% DIV 12) Note: Get Keyboard QNote & Octave
1442         qstr$=v16w10x8500o'&oct%&QNotes(qn)
1443         SOUND_AY :PLAY 1,qstr$:TDel dur:SOUND_AY:TDel del
1444       END IF
1445     END FOR n
1446   END FOR l
1447   IF lmin<=lmax-2:lmin=lmin+2:RChg 2:ELSE EXIT Play_lp
1448 END REPEAT Play_lp
1449 BEEP :kn%=nt%:sn%=0:ds=12:NChg 0:mp%=2:SChg mp%,0
1450 END DEFINE

```

```

1452 DEFine FuNction QNotes(qn) :REMark QSound Notes
1453 RESTORE 1445:oct%=5-qn DIV 12:oct$='o'&oct%:not%=11-qn MOD 12
1454 FOR i=0 to 11:READ not$:IF not%='i:str$=oct$&not$:RETurn str$
1455 DATA 'C','C#','D','D#','E','F','F#','G','G#','A','A#','H'
1456 END DEFine

```



1458 **DEFINE PROCEDURE DSymbol** :REMark Select Music Symbol **Note:** Display Symbol

1459 **SSpace**:INK 0:as%=0:n%=kn%:sym=ds%:shp=kn%:IF kn%>27:n%=kn% MOD 27+3

1460 IF ds%>7

1461 **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

1462 END IF

1463 IF ds%>7 AND MKey(n%) =-6 :lu%=nu% -3:**Ledger**:lu%=lu%-3:**Ledger**

1464 IF ds%>7 AND MKey(n%) =-4.5 :lu%=nu% -3:**Ledger**

1465 IF ds%>7 AND MKey(n%) =-3 :lu%=nu% -3:**Ledger**

1466 IF ds%>7 AND MKey(n%) =13.5 :lu%=nu%+12:**Ledger**

1467 IF ds%>7 AND MKey(n%)>=15 :lu%=nu%+12:**Ledger**:lu%=lu%+3:**Ledger**

1468 IF ds%>7:nu%=nu%+MKey(n%) :IF kn%=27:nu%=30:n%=3

1469 **SElect ON sym**

1470 = 0:nv=0 :N\$='Space'

1471 = 1:nv=0 :EBar :N\$='End Bar'

1472 = 2:nv=0 :SBar :N\$='Bar Seperator'

1473 = 3:nv=4 :SBRest :N\$='Semibreve Rest'

1474 = 4:nv=2 :MRest :N\$='Minim Rest'

1475 = 5:nv=1 :CRest :N\$='Crotchet Rest'

1476 = 6:nv=0.5 :QRest :N\$='Quaver Rest'

1477 = 7:nv=0.25 :QRest:SRest :N\$='SemiQuaver Rest'

1478 = 8:nv=4 :Semibreve :N\$='Semibreve'

1479 = 9:nv=3 :Minim :Dot :N\$='Minim+Dot'

1480 =10:nv=2 :Minim :N\$='Minim'

1481 =11:nv=1.5 :Crotchet :Dot :N\$='Crotchet+Dot'

1482 =12:nv=1 :Crotchet :N\$='Crotchet'

1483 =13:nv=0.75 :Quaver : Dot :N\$='Quaver+Dot'

1484 =14:nv=0.5 :Quaver :N\$='Quaver'

1485 =15:nv=0.25 :Semiquaver :N\$='Semiquaver'

1486 **END SElect**

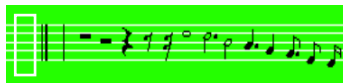
1487 IF ds%>7 AND kn%>26 :QNote nu%

1488 IF ds%>7 AND as% =1 :Sharp

1489 IF ds%>7 AND ar% =1 :Staccato

1490 IF ds%>7 AND ar% =2 :Tenuto

1491 **END DEFINE**



1493 **DEFINE PROCEDURE SSpace** :REMark Clear Stave Entry

1494 LOCAL x,y,su:x=na%-2.6:y=nu%+21.8:INK 4

1495 FILL 1:LINE x,y TO x+6.6,y TO x+6.6,y-30.8 TO x,y-30.8 TO x,y:FILL 0

1496 INK 7:FOR su=0 TO 12 STEP 3:LINE x,nu%+su TO x+7,nu%+su

1497 **END DEFINE**

1499 **DEFINE PROCEDURE Ledger** :REMark Extra Stave line

1500 INK 7:LINE na%-2.5,lu% TO na%+4.5,lu%:INK 0

1501 **END DEFINE**

1503 **DEFINE PROCEDURE QNote(nu)** :REMark Note Octave Label

1504 INK 0:x1=na%-2:x2=na%+3.5:y1=lu%:C\$=OChk\$(kn%-26)

1505 **SElect ON nu=13 TO 24**:y1=36

1506 **SElect ON nu=25 TO 33**:y1=13

1507 **SElect ON nu=53 TO 64**:y1=77

1508 **SElect ON nu=65 TO 73**:y1=53

1509 STRIP 4:C\$=CURSOR x1,y1,1,1:PRINT C\$:STRIP 0

1510 LINE x1,y1 TO x2,y1 TO x2,y1-5.5 TO x1,y1-5.5 TO x1,y1

1511 **END DEFINE**





1513 **DEFine PROCEDURE EBar** :REMark End Bar
 1514 FILL 1:LINE na%+1,nu% TO na%+1,nu%+12 TO na%+1.6,nu%+12
 1515 LINE TO na%+1.6,nu% TO na%+1,nu%:FILL 0:SBar
 1516 **END DEFine**



1518 **DEFine PROCEDURE SBar** :REMark Bar
 1519 LINE na%,nu% TO na%,nu%+12.5
 1520 **END DEFine**



1522 **DEFine PROCEDURE SBRest** :REMark Semibreve Whole Note
 1523 FILL 1:LINE na%-1,nu%+7.5:LINE_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0
 1524 **END DEFine**



1526 **DEFine PROCEDURE MRest** :REMark Minim Half Note
