SD Card Performance in CircuitPython

Created by Mikey Sklar

https://learn.adafruit.com/microsd-optimization-circuitpython

Last updated on 2026-06-09 05:12:49 PM UTC

Table of Contents

Overview 3

Board Selection 4

- High Performance Boards
- Comparison Chart
- Metro RP2350
- Feather ESP32-S3
- Results

Card Tiers 7

- Card Ratings Compared
- SDIO with an ESP32-S3
- SPI with a Metro RP2350

Filesystems 9

- exFAT for CircuitPython
- Which FAT?
- Slow Startup Due to USB SD Card Presentation

SD Drivers 10

- SD Driver to Chip Mapping

Board Testing 11

- PyPortal M4
- 3.5" FeatherWing + Feather RP2350
- Feather RP2350 + Adalogger FeatherWing
- Feather RP2040 Adalogger
- Memento Camera

Benchmark Setup 17

- Metro RP2350 mount_sd.py
- Feather ESP32-S3 mount_sd.py
- PyPortal M4 mount_sd.py
- Feather RP2350 + 3.5" TFT FeatherWing mount_sd.py
- Feather RP2350 + Adalogger FeatherWing mount_sd.py
- Memento ESP32-S3 mount_sd.py

Overview

This guide will focus on getting the most out of an SD card with CircuitPython. You might be building a project to that needs to store some bitmaps for a digital picture frame, or MP3 files for a tiny audio player. So you use a microcontroller board, a decent SD card and a TFT display. Unfortunately the result can feel a little sluggish. It works, but the images load a little slowly. Or the songs play, but displaying an index of all the song names takes longer than expected. Or you're logging sensor data as fast as it comes in. A model rocket capturing every reading on the way up, and you discover the SD card writes are taking too long and you can't record all the data quickly enough to keep pace. You hope there must be a way to make things go faster.

This guide will start with the fastest setups available on today's Adafruit boards, and then show the performance degradation as you vary each of these factors:

- Board Selection
- Card Tiers
- Filesystems
- SD Drivers
- Board Testing
- Benchmark Setup

Unlike most guides, which build up to the best setup, this one leads with it. SD card activity can be divided up into two broad categories:

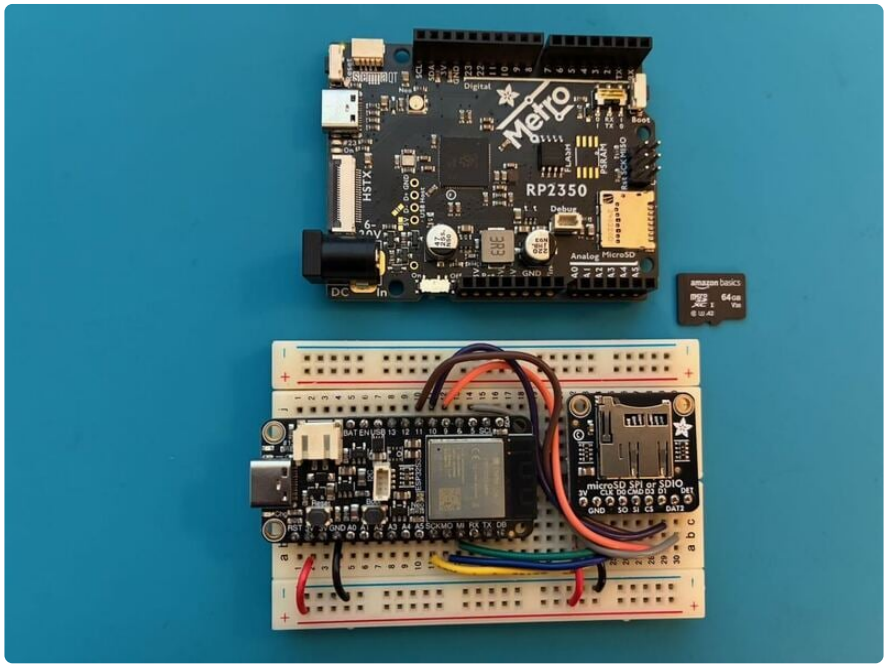
- bulk media read/write (bitmap images, MP3s, GIFs)

