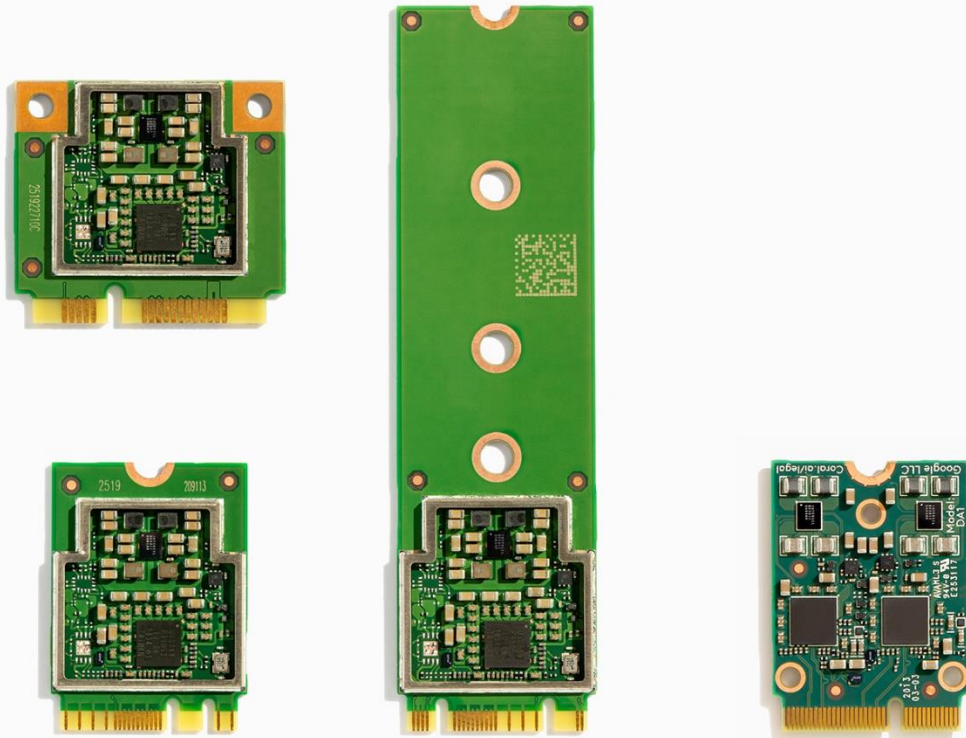


Get started with the M.2 or Mini PCIe Accelerator

To get started with either the Mini PCIe or M.2 Accelerator, all you need to do is connect the card to your system, and then install our PCIe driver, Edge TPU runtime, and the TensorFlow Lite runtime. This page walks you through the setup and shows you how to run an example model.

The setup and operation is the same for both Mini PCIe and M.2 form-factors, including the M.2 Accelerator with Dual Edge TPU.



Requirements

- A computer with one of the following operating systems:
 - Linux: 64-bit version of Debian 10 or Ubuntu 16.04 (or newer), and an x86-64 or ARMv8 system architecture
 - Windows: 64-bit version of Windows 10, and x86-64 system architecture
- All systems require support for MSI-X as defined in the PCI 3.0 specification
- At least one available Mini PCIe or M.2 module slot
- Python 3.6-3.9

1: Connect the module

1. Make sure the host system where you'll connect the module is shut down.

2. Carefully connect the Coral Mini PCIe or M.2 module to the corresponding module slot on the host, according to your host system recommendations.

2: Install the PCIe driver and Edge TPU runtime

Next, you need to install both the Coral PCIe driver and the Edge TPU runtime. You can install these packages on your host computer as follows, either [on Linux](#) or [on Windows](#).

The Coral ("Apex") PCIe driver is required to communicate with any Edge TPU device over a PCIe connection, whereas the Edge TPU runtime provides the required programming interface for the Edge TPU.

2a: On Linux

Before you install the PCIe driver on Linux, you first need to check whether you have a pre-built version of the driver installed. (Older versions of the driver have a bug that prevents updates and will result in failure when calling upon the Edge TPU.) So first follow these steps:

1. Check your Linux kernel version with this command:

```
uname -r
```

If it prints 4.18 or lower, you should be okay and can skip to begin installing our PCIe driver.

