<u>Working with microcontroller-generated audio frequencies</u>

(adapted from the Machine Science tutorial)

If we attach a speaker between a microcontroller output pin and ground, we can "click" the speaker in the same way we blink an LED. When the pin "goes high" (5 volts at the output) the current creates an electromagnetic field that causes the speaker to flex. When the pin "goes low" (zero volts at the output) the speaker contracts back to its previous position. Constantly flexing the speaker back and forth at the right speed will generate an audible tone.
The process of the pin going high for a duration and then going low for a duration is called a "**cycle**", and the total high-pin duration plus the low-pin duration is referred to as the "**period**" of the frequency.

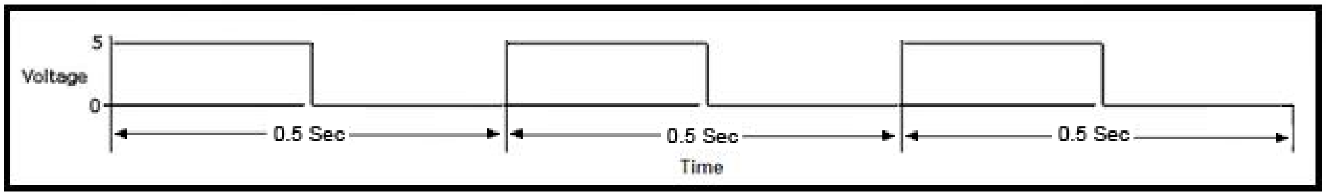
Frequency can be calculated two different ways:

frequency = 1 / period

-or-

frequency = cycles / second

Both equations produce the same result. Consider the example shown in the figure below:



frequency = 1 / period
frequency = 1 / (0.5 seconds)
frequency = 2 cycles / second
-or-
frequency = cycles / second
frequency = (3 cycles) / (1.5 seconds)
frequency = 2 cycles / second

The most common unit of frequency is the Hertz. One cycle per second is equal to one Hertz.

1 Hertz = 1 cycle / second

Therefore, in the example shown above, the frequency is 2 Hertz.

Now that you know how frequency and period are related, it is possible to determine the frequency of the tone your speaker will produce, just by looking at your code. The period of the tone will be the sum of the two delays in each cycle of your for... loop. Using the Machine Science compiler, we can make use of two delay functions: **delay_ms();** -delay in milliseconds and **delay_us();** -delay in micro-seconds (note use of letter "u" to stand for "µ" -the Greek letter "Mu".)

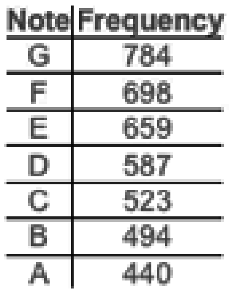
For example, in the following code, the delays are 1,000 microseconds each:

```
   digitalWrite(7, HIGH);
delayMicroseconds(1000);    <------------ a 1000-microsecond delay
    digitalWrite(7, LOW);
delayMicroseconds(1000);    <------------ a 1000-microsecond delay
```

Therefore, to calculate frequency:
frequency = 1 / (1000 microseconds + 1000 microseconds)
frequency = 1 / (2000 microseconds)
frequency = 1 / (0.002 seconds)
frequency = 500 hertz

Playing Specific Frequencies

By adjusting the delays in your for... loop, you can program the microcontroller to produce a tone with a specific frequency--in other words, a musical note. The table below shows seven musical notes, together with their frequencies.



Note	Frequency
G	784
F	698
E	659
D	587
C	523
B	494
A	440

The note A has a frequency of 440 cycles per second; the note B has a frequency of 494 cycles per second; and so on.

