# Lesson 14 – The Machine Module

# Getting Started

The micro:bit **machine module** contains a number of functions related to the micro:bit hardware. The module allows you access to the restricted hardware such as the **CPU**, timers, buses, etc. **Be aware that if used incorrectly they can lead to malfunction, lockups, crashes of your board, and in extreme cases, hardware damage.**

## **Unique ID**

Every micro:bit has its own unique ID number which is used to distinguish each micro:bit. There are several hundred thousand micro:bits all over the world each with their own unique ID number. You can use the machine module to return the ID number as a byte string. Copy the program below and download the code to your micro:bit. Run it and it will display the ID of your micro:bit.

from microbit import \*

import machine

x = machine.unique\_id()

display.scroll(x)

Complete the table below and compile a list of five of micro:bit IDs from other learners.

|  |  |
| --- | --- |
| **Student name** | **Micro:bit ID** |
| *Example student one* | ??2?@!wW |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## **Machine Frequency**

The Central Processing Unit, the CPU, is like the brain of each micro:bit. It is responsible for processing all the program instructions and then responding. For example, when you press Button A, the smiley face is displayed on the LED matrix. This is controlled by the CPU. It is possible to return the frequency that the CPU is running at. This is a measure of how hard it is working and is measured in **Hertz** (Hz). The higher the frequency the more work the CPU is carrying out. The harder the CPU works the hotter it gets.

Write out the program below and copy download it to the micro:bit.

from microbit import \*

import machine

cpu = machine.freq()

display.scroll(cpu)

Write code to display the frequency at the start of a program and then set the CPU a task, such as displaying an image on the LEDs or scrolling a message. Then run the frequency code at the end of the program and see how it has changed

## **Kernel Panic**

A fun, simple program is the **kernel** panic. Basically, in computing the kernel is a key part of an operating system that loads and remains in the memory of the computer. If an error occurs it goes into a panic, which usually means that it displays some sort of error message. Copy out the program below and run it to enter a panic mode. This requires a restart. Pass a number between 0 and 255 into the code to indicate the status. This number could be used to refer to an error code, which the user could look up.

from microbit import \*

panic(255)

## **Reset**

This code resets the micro:bit in a manner similar to pushing the external **RESET** button. It is useful for ending and restarting a program or breaking out of a loop. Try out the code.

## machine.reset()

## **Running time**

The machine module can keep track of how long the micro:bit has been running for, or how long has passed since the last event. This uses the code microbit.running\_time(). It returns the number of milliseconds since the board was switched on or restarted. The time is measured in milliseconds which are 1000th of a second. One second is 1000 milliseconds, 2 seconds are 2000. Try out the program below which stores the running time in a variable called *x* and then displays the value on the LED matrix.

from microbit import \*

while True:

x = running\_time()

display.scroll(x)

## Success Criteria

* Try out the programs
* Adapt the one of the programs so that it occurs within another program, such as, using the time running to trigger another event, trigger a kernel panic is Button B is pressed or resting the micro:bit if it is shaken

## Pro-tip

## Test Time

Download each of the programs to the micro:bit and test them out.

## Stretch Tasks

Adapt the running program using selection so that it responds with an output at different times. For example, if the micro:bit has been running for 10 seconds then it flashes a sad face on the LEDs, after 20 seconds the micro:bit scrolls a message. You can combine this code with the reset code so that the program loops.

## Final Thoughts

Most of the machine module programs and code can be used in the micro:pet project. Don’t forget to refer back to them.