 1527 FILL 1:LINE na%-1,nu%+6.2:LINE_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0
 1528 **END DEFine**



1530 **DEFine PROCEDURE CRest** :REMark Crotchet Quarter Note
 1531 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
 1532 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
 1533 **END DEFine**



1535 **DEFine PROCEDURE QRest** :REMark Quaver Eighth Note
 1536 LINE na%+.8,nu%+3 TO na%+2,nu%+9:LINE_R TO -2,-2,
 1537 FILL 1:CIRCLE_R 0,.6,.6:FILL 0
 1538 **END DEFine**



1540 **DEFine PROCEDURE SRest** :REMark Semiquaver Sixteenth Note
 1541 LINE na%,nu% TO na%+1.5,nu%+6 TO na%-.5,nu%+3.5
 1542 FILL 1:CIRCLE_R 0,.8,.6:FILL 0
 1543 **END DEFine**



1545 **DEFine PROCEDURE Head**
 1546 CIRCLE na%,nu%,1.5,.6,-PI/4
 1547 **END DEFine**

1549 **DEFine PROCEDURE Stem**
 1550 IF n%<13
 1551 LINE na%-1.1,nu%-.5 TO na%-1.1,nu%-6
 1552 ELSE
 1553 LINE na%+1.2,nu%+.5 TO na%+1.2,nu%+6
 1554 END IF
 1555 **END DEFine**



1557 **DEFine PROCEDURE Flag1**
 1558 IF n%<13:LINE_R TO 2,1.5 TO 0,2.5:ELSE LINE_R TO 2,-1.5 TO 0,-2.5
 1559 **END DEFine**

1561 **DEFine PROCEDURE Flag2**
 1562 IF n%<13:LINE_R TO 0,-1 TO -2,-1:ELSE LINE_R TO 0,1 TO -2,1
 1563 **END DEFine**



1565 **DEFine PROCEDURE Semibreve** :REMark One Musical Note - 4 Beats
1566 CIRCLE na%,nu%,1.4,.7,PI/2
1567 **END DEFine**

1569 **DEFine PROCEDURE Minim** :REMark 2 Beats
1570 **Head:Stem**
1571 **END DEFine**



1573 **DEFine PROCEDURE Crotchet** :REMark 1 beat
1574 FILL 1:Head:FILL 0:Stem
1575 **END DEFine**

1577 **DEFine PROCEDURE Quaver** :REMark ½ Beat
1578 **Crotchet na%,nu%:Flag1**
1579 **END DEFine**



1581 **DEFine PROCEDURE Semiquaver** :REMark ¼ Beat
1582 **Quaver na,nu:Flag2**
1583 **END DEFine**



1585 **DEFine PROCEDURE Sharp**
1586 LINE na%-2.5,nu%+4 TO na%,nu%+4.5:LINE na%-2.5,nu%+3 TO na%,nu%+3.5
1587 LINE na%-2,nu%+4.5 TO na%-2,nu%+2:LINE na%-1,nu%+5 TO na%-1,nu%+2.5
1588 **END DEFine**

1590 **DEFine PROCEDURE Dot**
1591 FILL 1:CIRCLE na%+2.5,nu%,6:FILL 0
1592 **END DEFine**



1594 **DEFine PROCEDURE Staccato**
1595 INK 0:FILL 1
1596 IF n%<13:CIRCLE na%+1,nu%+2.8,.6:ELSE CIRCLE na%+.3,nu%-2.8,.6
1597 FILL 0:INK 7
1598 **END DEFine**



1600 **DEFine PROCEDURE Tenuto**
1601 IF n%<13:LINE na%,nu%+4:ELSE LINE na%,nu%-3
1602 INK 0:FILL 1:LINE_R TO 2,0 TO 0,-.5 TO -2,0 TO 0,.5:FILL 0:INK 7
1603 **END DEFine**



1605 **DEfIne PROCedure GClef**
 1606 na%=16:nu%=56:SSpace:INK 0:LINE 16,58:ARC_R TO 1,3,-PI
 1607 ARC_R TO 0,-5,-PI TO -2,7,-3*PI/4:LINE_R TO 5,7:ARC_R TO -3,1,PI
 1608 LINE_R TO 0,-18:FILL 1:CIRCLE_R -1,0,.8:FILL 0:INK 7
 1609 **END DEfIne**



1611 **DEfIne PROCedure FClef**
 1612 INK 0:FILL 1:CIRCLE 24,25,1.2:FILL 0
 1613 ARC 23,26 TO 27.5,26,-PI:LINE_R TO 0,-2:ARC_R TO -5.5,-6,-PI/4
 1614 FILL 1:CIRCLE 29,26,.6:FILL 0:FILL 1:CIRCLE 29,23,.6:FILL 0
 1615 **END DEfIne**

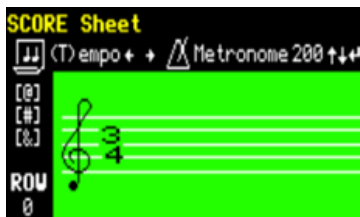


1621 **DEfIne PROCedure TimeSig(x,y)**
 1622 LINE x,y-5 TO x,y TO x+6,y TO x+6,y-6:ARC TO x+4,y-6,-PI/2
 1623 LINE TO x-1,y-6:ARC TO x,y-7,PI/4:LINE TO x+5,y-7
 1624 CIRCLE x+2,y-4,.8,.6,-PI/3:LINE x+2.5,y-4 TO x+2.5,y-1
 1625 CIRCLE x+4,y-4,.8,.6,-PI/3:LINE x+4.5,y-4 TO x+4.5,y-1
 1626 **END DEfIne**



1617 **DEfIne PROCedure Metronome(x,y)**
 1618 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
 1619 **END DEfIne**

1628 **DEfIne PROCedure Tempo** :REMark Set Beat Signature & Metronome Rate
 1629 CURSOR 68,64:PRINT ' ← → ':CURSOR 184,64:PRINT ' ↑ ↓ ← → ':BLOCK 2,4,202,66,7
 1630 **REPeat Tip**
 1631 **TPrint bn%:k=CODE(INKEY\$(-1))**
 1632 **SElect ON k**