2. If your kernel version is 4.19 or higher, now check if you have a pre-build Apex driver installed:

```
3. lsmod | grep apex
```

If it prints nothing, then you're okay and continue to install our PCIe driver.

If it does print an Apex module name, stop here and follow the [workaround to disable Apex and Gasket](#).

Now install the PCIe driver and runtime as follows:

1. First, add our Debian package repository to your system (be sure you have an internet connection):

```
2. echo "deb https://packages.cloud.google.com/apt coral-edgetpu-stable main" | sudo tee
   /etc/apt/sources.list.d/coral-edgetpu.list
3.
4. curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
5.
6. sudo apt-get update
```

7. Then install the PCIe driver and Edge TPU runtime packages:

```
8. sudo apt-get install gasket-dkms libedgetpu1-std
```

9. If the user account you'll be using does not have root permissions, you might need to also add the following udev rule, and then verify that the "apex" group exists and that your user is added to it:

```
10. sudo sh -c "echo 'SUBSYSTEM==\"apex\", MODE=\"0660\", GROUP=\"apex\"' >> /etc/udev/rules.d/65-
    apex.rules"
11.
12. sudo groupadd apex
13.
14. sudo adduser $USER apex
```

15. Now reboot the system.

16. Once rebooted, verify that the accelerator module is detected:

```
17. lspci -nn | grep 089a
```

You should see something like this:

```
03:00.0 System peripheral: Device 1ac1:089a
```

The 03 number and System peripheral name might be different, because those are host-system specific, but as long as you see a device listed with 089a then you're okay to proceed.

18. Also verify that the PCIe driver is loaded:

```
19. ls /dev/apex_0
```

You should simply see the name repeated back:

```
/dev/apex_0
```

Now continue to [install PyCoral and TensorFlow Lite](#).

2b: On Windows

You can install both the PCIe driver and the Edge TPU runtime on Windows using our install script as follows:

1. First, make sure you have the latest version of the [Microsoft Visual C++ 2019 redistributable](#).
2. Then [download edgetpu runtime 20220308.zip](#).
3. Extract the ZIP files and double-click the install.bat file inside.

A console opens to run the install script. When it asks whether you want to enable the maximum operating frequency, you can answer either "yes" or "no" and it has no effect, because this setting only affects devices that operate over USB. Because this device instead operates over PCIe, it uses the maximum operating frequency by default, and may perform power throttling based on the Edge TPU temperature, as specified by [PCIe driver parameters](#).

That's it. Now install PyCoral and TensorFlow Lite...

3: Install the PyCoral library

PyCoral is a Python library built on top of the TensorFlow Lite library to speed up your development and provide extra functionality for the Edge TPU.

We recommend you start with the PyCoral API, and we use this API in our example code below, because it simplifies the amount of code you must write to run an inference. But you can build your own projects using TensorFlow Lite directly, in either Python or C++.

To install the PyCoral library, use the following commands based on your system.

3a: On Linux

If you're using Debian-based Linux system, install PyCoral (and TensorFlow Lite) as follows:

```
sudo apt-get install python3-pycoral
```

3b: On Windows

If you're using Windows, install PyCoral (and TensorFlow Lite) as follows:

```
python3 -m pip install --extra-index-url https://google-coral.github.io/py-repo/ pycoral~=2.0
```

Windows users: Instead of typing python3 as shown here (and elsewhere in our docs), you can use the [py launcher](#). Just be sure you use Python 3.5 or newer.

Alternatively, you can [download a specific PyCoral wheel file](#) and pass it to pip install.

4: Run a model on the Edge TPU

Now you're ready to run an inference on the Edge TPU.

Windows users: The following code relies on a Bash script to install dependencies. If you're new to using Bash on Windows, we suggest you try either [Windows Subsystem for Linux](#) or Git Bash from [Git for Windows](#).