Let's start with the following code:

```
//--- "Speaker_Start"  Plays a pitch thru a speaker and lights an LED     ---
//--- Set up:  > Attach a piezo speaker between ground and Digital pin 8
//             > Wire an LED from Digital pin 7 thru a 330 ohm resistor to gnd

#include "machinescience.h"    // included software libraries to use
#include "lcd.h"
int i;                            // Create variable for counting iterations

 void setup() {                   // Only do once at startup
   pinMode(8, OUTPUT);            // Define Speaker pin
   pinMode(6, OUTPUT);            // Define LED pin
   lcd_init();                   // initialize lcd screen
   }

  void loop() {                  // repeat forever
   lcd_clear();
   lcd_instruction(GOTO_LINE1);
   lcd_text("Beep!");            // Print message to LCD screen
   digitalWrite(6, HIGH);        // Turn on LED
   for(i = 0; i < 300; i++)      // number of vibrations (duration)
      {
      digitalWrite(8, HIGH);     // speaker Pin high
      delayMicroseconds(1500);   // delay before releasing speaker
      digitalWrite(8, LOW);      // speaker Pin low
      delayMicroseconds(1500);
     }                           // end loop for one pitch
   digitalWrite(6, LOW);         // Turn off LED
   lcd_clear();
   lcd_text("Quiet.");           // Print message to LCD screen
   delay(1000);                  // wait 1 sec (1000 milliseconds)
}
```

The delays in the above code use an integer of 1500 in the delay statement to generate delays of 1500 millionths of a second. Now, if we wanted to figure out how long the delays in our on/off states (pulses) should be to produce a specific pitch (note), how would we calculate that?

Calculating Pulse Length from Frequency

Let's use note “A” (440 hz) as an example. It's accomplished in two steps:

1. **period = 1000000 / frequency**

Each period (cycle, in hertz) is the length of one second divided by the frequency. We are using a delay measurement is in microseconds (`delay_us();`) so a duration of one second is one million microseconds (1000000). Therefore our period will be 1000000 / 440, or 2272.7272. once we know the total period of one cycle is 2272.7272 millionths of a second, the last part is easy. The delay for each pulse-on and pulse-off is half of that.

2. **pulse = period / 2**

2272.7272 divided by 2 is approximately 1136.3636. So the code in our main loop to generate a frequency of A-440 for a duration of 200 clicks would be:

```
   for(i = 0; i < 200; i++)     // For 200 times (duration)
      {
      digitalWrite(6, HIGH);      // speaker Pin high
      delayMicroseconds(1136);   // delay before releasing speaker
      digitalWrite(6, LOW);       // speaker Pin low
      delayMicroseconds(1136);
     }                           // end loop for one pitch
```

Using a custom function

Let's improve on the previous program with a version that does this math for us in a separate function so our main loop can just play a pitch without having to deal with switching pins high and low. Let's also use variables of type “long” when we need to do floating point calculations.

```
//------------------- a168_Speaker_w_function.c ---------------------
//---  Plays a pitch thru a speaker and lights an LED
//- Set up:  > Attach a piezo speaker between ground and D6 (pin 12)
//           > Wire an LED from D7 (pin 13) thru a 330 ohm resistor to gnd

//---- Here we define a function called “play_pitch”--------
void play_pitch(int freq){    // Define a function called “play_pitch”
   int i;                     // Create variable for counting iterations
   long period;               // Create floating point variable “period”
   long pulse;                // Create floating point variable “pulse”
   period = 1000000/freq;     // Calculate period
   pulse = (period / 2 );     // Calculate pulse
   for(i = 0; i < 200; i++){  // Loop 200 times (duration)
      digitalWrite(6, HIGH);    // Set speaker Pin high
      delayMicroseconds(pulse); // delay for our calculated duration
      digitalWrite(6, LOW);     // Set speaker Pin low
      delayMicroseconds(pulse); // delay for our calculated duration
     }                        // end loop
  }                           // end function definition
//---- Having defined it, now we can use it in our main loop that follows:
void setup(){
    pinMode(6, OUTPUT);      // Define Speaker pin
    pinMode(7, OUTPUT);      // Define LED pin
}
```

```
    void loop (){
   digitalWrite(7, HIGH);     // Turn on LED
     play_pitch(440);        // Run the function we defined above
   digitalWrite(7, LOW);      // Turn off LED
}
```

Adding duration calculation to our custom function:

So far, every time we have played a pitch we've done so by clicking the speaker 200 times. The number of time we click the speaker determines the **duration** of the pitch. This presents a problem if we want to play a melody with each note being the same duration -because click 200 high frequency pulses takes a shorter amount of time than clicking 200 low frequency pulses. We can solve this by including in our function a calculation of duration as well, so the number of times the speaker clicks changes based on the frequency. We'll use a variable called *cycles* in an equation written so that higher frequencies have more speaker clicks. The algorithm has three parts:

1. **We make the following calculations:**

```
period = 1000000 / freq;
pulse = period / 2;
cycles = duration * (freq / 1000.00);
```

2. **We use the variable “cycles” in our For Loop (instead of a hard-coded value like 200)**

```
for(i=0; i<=cycles; i++)
```

3. **In our main loop, when we call our play_pitch() function, we pass along frequency and duration:**

```
play_pitch(440, 100);
```

So putting that together, we can make the following program:

```
//------------------- speaker_freq_dur.c ---------------------
//---  Plays a pitch thru a speaker and lights an LED
//- Set up:  > Attach a piezo speaker between ground and D6 (pin 12)
//           > Wire an LED from D7 (pin 13) thru a 330 ohm resistor to gnd

//---- Here we define a function called “play_pitch”
void play_pitch(int freq, int duration){
   int i;                    // Create variable for counting iterations
   long period;              // Create floating point variable
   long pulse;               // Create floating point variable
   long cycles;              // variable for number of cycles (duration)
   period = 1000000/freq;                //Calculate period
   pulse = (period / 2 );                //Calculate pulse
   cycles = duration * (freq / 1000.00);
   for(i = 0; i < cycles; i++){         // Loop for number of cycles
      digitalWrite(6, HIGH);     // Set speaker Pin high
      delayMicroseconds(pulse);  // delay for our calculated duration
      digitalWrite(6, LOW);      // Set speaker Pin low
      delayMicroseconds(pulse);  // delay for our calculated duration
     }                           // end loop
  }                              // end function definition
//---- Having defined it, now we can use it in the program that follows
void setup(){
    pinMode(6, OUTPUT);       // Define Speaker pin
    pinMode(7, OUTPUT);       // Define LED pin
}
```

```
    void loop() {
   digitalWrite(7, HIGH);          // Turn on LED
     play_pitch(523, 400);      // Run function with frequency & duration
     play_pitch(523, 400);
     play_pitch(523, 400);
     play_pitch(587, 200);
     play_pitch(659, 400);
   digitalWrite(7, LOW);           // Turn off LED
}
```

Now if you initialized variable names like: `int A = 440;` or `int longNote=800;` or `int shortNote=100;`

it would be possible to write statements like:

```
             play_pitch(A, shortNote);
```

Try different variations. Here's a version that generates random pitches in an infinite loop:

```
//------------------ speaker_random_freq.c ---------------------
//---  Plays a pitch thru a speaker and lights an LED
//- Set up:  > Attach a piezo speaker between ground and D6 (pin 12)

//---- Here we define a function called play_pitch
void play_pitch(int freq, int duration){
   int i;                     // Create variable for counting iterations
   long period;               // Create floating point variable
   long pulse;                // Create floating point variable
   long cycles;               // variable for number of cycles (duration)
   period = 1000000/freq;                //Calculate period
   pulse = (period / 2 );                //Calculate pulse
   cycles = duration * (freq / 1000.00);
   for(i = 0; i < cycles; i++){          // Loop for number of cycles
      digitalWrite(6, HIGH);     // Set speaker Pin high
      delayMicroseconds(pulse);  // delay for our calculated duration
      digitalWrite(6, LOW);      // Set speaker Pin low
      delayMicroseconds(pulse);  // delay for our calculated duration
     }                           // end loop
  }                              // end function definition
//---- Having defined it, now we can use it in our main loop that follows
void setup (){
   int r_freq;
   srand((int)15);            //Set "seed" for random number generator
   pinMode(6, OUTPUT);        // Define Speaker pin
}
  void loop() {
   r_freq = rand();           // Grab a random number, store it in "freq"
   r_freq = r_freq/30;        // scale freq by some amount
   play_pitch(r_freq, 150);   // Run function with frequency & duration
 }
}
```