 1633 =192:IF bn%> 1:bn%=bn% -1
 1634 =200:IF bn%< 7:bn%=bn%+1
 1635 =208:IF mn%<240:mn%=mn%+10
 1636 =216:IF mn%> 30:mn%=mn% -10
 1637 = 10:**EXIT Tip**
 1638 **END SElect**
 1639 **END REPeat Tip**
 1640 BLOCK 18,10,68,64,0:BLOCK 20,10,184,64,0
 1641 **END DEfIne**



1643 **DEfIne PROCedure TPrint** :REMark Print Tempo
 1644 INK 7:CURSOR 164,64:PRINT FILL\$(' ',3-LEN(mn%))&mn%
 1645 na%=24:nu%=56:SSpace:IF bn%=1::RETuRn
 1646 CSIZE 2,0:INK 0:STRIP 4:CURSOR 54,97:PRINT TP\$(bn%,1)
 1647 CURSOR 56,105:PRINT TP\$(bn%,2):CSIZE 0,0:INK 7:STRIP 0
 1648 **END DEfIne**



1700 REMark File Management

1702 DEFine PROCEDURE KLoad

1703 INK 5:CURSOR 85,22:PRINT '(F)ind:DFn\$=Drv\$(dn%)

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit

1704 FMess 62,22:f%=0:SelPath:IF ck%=0:QLSMenu:RETURN:ELSE FCheck

1705 IF ck%=0 OR eck%=1

1706 CURSOR 10,36:PRINT 'File Not Found...':PAUSE 50:eck%=0:QLSMenu:RETURN

1707 END IF

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit
Loading.....

1708 OPEN_IN#99,DFn\$:CURSOR 10,36:PRINT 'Loading...'

1709 FOR kn=1 TO 96:FOR kp=0 TO 7:INPUT#99,Bkeys(kn,kp):END FOR kp:END FOR kn

1710 FOR sl=0 TO 9

1711 CURSOR 56+sl*6,36:PRINT '':PAUSE 5

1712 FOR sn=0 TO 23:FOR sp=0 TO 4:INPUT#99,Score(sl,sn,sp):END FOR sp:END FOR sn

1713 END FOR sl

1714 INPUT#99,bn%,mn%:CLOSE#99:mn%=(mn% DIV 10)*10

1715 sl%=0:sn%=0:sp%=0:ck%=0:INK 7:QLSMenu

1716 END DEFine

1718 DEFine PROCEDURE KSave

1719 FMess 104,22:SelPath:IF ck%=0:QLSMenu:RETURN:ELSE FCheck

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit
dos1_Beethoven5th_qbs

1720 IF eck%=1

1721 CURSOR 10,36:PRINT 'DEVICE ERROR...'

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit
DEVICE ERROR...

1722 PAUSE 50:eck%=0:QLSMenu:RETURN

1723 END IF

1724 IF ck%=1

1725 CURSOR 10,36:PRINT 'Overwrite Y/N':PAUSE

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit
Overwrite Y/N

1726 IF KEYROW(5)>64:ck%=0:QLSMenu:RETURN

1727 END IF

(M)ode (L)oad (S)ave (F)ind (E)dit (Q)uit
Saving.....

1728 DELETE DFn\$:OPEN_NEW#99,DFn\$:CURSOR 10,36:PRINT 'Saving...'

1729 REMark Bkeys(96,7) - kn%(0-96) kp%(0-8) d,p,h,t,s,w,f,r

1730 FOR kn=1 TO 96:FOR kp=0 TO 7:PRINT#99,Bkeys(kn,kp):END FOR kp

1731 REMark Score(sl%,sn%,sp%) - sl%(0-9) sn%(0-23) sp%(0-4) kn%,ds%,ar%,nv%

1732 FOR sl=0 TO 9

1733 CURSOR 60+sl*6,36:PRINT '':PAUSE 5

1734 FOR sn=0 TO 23:FOR sp=0 TO 4:PRINT#99,Score(sl,sn,sp):END FOR sp:END FOR sn

1735 END FOR sl

1736 PRINT#99,bn%\mn%:CLOSE#99

1737 sl%=0:sn%=0:sp%=0:ck%=0:PAUSE 50:QLSMenu

1738 END DEFine

1740 DEFine PROCEDURE QLSMenu

1741 INK 7:CURSOR 1,22:PRINT '(M)ode (L)oad (S)ave (E)dit (Q)uit'

QBITS QL Sounds
(M)ode (L)oad (S)ave (E)dit (Q)uit

1742 BLOCK 208,18,0,34,0

1743 END DEFine

1745 DEFine PROCEDURE FMess(mx,my)

1746 INK 7:CURSOR mx,my:PRINT ' ← ':BLOCK 10,3,mx,my+4,7:BLOCK 2,4,mx+18,my+2,7

1747 END DEFine



Note: Load - first **Select Device**, this can be changed with Up/Down cursors with preset **Device** names or use (E)dit to change and add a **SubDIR_**. Press 'F' (F)ind and any Filenames with appendix '_qls' found can be selected with Left/Right Cursors followed by Enter, otherwise a **No Files Found** is displayed.

Note: Save - default or current **Devise/SuBDir/fFilename** is displayed Press 'E' to (E)dit If the filename already exists an **Overwrite Y/N** prompt is given. A **'DEVICE ERRPOR'** is shown if **Device/SubDIR** is not available.

Note: Device/SubDIR/Filename will return **'No Files Found'** or **'DEVICE ERROR'** if entry is mistyped.