Follow these steps to perform image classification with our example code and MobileNet v2:

1. Download the example code from GitHub:

```
2. mkdir coral && cd coral
3.
4. git clone https://github.com/google-coral/pycoral.git
5.
   cd pycoral
```

6. Download the model, labels, and bird photo:

```
bash examples/install_requirements.sh classify_image.py
```

7. Run the image classifier with the bird photo (shown in figure 1):

```
8. python3 examples/classify_image.py \
9. --model test_data/mobilenet_v2_1.0_224_inat_bird_quant_edgetpu.tflite \
10. --labels test_data/inat_bird_labels.txt \
    --input test_data/parrot.jpg
```



Figure 1. parrot.jpg

You should see results like this:

```
INFO: Initialized TensorFlow Lite runtime.
```

```
----INFERENCE TIME----
```

```
Note: The first inference on Edge TPU is slow because it includes loading the model into Edge TPU memory.
```

```
11.8ms
```

```
3.0ms
```

```
2.8ms
```

```
2.9ms
```

```
2.9ms
```

```
-----RESULTS-----
```

```
Ara macao (Scarlet Macaw): 0.75781
```


Congrats! You just performed an inference on the Edge TPU using TensorFlow Lite.

To demonstrate varying inference speeds, the example repeats the same inference five times. Your inference speeds might differ based on your host system.

The top classification label is printed with the confidence score, from 0 to 1.0.

To learn more about how the code works, take a look at the [classify_image.py source code](#) and read about how to [run inference with TensorFlow Lite](#).

Note: The example above uses the PyCoral API, which calls into the TensorFlow Lite Python API, but you can instead directly call the TensorFlow Lite Python API or use the TensorFlow Lite C++ API. For more information about these options, read the [Edge TPU inferencing overview](#).

Next steps

Important: To sustain maximum performance, the Edge TPU must remain below the maximum operating temperature specified in the datasheet. By default, if the Edge TPU gets too hot, the PCIe driver slowly reduces the operating frequency and it may reset the Edge TPU to avoid permanent damage. To learn more, including how to configure the frequency scaling thresholds, read how to [manage the PCIe module temperature](#).

To run some other models, such as real-time object detection, pose estimation, keyphrase detection, on-device transfer learning, and others, check out our [example projects](#). In particular, if you want to try running a model with camera input, try one of the several [camera examples](#).

If you want to train your own model, try these tutorials:

- [Retrain an image classification model using post-training quantization](#) (runs in Google Colab)
- [Retrain an image classification model using quantization-aware training](#) (runs in Docker)
- [Retrain an object detection model using quantization-aware training](#) (runs in Docker)

Or to create your own model that's compatible with the Edge TPU, read [TensorFlow Models on the Edge TPU](#).

The following section describes how the power throttling works and how to customize the trip points.

Troubleshooting on Linux

Here are some solutions to possible problems on Linux.

HIB error

If you are running on ARM64 platform and receive error messages such as the following when you run an inference...

```
HIB Error. hib_error_status = 0000000000002200, hib_first_error_status = 0000000000000200
```

... You should be able to solve it if you modify your kernel command line arguments to include `gasket.dma_bit_mask=32`.

For information about how to modify your kernel command line arguments, refer to your respective platform documentation. For bootloaders based on U-Boot, you can usually modify the arguments either by modifying the `bootargs` U-Boot environment variable or by setting `othbootargs` environment variable as follows:

```
=> setenv othbootargs gasket.dma_bit_mask=32
=> printenv othbootargs
othbootargs=gasket.dma_bit_mask=32
=> saveenv
```

If you make the above change and then receive errors such as, `DMA: Out of SW-IOMMU space`, then you need to increase the `swiotlb` buffer size by adding another kernel command line argument: `swiotlb=65536`.