- capturing a lot of small data writes quickly (sensor data logging)

It turns out you would use different hardware for each of these use cases:

- bulk media with a Feather ESP32-S3 with a SDIO breakout board
  - 7 MB/s reads (3x faster than RP2350)
  - 4.5 MB/s writes (2x faster than RP2350)
- small data writes with a Metro RP2350 using SPI
  - 2,588 lines per second (1.7x faster than ESP32-S3)

Board Selection



High Performance Boards

The Metro RP2350 and Feather ESP32-S3 have shown the best SD card performance during benchmarking. These boards both produced consistent results. There are several contributing features that have come together to make this possible. By comparing these two setups in more detail, it will become clear what was done to

achieve these speeds. The RP2350 runs the SPI bus to the SD card at 25 MHz (the SPI-mode ceiling); SDIO allows a higher 40 MHz clock.

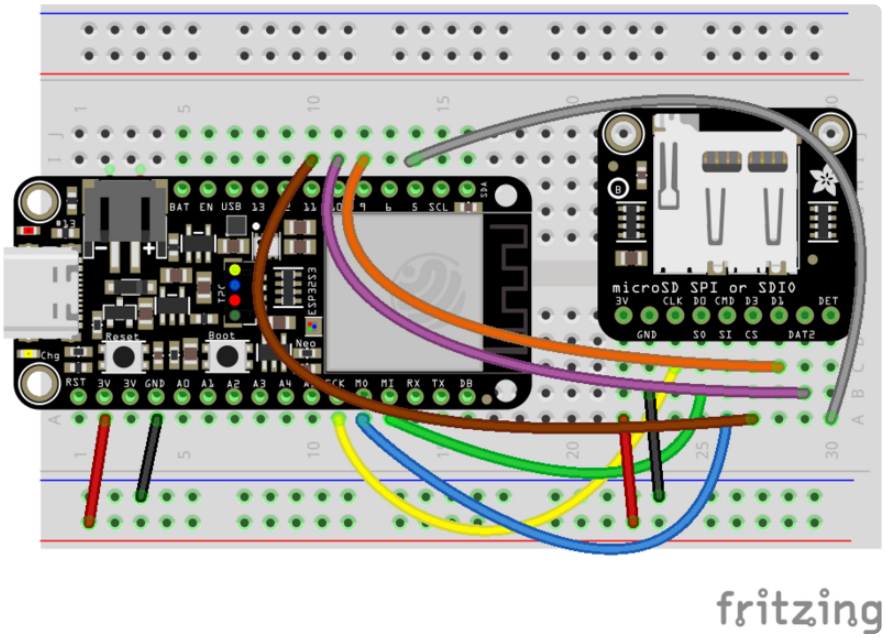
Comparison Chart

	Metro RP2350	Feather ESP32-S3
SD interface	SPI (`sdcardio`)	4-bit SDIO (`sdioio`)
Clock	25 MHz	40 MHz
SD card	64 GB Amazon Basics U3/A2	64 GB Amazon Basics U3/A2
Filesystem	exFAT	exFAT
Breakout required	None (built-in slot)	SD SDIO Breakout PID: 4682
Soldering required	None	headers / breadboard wiring
Other devices on bus	None	None

Metro RP2350

The Metro RP2350 has a built-in SD slot whose four data lines are wired for SDIO. No soldering or add-ons are required. Today CircuitPython drives it in SPI mode using `sdcardio`. There is no PIO implementation of SDIO (at this time). This setup is pure convenience: simply booting the board with a FAT-formatted SD card is enough to get started. The host OS will see the SD card being passed through over USB. RP2040 and RP2350 boards will be using SPI mode with CircuitPython.

Feather ESP32-S3



The Feather ESP32-S3 used here sits on a breadboard, with short jumper wires connecting it to the [Adafruit microSD SPI or SDIO breakout]() board. This setup is running SDIO with the `sdioio` library. The SDIO clock frequency was run at 40 MHz during testing.