```

1749 DEFine PROCEDURE SelPath                :REMark Select Device_Filename
1750 QBold 1,5,0,1,32,' ↑':QBold 1,5,0,1,40,' ↓':INK 7 QBold 1,5,0,80,44,' ← →'
1751 REpeat Fsel
1752 sl%=LEN(DFn$):CURSOR 10,36:PRINT DFn$&FILL$(' ',32-sl%)
1753 k=CODE(INKEY$(5))
1754 SElect ON k
1755 =192:IF f%> 1 :f%=f% -1:DFn$=DFn$(1 TO 5)&SFN$(f%)
1756 =200:IF f%<fm%:f%=f%+1:DFn$=DFn$(1 TO 5)&SFN$(f%)
1757 =208:dn%=dn%+1:IF dn%>15:dn%= 0:END IF :DFn$(1 TO 5)=Drv$(dn%)
1758 =216:dn%=dn% -1:IF dn%< 0:dn%=15:END IF :DFn$(1 TO 5)=Drv$(dn%)
1759 =69,101:Ed_Str                          :REMark (E)dit Filename
1760 =70,102:SFiles                          :REMark (F)ind Filenames
1761 = 10:IF fm%>0:ck%=1:EXIT Fsel           :REMark Action Load/Save
1762 = 32: ck%=0:EXIT Fsel                  :REMark Abort Load/Save
1763 END SElect
1764 END REPEAT Fsel
1765 END DEFine

1767 DEFine PROCEDURE FList                  :REMark Create File List
1768 DELETE DFn$(1 to 5)&'FList':OPEN_NEW#9,DFn$(1 to 5)&'FList':DIR#9,DFn$:CLOSE#9
1769 END DEFine

```

```

1771 DEFine PROCEDURE FCheck                :REMark Checks access of File
1772 BLOCK 208,18,0,34,0:CURSOR 10,36:PRINT 'Searching...':PAUSE 30
1773 ck%=0:eck%=0:FList:OPEN_IN#9,DFn$(1 to 5)&'FList':dl%=LEN(DFn$)
1774 REpeat dir_lp
1775 IF EOF(#9):BLOCK 208,10,0,36,0 :CLOSE#9:ck%=0:EXIT dir_lp
1776 INPUT#9,RFn$:IF RFn$=DFn$(6 to dl%):CLOSE#9:ck%=1:EXIT dir_lp
1777 END REPEAT dir_lp
1778 END DEFine

```

