

Stylobit Code

The code for the Stylobit is complex, as it handles a lot of different things, but we'll break down each section and explain what it all does.

```
on start
  led enable false
  set built-in speaker ON
  set volume 255
  set audio pin P0
  set pull pin P1 to up
  set pull pin P2 to up
  set pull pin P3 to up
  set pull pin P4 to up
  set pull pin P5 to up
  set pull pin P6 to up
  set pull pin P7 to up
  set pull pin P8 to up
  set pull pin P11 to up
  set pull pin P12 to up
  set pull pin P13 to up
  set pull pin P14 to up
  set vbuttonPushCounter to 1
  set vbuttonState to 1
  set vlastButtonState to 1
  set octaveMultiplier to 1

forever
  if digital read pin P1 = 0 then
    call playSound 262 x octaveMultiplier
  else if digital read pin P2 = 0 then
    call playSound 277 x octaveMultiplier
  else if digital read pin P3 = 0 then
    call playSound 294 x octaveMultiplier
  else if digital read pin P4 = 0 then
    call playSound 330 x octaveMultiplier
  else if digital read pin P5 = 0 then
    call playSound 349 x octaveMultiplier
  else if digital read pin P6 = 0 then
    call playSound 392 x octaveMultiplier
  else if digital read pin P7 = 0 then
    call playSound 440 x octaveMultiplier
  else if digital read pin P8 = 0 then
    call playSound 494 x octaveMultiplier
  else
    ring tone (Hz) 0 Hz

forever
  set vbuttonState to digital read pin P11
  if vbuttonState != vlastButtonState then
    if vbuttonState = 0 then
      change vbuttonPushCounter by 1
      pause (ms) 100
    if vbuttonPushCounter + floor vbuttonPushCounter
      set vactivatedState to 1
    else
      set vactivatedState to 0
    set vlastButtonState to vbuttonState

forever
  if digital read pin P12 = 0 then
    set octaveMultiplier to 0.5
  else if digital read pin P13 = 0 then
    set octaveMultiplier to 1
  else if digital read pin P14 = 0 then
    set octaveMultiplier to 2
  else
    set octaveMultiplier to octaveMultiplier

function playSound num
  if vactivatedState = 1 then
    for index from 0 to 10
      do
        ring tone (Hz) num + index
        pause (ms) 1
    for index2 from 0 to 10
      do
        ring tone (Hz) num - index2
        pause (ms) 1
  else
    ring tone (Hz) num
```

Stylobit Code

Let's start with the **on start** section.

Typically this section deals with setting up things that only need to happen once at the beginning of the program. We turn off the built-in LED matrix, we turn on the built-in speaker, as well as set the volume, and then we set our pins and a few variables.

Pins 1 through 8 are for our notes and are set as "up" pins, which means they are normally **off** (or **high**) and when a pin is connected to ground (**GND**) they will get pulled down, and set to **low**.

We've also got Pin 11 which will be used to turn on and off the vibrato effect, and Pin 12, 13, and 14 which will be used for setting the octave low, middle, or high.

Pin 11 uses the built-in **Button B** on the front of the micro:bit (though you can also connect your own button if you prefer.)

Pins 12, 13, and 14 will each have a pushbutton connected, and our code will check for one of them to be pressed and then keep that setting for as long as the program is running, changing only when you press a different button.

Finally we set the variables **vbuttonPushCounter**, **vbuttonState**, and **vlastButtonState**. These will be used to track buttons presses for Pin 11 and then toggle the vibrato mode on or off.

The last variable is **octaveMultiplier**, which gets set to **1** but will change based on buttons 12, 13, or 14 being pressed. (Each time you turn on the Stylobit it will default to the middle octave.)

```
on start
  led enable false
  set built-in speaker ON
  set volume 255
  set audio pin P0
  set pull pin P1 to up
  set pull pin P2 to up
  set pull pin P3 to up
  set pull pin P4 to up
  set pull pin P5 to up
  set pull pin P6 to up
  set pull pin P7 to up
  set pull pin P8 to up
  set pull pin P11 to up
  set pull pin P12 to up
  set pull pin P13 to up
  set pull pin P14 to up
  set vbuttonPushCounter to 1
  set vbuttonState to 1
  set vlastButtonState to 1
  set octaveMultiplier to 1
```

Stylobit Code

Next up is the first **forever** loop. (We've created a few forever loops to keep the code a bit more readable.)

The important part of this section is that it checks for each of the Pins (1 through 8) being touched/pressed, and then does something.

Since we set our pins to **up** (which equals **1**) we need to check if they get pulled **down** to ground (which equals **0**).

We use a large **if/else** statement, which checks each button, and if a Pin is triggered it does the appropriate thing (which in this case, is call the **playSound** function.)

If none of the Pins are triggered it does the last else step, and plays a tone with **0**, which means it really plays nothing, and we'll just get silence from the speaker.

Since only one note can be played at a time our Stylobit is referred to as **monophonic**. An instrument that can play multiple notes at the same time is known as **polyphonic**. (Remember, **mono** means **single** and **poly** means **many**.)