Feather ESP32-S3	Breakout #4682	SDIO Signal	Wire Color
SCK	CLK	Clock	▯ Yellow
MOSI	CMD	Command	▯ Blue
MISO	D0	Data 0	▯ Green
D9	D1	Data 1	▯ Orange
D10	DAT2	Data 2	▯ Purple
D11	D3	Data 3	▯ Brown
3V	3V	Power	▯ Red
GND	GND	Ground	● Black

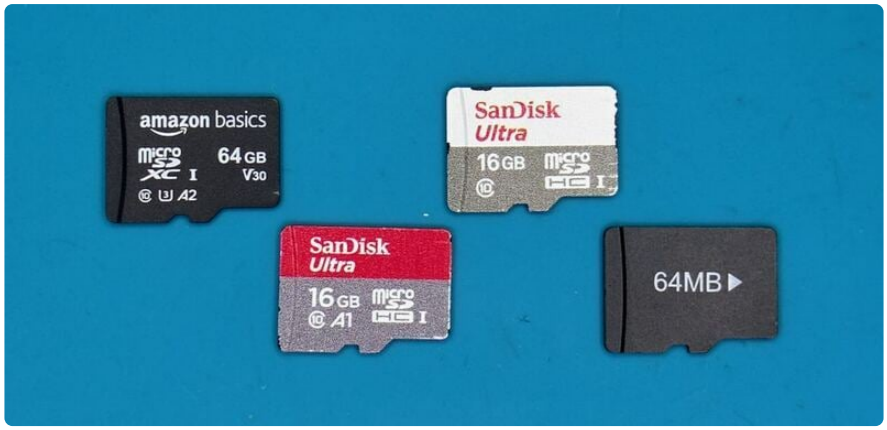
Results

The following results are based on the benchmarking style Jeff Epler originally published in his excellent guide, [Adafruit microSD SPI or SDIO Card Breakout Board]() (https://adafru.it/1aEt).

Bus	Bulk read (MB/s)	Bulk write (MB/s)	Logging (lines/sec)
RP2350 SPI @25MHz	2.3	2.3	2581
ESP32-S3 SDIO @40MHz	7.1	4.5	1489
SDIO vs SPI	**3.1× (better)**	**2.0× (better)**	**0.6× (worse)**

- Bulk reads and writes with the ESP32-S3 can be 2-3 times faster than the RP2350.
- Choose the ESP32-S3 for media such as bitmaps, WAV, and MP3 files.
- For frequent tiny writes, the RP2350 is the clear choice.

Card Tiers



Card Ratings Compared

There are two different stories here. With SDIO, the card absolutely matters. The SDIO setup requires a high tier card with an A2 / U3 rating to keep up. It was able to read over 7 MB/s.

The other story with an SPI setup is that the card won't have a significant effect on bulk throughput. With a single 1-bit SPI line, the cards are indistinguishable. A2, A1, and Class 10 cards all perform the same; throughput only drops with the unlabeled 64 MB card.

If you need to brush up on your card ratings, they're explained in detail in the [Understanding microSD and SD cards: speeds, markings and more](https://adafru.it/1aEu) (https://adafru.it/1aEu).

Understanding microSD and SD cards: speeds, markings and more
By Isaac Wellish
 Markings: speed, size and class

https://learn.adafruit.com/understanding-microsd-and-sd-cards-speeds-markings-and-more/markings-speed-size-and-class

SDIO with an ESP32-S3

Card	Tier	FS	Write MB/s	Read MB/s	Logging (lines/s)
Amazon Basics 64 GB	A2 / U3	exFAT	4.52	7.13	1489
SanDisk 16 GB	A1	FAT32	2.20	5.98	1196
SanDisk 16 GB	Class 10	FAT32	2.44	6.02	1196
64 MB	none	FAT16	0.78	4.03	~1337 (erratic)

SPI with a Metro RP2350

Card	Tier	FS	Write MB/s	Read MB/s	Logging (lines/s)
Amazon Basics 64 GB	A2 / U3	exFAT	2.31	2.27	3861
SanDisk 16 GB	A1	FAT32	2.32	2.10	3661
SanDisk 16 GB	Class 10	FAT32	2.32	2.10	3656

64 MB	none	FAT16	0.94	2.18	2902

Filesystems

exFAT for CircuitPython

You might be surprised to know you can use high capacity SD cards > 32 GB with CircuitPython. These large-capacity cards require exFAT (large filesystem support); exFAT support was added to CircuitPython a while ago, but this was not well publicized.