pcieport error

If you see a lot of errors such as the following:

```
pcieport 0000:00:01.0: PCIe Bus Error: severity=Corrected, type=Data Link Layer, id=0008(Transmitter ID)
pcieport 0000:00:01.0: device [10de:0fae] error status/mask=00003100/00002000
pcieport 0000:00:01.0: [ 8] RELAY_NUM Rollover
pcieport 0000:00:01.0: [12] Replay Timer Timeout
pcieport 0000:00:01.0: PCIe Bus Error: severity=Uncorrected (Non-Fatal), type=Transaction Layer, id=0008(Requester ID)
pcieport 0000:00:01.0: device [10de:0fae] error status/mask=00004000/00000000
```

... You should be able to solve it if you modify your kernel command line arguments to include `pcie_aspm=off`. For information about how to modify your kernel command line arguments, refer to your respective platform documentation. If your device includes U-Boot, see the previous [HIB error](#) for an example of how to modify the kernel commands. For certain other devices, you might instead add `pcie_aspm=off` to an APPEND line in your `system /boot/extlinux/extlinux.conf` file:

```
LABEL primary
MENU LABEL primary kernel
LINUX /boot/Image
INITRD /boot/initrd
APPEND ${cbootargs} quiet pcie_aspm=off
```

Workaround to disable Apex and Gasket

The following procedure is necessary only if your system includes a pre-build driver for Apex devices (as per the first steps for [installing the PCIe driver](#)). Due to a bug, updating this driver with ours can fail, so you need to first disable the apex and gasket modules as follows:

1. Create a new file at `/etc/modprobe.d/blacklist-apex.conf` and add these two lines:

```
2. blacklist gasket
3. blacklist apex
```

4. Reboot the system.
5. Verify that the apex and gasket modules did not load by running this:
6. `lsmod | grep apex`

It should print nothing.

7. Now follow the rest of the steps to [install the PCIe driver](#).
8. Finally, delete `/etc/modprobe.d/blacklist-apex.conf` and reboot your system.

Manage the PCIe module temperature

Coral products that integrate the Edge TPU over PCIe must be operated using the Coral PCIe driver. This driver handles all device communications, but it also allows you to respond to the Edge TPU temperature and configure dynamic frequency scaling (DFS) thresholds. This page describes how you can use these features to maintain an optimal operating temperature with a PCIe-based Edge TPU.

This document applies to only the following products:

- [System-on-Module](#)
- [Mini PCIe Accelerator](#)
- [M.2 Accelerator A+E key](#)
- [M.2 Accelerator B+M key](#)
- [M.2 Accelerator with Dual Edge TPU](#)
- [Accelerator Module](#)

Note: To install the Coral PCIe driver, see the "get started" guide for your product (follow the above links).

PCIe parameters overview

The PCIe products listed above do not include a thermal solution to dissipate heat from the system. So in order to sustain maximum performance from the Edge TPU and avoid permanent damage, you must design your system so the Edge TPU always operates below the maximum operating temperature specified in the product datasheet.

To help you do so, the Coral PCIe driver includes some programmable parameters that help you manage the Edge TPU temperature in the following ways:

- Read the Edge TPU temperature and then, if necessary, activate a cooling solution (such as a fan) or load-balance your work across other Edge TPUs in the system.
- Use dynamic frequency scaling (DFS)—also known as throttling—to incrementally reduce the Edge TPU operating frequency as it heats up.
- Shut down the Edge TPU when it reaches a critical temperature (highly recommended).

To employ any combination of these strategies, you need to read or write the Coral PCIe driver parameters defined in the following tables.

Exactly how you can read and write these parameters depends on your operating system, and is explained in the following sections (see the instructions [for Linux](#) and [for Windows](#)).

Read the Edge TPU temperature

You can periodically read the Edge TPU temperature using the temp parameter, and then respond with your own strategies to cool the system or load-balance your work.

Table 1. Read-only temperature parameter		
Parameter	Description	Units
temp	<p>The current Edge TPU junction temperature. On Linux, this is available via device-specific sysfs nodes only (not from the kernel module).</p> <p>On Windows, this is available via performance counters only (not from the Windows Registry).</p>	Millidegree Celsius

Use dynamic frequency scaling

By default, the Coral PCIe driver runs the Edge TPU at the maximum frequency of 500 MHz. Under some circumstances, extended operation at this frequency can cause overheating. So the PCIe driver includes a power throttling mechanism (known as dynamic frequency scaling, or DFS) that's enabled by default. This system periodically checks the Edge TPU temperature, and as it reaches the "trip points" specified by parameters in table 2, it reduces the Edge TPU operating frequency in 50-percent increments.