You'll notice we don't just call the **playSound** function with the frequency of the note, but we multiply the note by the **octaveMultiplier**.

If the **octaveMultiplier** is set to **1**, the frequency will not change. If the **octaveMultiplier** is set to **0.5** (because you pressed the button on Pin 12) the frequency value will be **half**. If the **octaveMultiplier** is set to **2** (because you pressed the button on Pin 12) the frequency value will be **double**.

Why does this work? Because when you double the frequency you move up one octave. That also means that when you halve a frequency you move down one octave!

```
forever
  if digital read pin P1 = 0 then
    call playSound 262 x octaveMultiplier
  else if digital read pin P2 = 0 then
    call playSound 277 x octaveMultiplier
  else if digital read pin P3 = 0 then
    call playSound 294 x octaveMultiplier
  else if digital read pin P4 = 0 then
    call playSound 330 x octaveMultiplier
  else if digital read pin P5 = 0 then
    call playSound 349 x octaveMultiplier
  else if digital read pin P6 = 0 then
    call playSound 392 x octaveMultiplier
  else if digital read pin P7 = 0 then
    call playSound 440 x octaveMultiplier
  else if digital read pin P8 = 0 then
    call playSound 494 x octaveMultiplier
  else
    ring tone (Hz) 0 Hz
```



Where do those numbers come from? They are the frequencies of the notes we've assigned to our eight keys. If you look at the **ring tone** function in Microsoft MakeCode you'll see how the notes translate to numbers.

You can change the notes we chose by changing the numbers in the code if you want to use different notes.

Stylobit Code

In our **playSound** function the most important (or at least the simplest) part is the **else** section. It contains the **ring tone** command that plays the note.

When we call the **playSound** function from our first **forever** we pass it a **parameter**, which is the frequency of the note. We capture that in the function and place it in the variable named **num**. Then we play that frequency with the **ring tone** command. Simple, right?

What about the **if** section of this code block? Well, the **if** section deals with the vibrato function of the Stylobit, so it's a bit more complex.

If the **vactivatedState** is equal to **1** that means we're playing the note with vibrato. (If you're not familiar with vibrato it is a pulsating change of pitch that gives sound some depth, or a warbling effect.)

When our vibrato is activated we don't just play the note like we do in the **else** section, we create two loops and we play the note 10 times for each loop, and during those first 10 notes we increase the frequency by 1 during each step of the loop, and for the next 10 notes we decrease the frequency by 1 during each step of the loop. We also pause 1 millisecond between each step to give the note some time to play and be heard. This gives the effect of playing the note at the correct frequency, then quickly playing the note 10 steps up, then 10 steps down, and repeating that cycle.

There are a number of ways to do vibrato with code, but we found this to be one of the simpler ones to understand. Like any other parts of the code you can adjust the numbers to see how they change the way the Stylobit makes sound. Experimentation is half the fun!

```
function playSound num
  if vactivatedState = 1 then
    for index from 0 to 10
      do
        ring tone (Hz) num + index
        pause (ms) 1
    for index2 from 0 to 10
      do
        ring tone (Hz) num - index2
        pause (ms) 1
  else
    ring tone (Hz) num
```

Stylobit Code

Our second **forever** loop is pretty simple. It check which of the three octave buttons has been pressed and then assigns **octaveMultiplier** to a number.

Remember when we set the **octaveMultiplier** to 1 in the start section? This is where we can change that.

If the button connected to Pin 12 is pressed, we set the **octaveMultiplier** multiplier to 0.5 which has the effect of cutting the frequency of a note in half, thus lowering it one octave.