The only boards that do not support exFAT are boards with very restricted firmware space, like the M0 (SAMD21) boards without extra flash (Trinket M0, GEMMA M0, Feather M0 Basic). M0 Express boards like the Metro M0 do have exFAT support

This means you can use a modern SD card, 64 GB or larger, with great specs like a A2 / U3 rating. Any card greater than 32 GB will need to be formatted with exFAT on your host computer. CircuitPython does not have tools to format these natively: CircuitPython is only able to format FAT12, FAT16 and FAT32.

exFAT provides some wins. Its larger clusters and lack of a FAT chain likely aid performance.

Which FAT?

The storage capacity of your SD card determines the filesystem that gets installed. Yes, you can do wacky things like format a 16GB card with exFAT, but that would be out of compliance with the SD Association capacity standard. This chart is shows how capacity of the SD card dictates the filesystem that is to be used.

Card Type	Capacity	Default Filesystem	CircuitPython Support
SDSC	≤ 2 GB	FAT16	✓ Fully supported
SDHC	4–32 GB	FAT32	✓ Fully supported
SDXC	64 GB–2 TB	exFAT	⚠ Boards with `CIRCUITPY_FULL_BUILD=1` only
SDUC	> 2 TB	exFAT	✗ Not supported

Slow Startup Due to USB SD Card Presentation

Currently, by default, CircuitPython will present a mounted SD card as a USB drive to the host computer. This is a nice convenience so you do not need to manually move the SD card to an SD card reader to read or write it directly. But this convenience comes at a cost: it causes the program to start slowly, because when the host computer mounts the USB drive, the OS will read a lot of metadata (the FAT tables) from the filesystem. FAT32 in particular has a large amount of metadata. Both FAT16 and exFAT have tiny FAT tables. Using FAT16 or exFAT avoids a 10 - 20 second slow start.

The operating system also typically reads the names of all the files in the top-level directory so it can display it in a file browsing window, and that can also take a while.

You can prevent this slow startup by disabling the presentation of the SD card as a USB drive. Turn it off by adding this line to the **settings.toml** file on the **CIRCUITPY**:

```
CIRCUITPY_SDCARD_USB = false
```

You can also disable USB presentation entirely if you are [building your own CircuitPython from scratch](https://adafru.it/Bfu) (https://adafru.it/Bfu). Most people will not do this, but it is mentioned here for completeness. You would add this setting to **mpconfigboard.mk** or elsewhere in the build:

```
CIRCUITPY_SDCARD_USB=0
```

SD Drivers

There are three ways to access SD cards from CircuitPython, two of which offer higher performance. The builtin modules `sdcardio` and `sdio` are part of the CIrcuitPython firmware and are implemented in C. Not all boards and ports implement both. The CircuitPython library, `adafruit_sdcard`, should be avoided when performance matters. It is written in pure Python and is slower.

SD Driver to Chip Mapping

Driver	Boards	Interface
`sdioio`	ESP32-S3, STM32F405, SAMD51	SDIO (requires extra wiring)

`sdcardio`	RP2350, RP2040, nRF52840, SAMD21 Express, SAMD51	SPI only
`adafruit_sdcard`	any SPI board	SPI only

`sdioio` — For boards with a native SDMMC peripheral. It requires wiring all four data lines. The fastest CircuitPython SD reads are then available. Many boards don't provide `sdioio`.

`sdcardio` — Built into CircuitPython firmware; there is nothing to install. On some boards, notably Metro RP2350 and Fruit Jam, the card auto-mounts at boot with no code needed. In most cases, this will be the library to use, but will not be as fast as `sdioio` in throughput. It is SPI only.