```

1780 DEFine PROCEDURE SFiles                :REMark Search For _qbs_Files
1781 DIM SFN$(20,32):f%=1:FList:OPEN_IN#9,DFn$(1 to 5)&'FList'
1782 REpeat dir_lp:
1783 IF EOF(#9) OR f%>20:fm%=f%-1:CLOSE#9:EXIT dir_lp
1784 INPUT#9,RFn$:IF '_qbs' INSTR RFn$<>0:SFN$(f%)=RFn$:f%=f%+1
1785 END REPEAT dir_lp
1786 IF fm%<1:CURSOR 104,36:PRINT 'No Files Found ':PAUSE 30
1787 IF fm%>1:f%=1:DFn$=DFn$(1 to 5)&SFN$(f%):END IF
1788 END DEFine

```

1800 REMark Text Editor

Note: Position Cursor within or at end of string to add a character. Use backspace [Ctrl Left] to Delete Left or Delete Key [CTRL Right] to remove character above Cursor from String. When edited check the first five characters represent a valid device [ie. win1_] and that the end of string is appended with '_qbs'

1802 DEFine PROCEDURE Ed_Str

1803 FMess 146,22:ch%=1:sx%=10:sy%=36:sm%=32

Note: sm% max string length

1804 str\$=DFn\$:sl%=LEN(str\$):cp%=sl%

1805 REPEAT ed_ip

1806 Ln_Prn:Ln_Cur:k\$=INKEY\$(-1):k=CODE(k\$)

1807 SElect ON k

1808 =10:IF str\$(6)=":DFn\$=str\$(1 TO 5):ELSE DFn\$=(str\$:END IF EXIT ed_ip

1809 =32:EXIT ed_ip

1810 =48 TO 57,65 TO 90,95,97 TO 122:Add_Ch

Note: Character Set 0123456789

1811 =194:IF cp%>1:cp%=cp%-1:Del_Ch

ABCDEFGHIJKLMNPOQRSTUVWXYZ

1812 =202:Del_Ch

_ abcdefghijklmnopqrstuvwxyz

1813 =192:IF cp%>1:cp%=cp%-1

1814 =200:IF cp%<sl%+1:cp%=cp%+1

1815 END SElect

1816 END REPEAT ed_ip

1817 INK 7:CURSOR 146,22:PRINT 'dit ':Drv\$(dn%)=str\$(1 to 5)

1818 END DEFine

1820 DEFine PROCEDURE Ln_Prn

1821 sl%=LEN(str\$):IF sl%>sm%:str\$=str%(1 TO sm%):cp%=sm%

1822 INK#ch%,5:CURSOR#ch%,10,36:PRINT#ch%,str\$&FILL\$(' ',32-sl%)

1823 END DEFine

1825 DEFine PROCEDURE Ln_Cur

1826 BLOCK#ch%,6,2,sx%+cp%*6-6,sy%+8,2

1827 END DEFine

1829 DEFine PROCEDURE Add_Ch

1830 IF cp%= 1 AND sl%= 0 :str\$=str\$&k\$

1831 IF cp%>=1 AND cp%<sl% :str\$=str\$(1 TO cp%-1)&k\$&str\$(cp% TO sl%)

1832 IF cp%>=1 AND cp%=sl% :str\$=str\$(1 TO cp%-1)&k\$&str\$(cp%)

1833 IF cp%> 1 AND cp%>sl%:str\$=str\$&k\$

1834 IF cp%=sm%:str\$(cp%)=k\$

1835 IF sl%<sm%:sl%=sl%+1:ELSE sl%=sm%

1836 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%

1837 END DEFine

1839 DEFine PROCEDURE Del_Ch

1840 IF cp%=sl%:str\$=str\$(1 TO sl%-1):sl%=sl%-1

1841 IF cp%>=1 AND cp%<sl%:str\$=str\$(1 TO cp%-1)&str\$(cp%+1 TO sl%):sl%=sl%-1

1842 IF cp%=sm%:str\$=str\$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1

1843 IF cp%=1 AND sl%=1:tr\$=":sl%=0

1844 END DEFine

1850 REMark Initialisation

1852 DEFine PROCEDURE Init_win