By reducing the operating frequency, the Edge TPU's inferencing speed becomes slower, but it also consumes less power and hopefully avoids reaching higher temperatures at which the Edge TPU may shut down or become permanently damaged.

As long as the chip does not shut down and the Edge TPU returns to lower temperatures, the DFS system restores the operating frequency in the reverse manner—ultimately returning to the maximum operating frequency.

Table 2. Parameters to configure dynamic frequency scaling

Parameter	Description	Default value	Units
trip_point0_temp	If the Edge TPU temperature reaches or exceeds this value, the system sets the operating frequency to "reduced" (250 MHz)	85000	Millidegree Celsius
trip_point1_temp	If the Edge TPU temperature reaches or exceeds this value, the system sets the operating frequency to "low" (125 MHz)	90000	Millidegree Celsius
trip_point2_temp	If the Edge TPU temperature reaches or	95000	Millidegree Celsius

Table 2. Parameters to configure dynamic frequency scaling

Parameter	Description	Default value	Units
	exceeds this value, the system sets the operating frequency to "lowest" (62.5 MHz)		
temp_poll_interval	The interval at which to read the temperature. Setting this to 0 disables DFS completely. This should be several seconds because the temperature reading doesn't change instantly. Yet, it also doesn't need to be much larger than the default because the overhead of switching the operating frequency is negligible, so it isn't necessary to implement hysteresis around the trip points.	5000	Milliseconds

Whatever values you set for the trip_point* parameters, they must evaluate as follows:

trip_point0_temp <= trip_point1_temp <= trip_point2_temp

If you set values that don't match this logic, the driver silently reverts to the default values in table 2.

Note: You cannot manually specify the Edge TPU operating frequency. The Coral PCIe driver always runs the Edge TPU at the maximum frequency (500 MHz), except when it's reduced by DFS, as described above.

Configure the shutdown/warning temperatures

The parameters in table 3 have different behaviors depending on whether you're using the Accelerator Module (the solderable module) or one of the PCIe card modules (such as the Mini PCIe Accelerator or an M.2 Accelerator):

- Accelerator Module: You can specify temperatures at which certain pins assert to warn you that the Edge TPU has reached that temperature. You can respond in whatever way suits your system, such as enabling a fan or shutting down the module.
- PCIe card modules: You can specify the temperature at which the Edge TPU will shut down. You will not receive any warnings. If you want to manually respond to temperature changes, you can instead poll the temp parameter in table 1.

Table 3. Parameters to shut down the Edge TPU

Parameter	Description		Default value	Units
	PCIe card modules	Accelerator Module		
hw_temp_warn1	Not available.	If the Edge TPU reaches or exceeds this temperature, the Edge TPU asserts the INTR line.	100000	Millidegree Celsius
hw_temp_warn1_en	Not available.	Enables/disables hw_temp_warn1.	1	Boolean: 1 = enabled 0 = disabled
hw_temp_warn2	If the Edge TPU reaches or exceeds this temperature, the Edge TPU shuts down. ¹ When the Edge TPU shuts down, it enters an idle state. Generally, you must then restart your system to resume work with the Edge TPU.	If the Edge TPU reaches or exceeds this temperature, the Edge TPU asserts the SD_ALARM line. It's your responsibility to shut down the Accelerator Module.	100000	Millidegree Celsius

Table 3. Parameters to shut down the Edge TPU

Parameter	Description		Default value	Units
	PCIe card modules	Accelerator Module		
hw_temp_warn2_en	Enables/disables hw_temp_warn2.		1	Boolean: 1 = enabled 0 = disabled

¹ This parameter is saved to a register in the Edge TPU (as are all parameters) and the shutdown mechanism is fully contained inside the PCIe card module. So even if the host system fails, the Edge TPU will safely shut down if it reaches this temperature.