`adafruit_sdcard` — Written in pure Python, it is slower than `sdcardio`. Use only when `sdcardio` isn't available in your board's firmware.

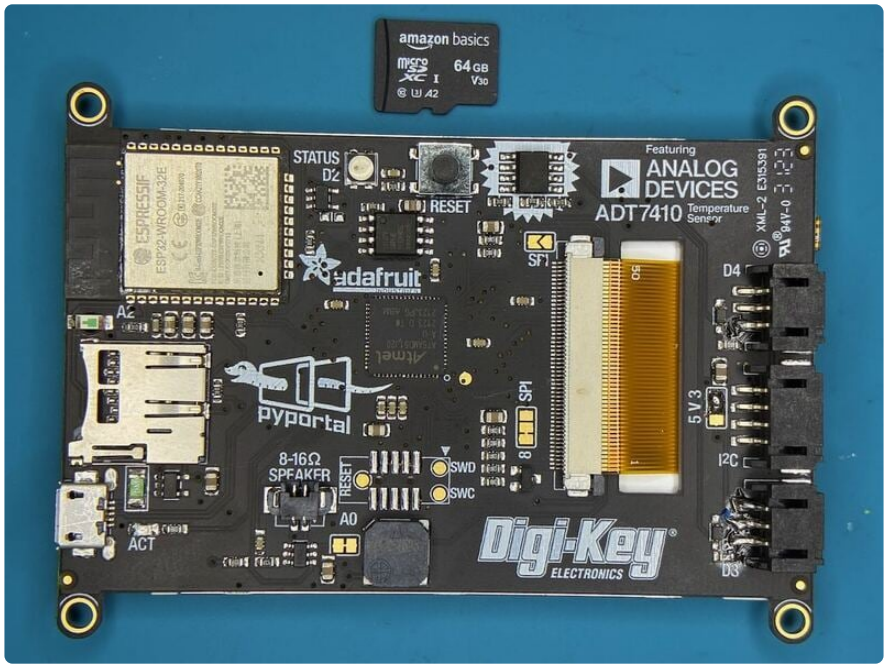
Board Testing

To align expectations with the reality of what can be done with CircuitPython today, here are benchmarks on a few boards in different configurations. These can be compared to the original baseline Metro RP2350 (SPI) / Feather ESP32-S3 (SDIO) boards as well as against each other for relative performance.

These tests use the following:

- PyPortal M4 (http://adafru.it/4116) - Everything is built-in (TFT, Controller, WiFi, SD card slot)
- 3.5" FeatherWing V2 (http://adafru.it/2090) + Feather RP2350 (http://adafru.it/6000)
- Feather RP2350 (http://adafru.it/6000) + Adalogger FeatherWing (http://adafru.it/2922)
- RP2040 Adalogger (http://adafru.it/5980)
- Memento Camera (http://adafru.it/5420) (ESP32-S3)