```

1853 OPEN#6,scr_:WINDOW#6,512,256,gx,gy:PAPER#6,0:BORDER#6,1,3:CLS#6
1854 OPEN#5,scr_:WINDOW#5,248,88,gx+254,gy+43:PAPER#5,0:CSIZE#5,0,0:INK#5,5
1855 OPEN#4,scr_:WINDOW#4,260,20,gx+14,gy+158:PAPER#4,4:CSIZE#4,0,1:INK#4,0
1856 OPEN#3,scr_:WINDOW#3,10,10,10,10
1857 WINDOW#2,504,220,gx+4,gy+4:PAPER#2,0
1858 WINDOW#1,504,220,gx+4,gy+2:PAPER 0:BORDER 1,5:SCALE 120,0,0
1859 WINDOW#0,512,32,gx,gy+224:PAPER#0,0:BORDER#0,1,3:INK#0,7:CSIZE#0,0,0
1860 CSIZE 2,1:OVER 1:IF Drv$(1)=":FOR d=1 TO 15:drv$(d)="drv"&CHR$(64+d)&"_'
1861 FOR i=2 TO 5:INK i:CUSOR 132+i,36+i:PRINT 'Exploring QL Sounds'
1862 CSIZE 0,1:QBold 1,7,0,160,150, 'Setting TIMER for QL Platform'
1863 sd=DATE:REPeat Chk_lp:IF sd<>DATE:sd=DATE:EXIT Chk_lp
1864 FOR tc=1 TO 1E15:IF sd<>DATE:EXIT tc:END IF :END FOR tc:ms=tc/50
1865 CSIZE 2,1:OVER 1:Init_keys:QBSFile$=":BEEP:TDel 20
1866 INK 2:FOR i=4 TO 6:CUSOR i,2:PRINT 'QBITS QLSounds'
1867 INK 6:FOR i=6 TO 7:CUSOR i,3:PRINT 'QBITS QLSounds'
1868 CSIZE 2,0:OVER 0:QBold 1,6,1,178,5,"":CSIZE 0,0
1869 INK 5:CIRCLE 76,115,2.8:CIRCLE 76,115,3.4:QLSHelp:QLSMenu
1870 END DEFine

```



1872 DEFine PROCEDURE QLSHelp

```

1873 LOCal i,s,x,y,str$
1874 RESTORE 1875:FOR z=1 TO 12:READ i,s,x,y,str$:QBold 0,i,s,x,y,str$
1875 DATA 6,0,6,16,'Select (M)ode <> Sheet'
1876 DATA 6,1,92,16,'BEEP',6,1,142,16,'SCORE',6,1,6,4,'Explore QLSounds'
1877 DATA 7,0,142,4,'Use Cursor keys Spacebar Enter',6,1,234,4,'←↑↓→'
1878 DATA 6,1,272,16,'TRACKER',6,1,352,16,'← →',6,1,432,16,'KEYBOARD'
1879 DATA 6,1,372,4,'←',7,0,332,16,'low CTRL high',6,1,412,4,'F2-F5+cdefgab'
1880 BLOCK#0,14,3,322,8,6:BLOCK#0,2,4,380,6,6
1881 END DEFine

```

Note: Help screen



1900 REMark Mode Setup

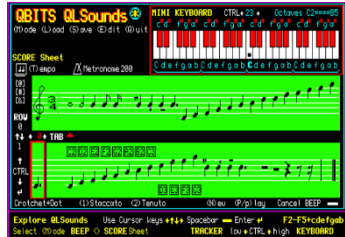
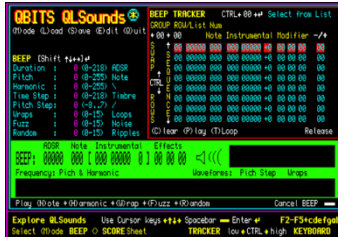
1902 DEFINE PROCEDURE QBMode

1903 BLOCK 200,26,0,50,0:BLOCK 500,143,0,75,0

1904 IF m%=0:MBEEP:TRACKER:rm%=0:rx%=23:TPatn 0:TInst 0:RowNu 7,py%

1905 IF m%=1:KeyBrd:MScore:kn%=23:ka%=146:ku%=50:KChg kn%:RChg 0

1906 END DEFINE



1908 DEFINE PROCEDURE MBEEP

1909 LOCAL i,s,x,y,str\$:m%=0 :BLOCK 496,62,1,145,4

1910 RESTORE 1911:FOR z=1 TO 15:READ i,s,x,y,str\$:QBOLD 1,i,s,x,y,str\$

1911 DATA 6,1,2,52,'BEEP',7,0,40,52,['Shift ↑ ↓ ← →] ← :BLOCK 2,4,118,54,7

1912 DATA 5,0,4, 62,'Duration' : (0-218) ADSR' :REMark d

1913 DATA 5,0,4, 72,'Pitch' : (0-255) Note' :REMark p

1914 DATA 5,0,4, 82,'Harmonic' : (0-255) \ ' :REMark h

1915 DATA 5,0,4, 92,'Time Step' : (0-218) Timbre' :REMark t

1916 DATA 5,0,4, 102,'Pitch Step' : (-8..7) / ' :REMark s

1917 DATA 5,0,4, 112,'Wraps' : (0-15) Loops' :REMark w

1918 DATA 5,0,4, 122,'Fuzz' : (0-15) Noise' :REMark f

1919 DATA 5,0,4, 132,'Random' : (0-15) Ripples' :REMark r

1920 DATA 0,0,50,146,'ADSR Note Instrumental Effects'

1921 DATA 0,0,8,175,'Frequency: Pitch & Harmonic'

1922 DATA 0,0,276,175,'Waveforms: Pitch Step Wraps':STRIP 0

1923 DATA 7,0,8,208,'Play (N)ote +(H)armonic +(W)rap +(F)uzz +(R)andom'

1924 DATA 7,0,400,208,'Cancel BEEP' :BLOCK 16,3,476,212,7

1925 n%=0:BRead:BAtr:BWave:BSet:x=116:y=30:INK 0:ARC x+8,y+3 TO x+8,y-3,PI/2

1926 ARC x+11,y+4 TO x+11,y-4,PI/2:ARC x+14,y+5 TO x+14,y-5,PI/2

1927 LINE x-1,y+1 TO x-1,y+1 TO x+4,y+4 TO x+4,y-4 TO x+1,y-1 TO x-1,y-1

1928 END DEFINE



1930 DEFINE PROCEDURE Tracker

1931 WINDOW#3,292,142,gx+214,gy+4:PAPER#3,0:INK#3,6:BORDER#3,1,2:CLS#3

1932 str\$='SEQUENCE':FOR i=1 TO 8:CUSOR#3,26,37+i*9:PRINT#3,str\$(i)

1933 str\$='SWAP' :FOR i=1 TO 4:CUSOR#3, 2,28+i*8:PRINT#3,str\$(i)

1934 str\$='ROWS' :FOR i=1 TO 4:CUSOR#3, 2,84+i*8:PRINT#3,str\$(i)

1935 RESTORE 1936:FOR z=1 TO 13:READ i,s,x,y,str\$:QBOLD 3,i,s,x,y,str\$

1936 DATA 6,1,0,2,'BEEP TRACKER',7,0,110,2,'CTRL' ← → ←'

1937 DATA 5,0,180,2,'Select from List':BLOCK#3,2,4,170,4,7

1938 DATA 6,0,86,25,'Note Instrumental Modifier',7,1,248,25,'-/+'

1939 DATA 6,0, 2,14,'GROUP ROW/List Num',7,0,2,24,' ← → ',7,0,26,35,' ↑'

1940 DATA 7,0,26,116,' ↓ ',7,0,12,66,' ↑ ',7,-1,2,76,'CTRL',7,0,12,84,' ↓'

1941 DATA 7,0,2,128,'(C)lear (P)lay (T)Loop Release'

1942 END DEFINE