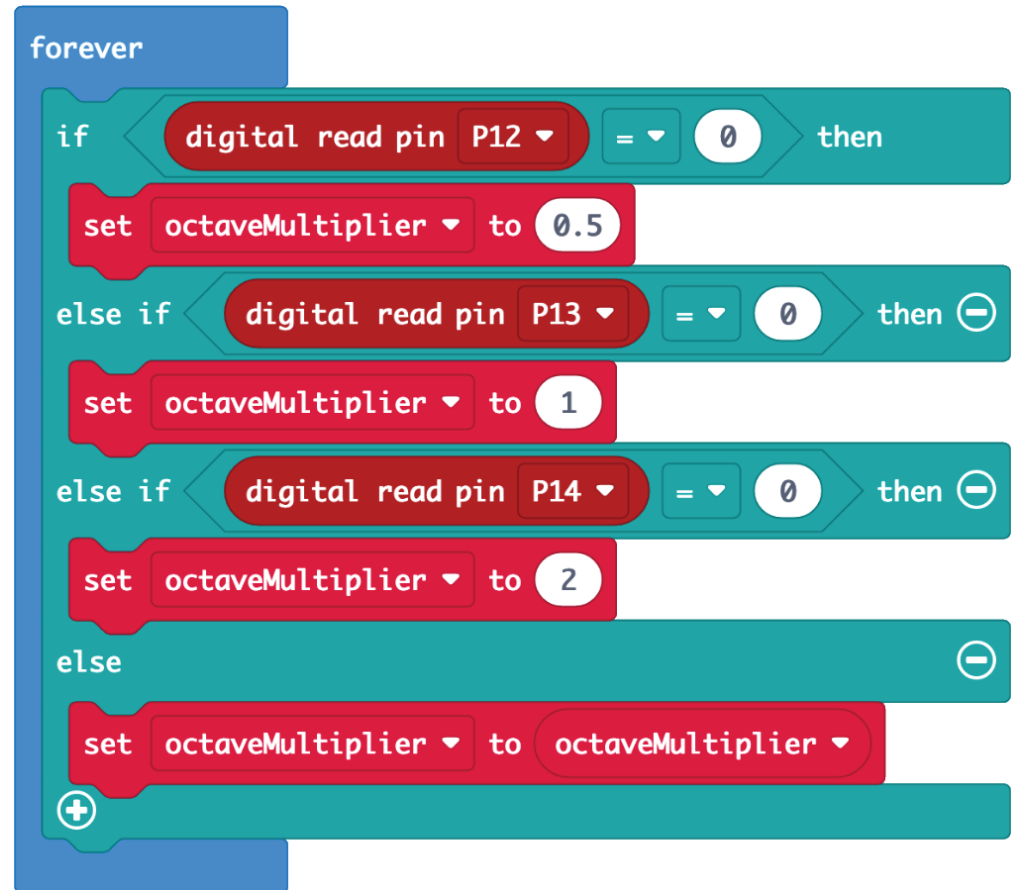
If the button connected to Pin 14 is pressed, we set the **octaveMultiplier** multiplier to 2 which has the effect of doubling the frequency of a note, thus raising it one octave.

Finally, if the button connected to Pin 13 is pressed, we set our **octaveMultiplier** to 1 and it takes the frequency as is without halving it or doubling it.

If no button is pressed our **else** section is triggered and we just set the **octaveMultiplier** to itself.

For our Stylobit we chose to use three buttons to move between three octaves (just the like Stylophone does) but you could modify the code so that pressing the low or high button moves down or up a full octave **for each press**.

With any code, you should feel free to experiment and take it further by adding new features. It can be a fun challenge and it's how you learn to write code.



```
forever
  if digital read pin P12 = 0 then
    set octaveMultiplier to 0.5
  else if digital read pin P13 = 0 then
    set octaveMultiplier to 1
  else if digital read pin P14 = 0 then
    set octaveMultiplier to 2
  else
    set octaveMultiplier to octaveMultiplier
```

The image shows a Scratch code block for a 'forever' loop. The code is written in a block-based language. The 'forever' loop contains four conditional blocks. The first block is an 'if' statement: 'if digital read pin P12 = 0 then'. This is followed by a 'set octaveMultiplier to 0.5' block. The second block is an 'else if' statement: 'else if digital read pin P13 = 0 then'. This is followed by a 'set octaveMultiplier to 1' block. The third block is another 'else if' statement: 'else if digital read pin P14 = 0 then'. This is followed by a 'set octaveMultiplier to 2' block. The fourth block is an 'else' statement, followed by a 'set octaveMultiplier to octaveMultiplier' block. The code is enclosed in a blue 'forever' loop block.

Stylobit Code

The final **forever** loop is a bit more complex. Rather than just checking if a button is pressed (in this case, the button connected to Pin 11, which on the micro:bit is also Button B) we are checking if the **state** of the button has change (meaning, was it not pressed, and then was it pressed).

We do this by saving the state of the button and by keeping track of the last state of the button.

Basically, we start our **vbuttonPushCounter** at 1 and when we press the button the vbuttonPushCounter goes to 2. We press it again and it goes to 3, etc. Increasing by 1 for each button press.

From there we do some maths, dividing the vbuttonPushCounter by the vbuttonPushCounter divided by 2 and rounded down with the floor command.


$2 / (2 / 2) = 2$ so our statement is true, and we set **vactivatedState** to 1

If the count is 2 and we divide by 1 we get 2, so the statement evaluates to **true** and our **vactivatedState** is set to 1.

If the count is 3 and we divide by 1 we get 1, so the statement evaluates to **false** and our **vactivatedState** is set to 0.

This continues on, and all **even** numbers can be divided by half their value and will equal 2 while the **odd** numbers get divided by half their value and then rounded down (thanks to our **floor** command) and do not equal 2.

$4 / (4 / 2) = 2$	TRUE
$5 / (5 / 2) = 2.5$	FALSE
$6 / (6 / 2) = 2$	TRUE
$7 / (7 / 2) = 2.33$	FALSE
etc...	

 If you want to learn more about this method look up the "Modulo operation" for a more in-depth explanation.

```
forever
  set vbuttonState to digital read pin P11
  if vbuttonState != vlastButtonState then
    if vbuttonState = 0 then
      change vbuttonPushCounter by 1
      pause (ms) 100
    if vbuttonPushCounter ÷ floor vbuttonPushCounter ÷ 2 = 2 then
      set vactivatedState to 1
    else
      set vactivatedState to 0
    set vlastButtonState to vbuttonState
```