Notice: The default values for the temperature warnings are conservative. You should change them based on your hardware's thermal properties. Just be sure the Edge TPU junction temperature never exceeds the maximum rating indicated in the product datasheet.

Warning: We strongly recommend that you use hw_temp_warn2 to shut down the Edge TPU before it exceeds the maximum operating temperature specified in the product datasheet. Failure to do so can result in permanent damage to the Edge TPU and surrounding components, and can possibly cause fire and other serious damage, injury, or death.

Using the parameters on Linux

On Linux, you can access the Coral PCIe driver parameters with files that are accessible as either kernel module parameters or sysfs nodes:

- The kernel module parameters are located in this path:
/sys/module/apex/parameters/
These parameters are persistent and applied at boot time. This is useful if you have multiple modules for which you want to apply the same settings. For details about how to edit these, see [how to specify kernel module parameters](#).
- The sysfs nodes for each module are located at paths such as this:
/sys/class/apex/apex_0
These sysfs nodes are created by the PCIe driver at boot time and allow you to set different settings for different PCIe modules. The file name includes a unique number for each Edge TPU connected via PCIe (such as apex_0, apex_1, apex_2, and so on).

Note: All kernel module parameters apply at system boot time and apply the same setting to all Edge TPUs, whereas the individual sysfs nodes take immediate effect and apply to separate Edge TPUs. Whether you decide to use the kernel module parameters or the individual sysfs node parameters, the files that specify each PCIe parameter are named the same as shown in tables 1, 2, and 3 (although the parameter to read the temperature is available *only* as a sysfs node).

Using the parameters on Windows

On Windows 10, you can access the Coral PCIe driver parameters using the Windows Registry as follows:

1. Launch Registry Editor (type "regedit" from the Run window; you must be admin).
2. Open the following path:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\coral\Parameters

You should see the PCIe parameters as registry keys, as shown in figure 1.

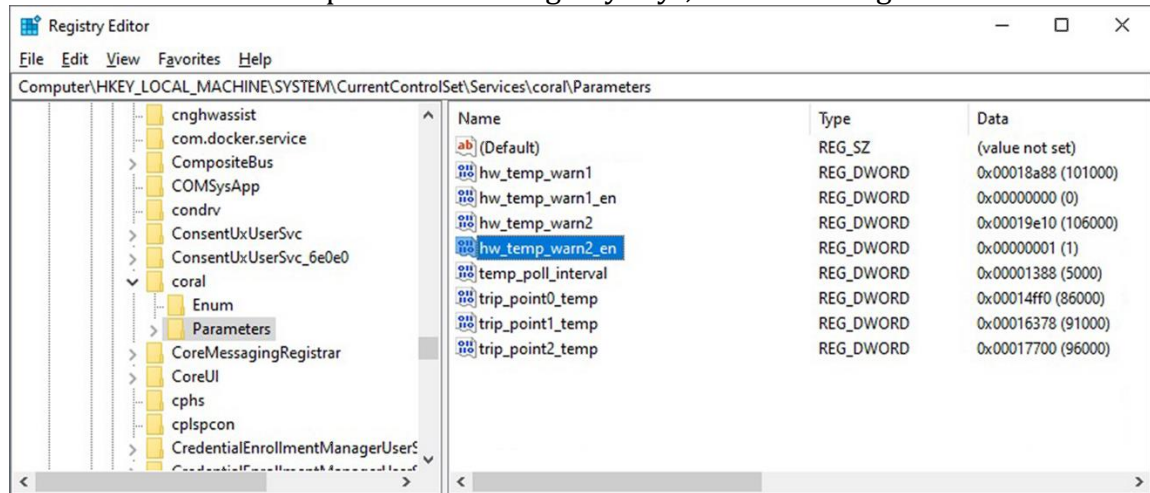



Figure 1. Coral PCIe parameters in Registry Editor

3. Double-click to edit any of the parameters.
4. Reboot your system to apply any changes.

Note: Each PCIe parameter available in the Windows Registry applies the same setting to all Edge TPUs—you cannot set different parameters for separate Edge TPU on Windows.