```

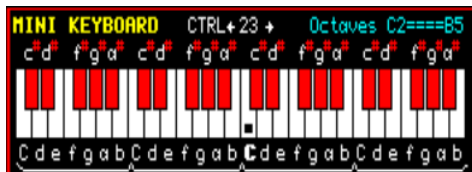
1944 DEFine PROCEDURE MScore
1945 LOCAL z,i,s,x,y,str$:m%=1:BLOCK 468,59,30,76,4:BLOCK 24,62,29,146,2
1946 BLOCK 445,59,53,148,4:na%=16:nu%=56:SSpace :GClef:Schg 2.0:INK 7
1947 For su=0 TO 12 STEP 3:LINE 20,56+su TO 202,56+su:LINE 22,16+su TO 202,16+su :
1948 RESTORE 19497:FOR z=1 TO 15:READ i,s,x,y,str$:QBold 1,i,s,x,y,str$
1949 DATA 6,1,0,52,'SCORE Sheet',7,0,24,64,'(T)empo    Metronome'
1950 DATA 7,0,4,80,'[@]',7,0,4,89,['#'],7,0,4,98,['&'],7,1,2,116,'ROW'
1951 DATA 7,1,4,136,'↑↓',7,1,24,136,'← →',7,1,8,162,'↑',7,1,8,184,'↓'
1952 DATA 7,1,56,137,'TAB',7,0,2,174,'CTRL',7,0,100,208,'(1)Staccato (2)Tenuto'
1953 DATA 7,0,8,196,'←',7,0,292,208,'(N)ew (P)p lay Cancel BEEP'
1954 BLOCK 2,4,14,198,7:BLOCK 16,3,476,212,7:TimeSig 3,85:Metronome 40,85
1955 END DEFine

```

