CPE 3150 KEYBOARD

LanChau LeTran, Joshua Cash, Alan Truong

**Project Description**

The following tables shows the frequency from the speaker for each note, and the preload value that we need for each frequency. To get the preload value, this is the formula that we used:

N = (1/ (4f\*(1/fosc)) = Fosc / 4f

This formula was used in excel to quickly calculated all the notes in each octave.

We have two octaves for each note in order to play the songs correctly as read off a music sheet. The frequency increases as we go from one octave.

This table is the lower octave that we used, along with the frequency and the preload value that we calculated.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C4-B6 | FOUND AT: | <https://pages.mtu.edu/~suits/notefreqs.html> | | | |
| Notes | Frequency | N | Size | N(floored) | Size |
| C | 261.63 | 7045.25475 | LONG | 7045 | LONG |
| C# | 277.18 | 6650.01082 | LONG | 6650 | LONG |
| D | 293.66 | 6276.81673 | LONG | 6276 | LONG |
| D# | 311.13 | 5924.37245 | LONG | 5924 | LONG |
| E | 329.63 | 5591.87574 | LONG | 5591 | LONG |
| F | 349.23 | 5278.04026 | LONG | 5278 | LONG |
| F# | 369.99 | 4981.8914 | LONG | 4981 | LONG |
| G | 392 | 4702.16837 | LONG | 4702 | LONG |
| G# | 415.3 | 4438.35781 | LONG | 4438 | LONG |
| A | 440 | 4189.20455 | LONG | 4189 | LONG |
| A# | 466.16 | 3954.11447 | LONG | 3954 | LONG |
| B | 493.88 | 3732.18191 | LONG | 3732 | LONG |

-Size identify the amount of bits in which the preload value is. Short is 8 bit, and long is 16 bit.

This table is the higher octave that we used, along with the frequency and the preload value that we calculated.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | 523.25 | 3522.6947 | LONG | 3522 | LONG |
| C# | 554.37 | 3324.94543 | LONG | 3324 | LONG |
| D | 587.33 | 3138.35493 | LONG | 3138 | LONG |
| D# | 622.25 | 2962.23383 | LONG | 2962 | LONG |
| E | 659.25 | 2795.98028 | LONG | 2795 | LONG |
| F | 698.46 | 2639.02013 | LONG | 2639 | LONG |
| F# | 739.99 | 2490.91204 | LONG | 2490 | LONG |
| G | 783.99 | 2351.11417 | LONG | 2351 | LONG |
| G# | 830.61 | 2219.15219 | LONG | 2219 | LONG |
| A | 880 | 2094.60227 | LONG | 2094 | LONG |
| A# | 932.33 | 1977.03603 | LONG | 1977 | LONG |
| B | 987.77 | 1866.07206 | LONG | 1866 | LONG |

Next we calculated the amount of time each note should last based on the tempo of the song. When reading music sheets, the duration of the note differs depending on what kind of note it is. For Hot Cross Buns, the tempo we used is 120 bpm. We looked up online the duration or period (in seconds) each whole, half, quarter, and eighth note is to use in our calculation.

N = ( Period \* Clock cycle ) / 2

This equation was used in excel to calculate N for each different type of notes.

The following table gives the preload value for tempo 120 used for Hot Cross Buns.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Time Wanted |  | N | Size | SHRINK N | New Size |
| whole | 1.99 |  | 7336135 | HUGE | 32605 | LONG |
| half | 0.99 |  | 3649635 | HUGE | 16220 | LONG |
| quarter | 0.49 |  | 1806385 | HUGE | 8028 | LONG |
| eighth | 0.24 |  | 884760 | HUGE | 3932 | LONG |

The original calculation for N was of size that was larger than 16 bit. So we shrink it down. Shrink N is the value of N divided by 255 to get down to a 16 bit value. To get the correct tempo, we wrap the shrunken value in a loop that does it 255 times in order to get N.

This next table is at a tempo of 144 for Tetris.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| whole | 1.658 |  | 6112217 | HUGE | 27165 | LONG |
| half | 0.824 |  | 3037676 | HUGE | 13500 | LONG |
| quarter | 0.407 |  | 1500406 | HUGE | 6668 | LONG |
| eighth | 0.208 |  | 766792 | HUGE | 3407 | LONG |