However, notice that the temp parameter is not available in the Windows Registry, because this parameter changes over time and is read-only. Instead, you can see the current temperature with the Windows Performance Monitor as follows:

1. Launch Performance Monitor (type "perfmon" in the Run window).
2. Select **Performance Monitor** in the left pane, and click **Add**  in the toolbar.
3. In the Add Counters dialog, select the **Coral PCIe Accelerator** counter, select which instances you want to view, and then click **OK**.

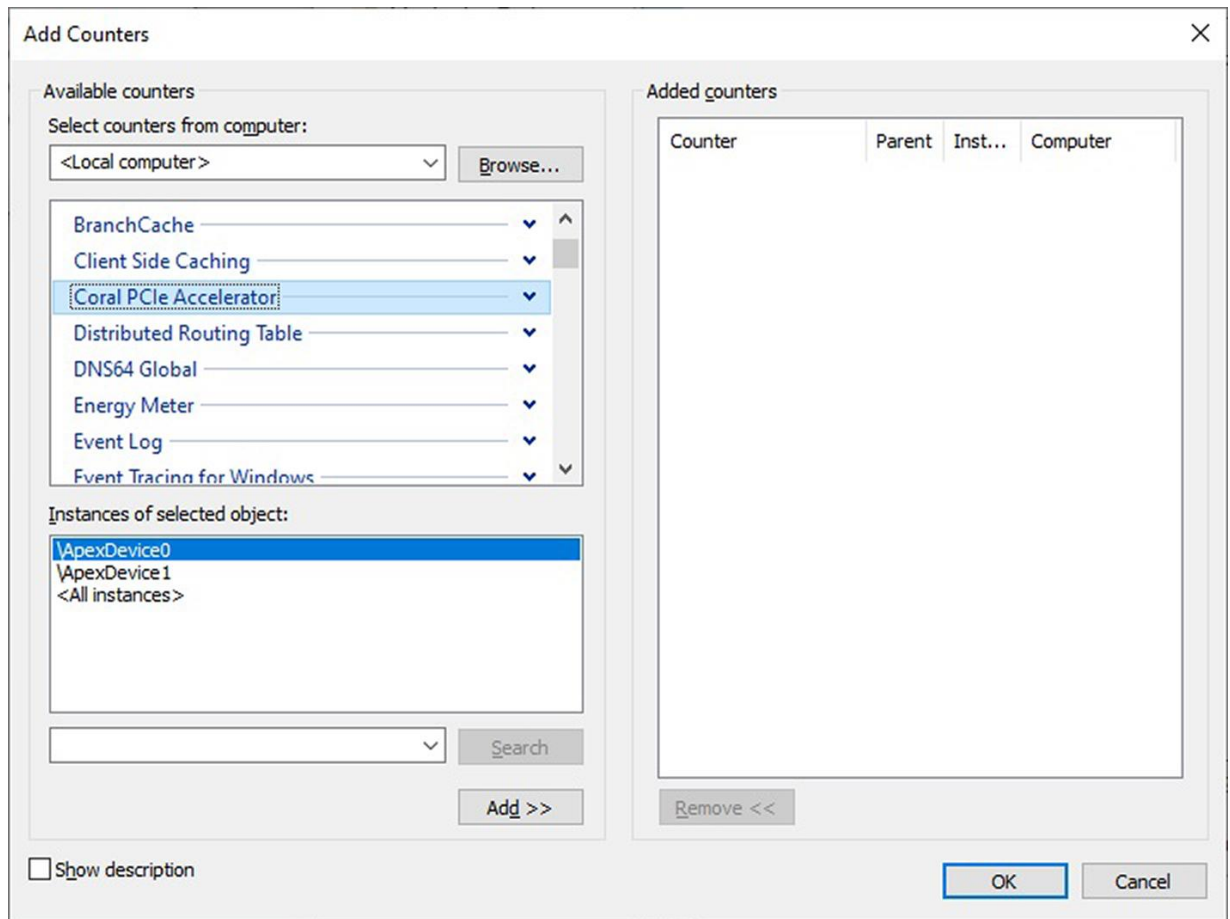


Figure 2. The Add Counters dialog

The activity chart then shows the Edge TPU temperature over time in degrees Celcius. But notice that the actual value below the chart is in millidegree Celsius (as indicated in table 1).