```

1957 DEFine PROCEDURE KeyBrd
1958 WINDOW#3,292,74,gx+214,gy+4:PAPER#3,0:BORDER#3,1,2:CLS#3
1959 QBold 3,6,1,0,2,'MINI KEYBOARD':QBold 3,7,0,110,2,'CTRL ← →'
1960 INK#3,5:CURSOR#3,186,2:PRINT#3,'Octaves C2====B5':REMark sn%=0:Schg 2,0
1961 FOR i=0 TO 27:BLOCK#3,9,30,4+i*10,26,7
1962 FOR i=0 TO 27
1963 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
1964 BLOCK#3,9,18,(i*10)-1,26,0:BLOCK#3,7,17,(i*10),26,2
1965 END FOR i
1966 RESTORE 1972:str$='Cdefgab':ka%=146:ku%=50:BLOCK#3,5,4,ka%,ku%,0
1967 FOR i=1 TO 20:READ sx$,sx$:CURSOR#3,sx%-6,14:PRINT#3,sx$:QHash
1968 FOR a=0 TO 3
1969 FOR b=1 TO 7:CURSOR#3,-5+a*70+b*10,58:PRINT#3,str$(b)
1970 END FOR a
1971 QBold 3,5,1,145,58,'C'
1972 DATA 'c',14,'d',25,'f',44,'g',55,'a',66
1973 DATA 'c',85,'d',96,'f',114,'g',125,'a',136
1974 DATA 'c',155,'d',166,'f',184,'g',195,'a',206
1975 DATA 'c',225,'d',236,'f',254,'g',265,'a',275
1976 LINE#3,4,6 TO 8,2 TO 76,2 TO 78,6
1977 LINE#3,78,6 TO 80,2 TO 148,2 TO 150,6
1978 LINE#3,150,6 TO 152,2 TO 220,2 TO 222,6
1979 LINE#3,222,6 TO 224,2 TO 292,2 TO 296,6
1980 END DEFine

```



```

1982 DEFine PROCEDURE QHash
1983 BLOCK#3,5,1,sx%,14,2 :BLOCK#3,5,1,sx%,16,2
1984 BLOCK#3,1,5,sx%+1,13,2:BLOCK#3,1,5,sx%+3,13,2
1985 END DEFine

```

```

1987 DEFine PROCEDURE QBold(ch%,col%,cs%,cx%,cy%,str$)
1988 INK#ch%,col%:OVER#ch%,1:IF cs%<0:cm%=0:ELSE cm%=cs%
1989 FOR a=1 TO LEN(str$)
1990 FOR b=0 TO cm%:CURSOR#ch%,cx%-6+b+a*(6+cs%),cy%:PRINT#ch%,str$(a)
1991 END FOR a:OVER#ch%,0
1992 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.

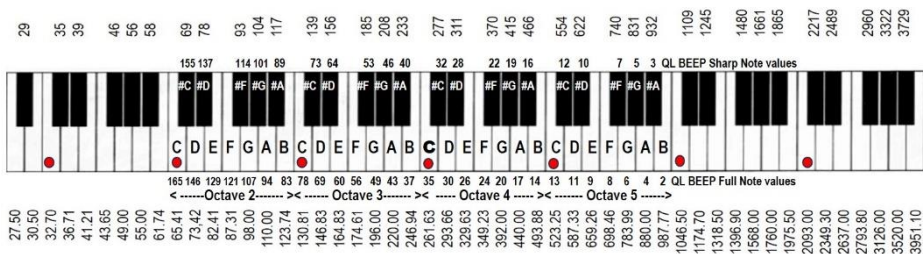
Pitch vs Frequency on the Sinclair QL by Marq

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	207.50 G3#	92	112.65
1	1176.71	47	203.77	93	111.54
2	1066.05 C6	48	200.17	94	110.45 A3
3	974.42 4	49	196.69 G3	95	109.39
	897.29	50	193.34	96	108.34
5	831.48 G5#	51	190.10	97	107.32
6	774.66 G5	52	186.96	98	106.31
7	725.11 F5#	53	183.93	99	105.32
8	681.52 F5	54	180.99	100	104.35 G2#
9	642.87 E5	55	178.15	101	103.40
10	608.37 D5#	56	175.39	102	102.47
11	577.38 D5	57	172.72	103	101.55
12	549.40 C5#	58	170.13	104	100.65
13	524.01 C5	59	167.61	105	99.76
14	500.85 B5	60	165.17	106	98.89
15	479.66	61	162.80	107	98.04 G2
16	460.19 A5#	62	160.49	108	97.20
17	442.24 A5	63	158.25	109	96.37
18	425.63	64	156.07	110	95.56
19	410.23 G4#	65	153.95	111	94.76
20	395.90 G4	66	151.89	112	93.97
21	382.54	67	149.88	113	93.20
22	370.06 F4#	68	147.93	114	92.44
23	358.36 F4	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	327.32 E4	72	140.59	118	89.52
27	318.13	73	138.87	119	88.82
28	309.45 D4#	74	137.19	120	88.13
29	301.22	75	135.55	121	87.45
30	293.43 D4	76	133.94	122	86.78
31	286.02	77	132.38	123	86.12
32	278.99 C4#	78	130.85 C3	124	85.47
33	272.28	79	129.36	125	84.83
34	265.90	80	127.90	126	84.20
35	259.80 C4	81	126.47	127	83.58
36	253.98	82	125.08	128	82.97
37	248.42 B4	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	233.09 A4#	86	119.79	132	80.61
41	228.40	87	118.54	133	80.04
42	223.89	88	117.31 A3#	134	79.48
43	219.55 A4	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	222	49.15 G1
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	186	58.24 A2#	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	197	55.13 A2	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
165	65.29 C2	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	255	43.00 F1
174	62.07	215	50.69		
175	61.73	216	50.47		
176	61.40	217	50.24		32.70 C1
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.