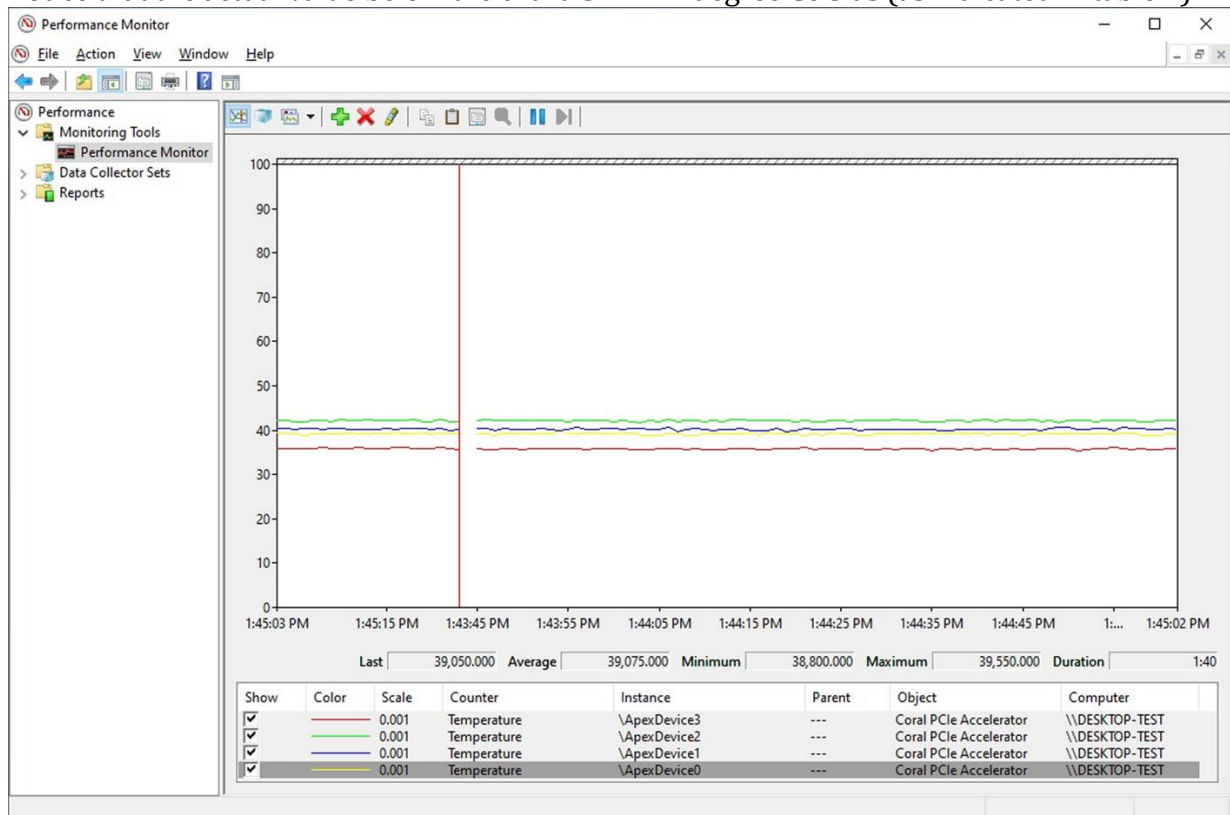


Figure 3. The temperature for multiple Edge TPUs in Performance Monitor

You can also get the Edge TPU temperature with the following PowerShell command:

```
Get-Counter -Counter '\Coral PCIE Accelerator(*)\Temperature'
```

Or, you can write your own tool to [consume performance counter data](#).