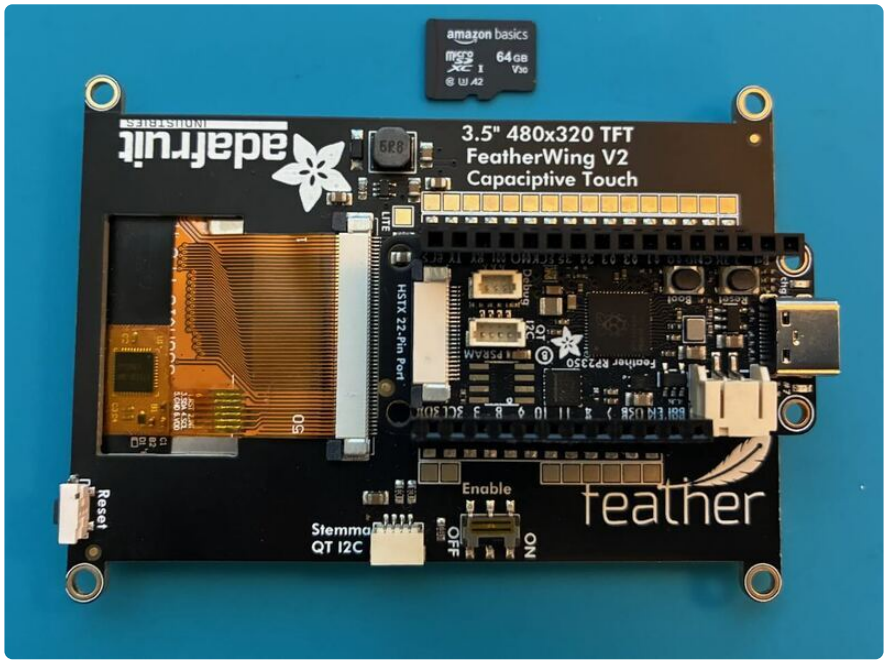
PyPortal M4



The PyPortal M4 has a nice screen, WiFi and an SD card slot. When benchmarking with a 64GB A2/U3 SD card, the results show excellent SPI based performance. It is right in line with the Metro RP2350. While the PyPortal is significantly slower than an SDIO setup for bulk read/write, it wins at logging lines per second. This board did require a 3 second pause between benchmarks to re-stabilize, something other boards did not struggle with.

Board	Bus	Bulk write	Bulk read	Logging
PyPortal M4	SPI @24MHz	2.5 MB/s	2.4 MB/s	2763 l/s
Metro RP2350	SPI @25MHz	2.3 MB/s	2.2 MB/s	2581 l/s
ESP32-S3	4-bit SDIO @40MHz	4.5 MB/s	7.1 MB/s	1489 l/s

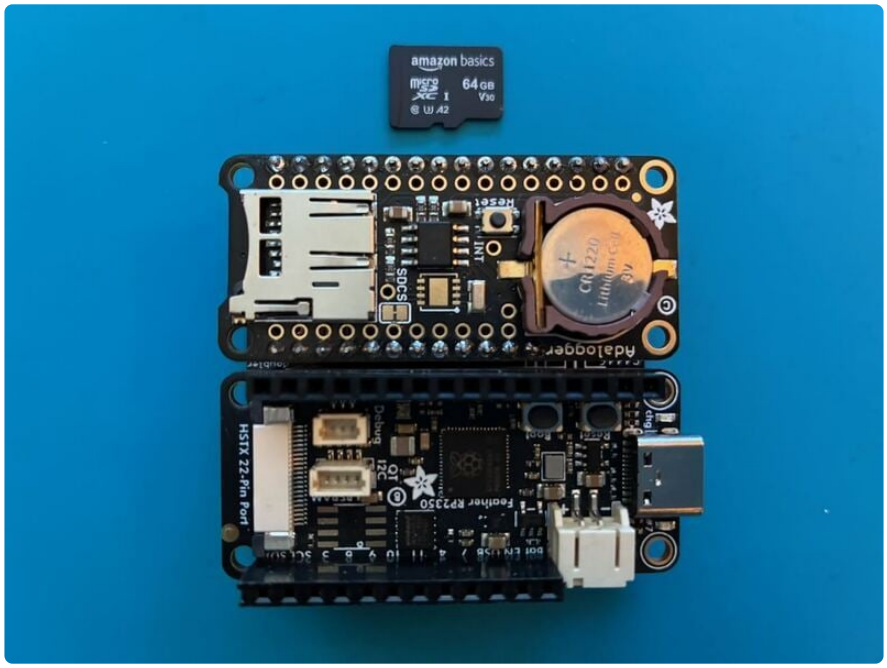
3.5" FeatherWing + Feather RP2350



There is a significant slowdown with the 3.5" FeatherWing V2 display, but it is still near 2 MB/s bulk read/write. It's not as fast as the PyPortal M4.

Board	Bus	Write	Read	Logging
Feather RP2350 +3.5" TFT FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2182 l/s
PyPortal M4	SPI @24MHz	2.5 MB/s	2.4 MB/s	2763 l/s
Metro RP2350	SPI @25MHz	2.3 MB/s	2.2 MB/s	2581 l/s
ESP32-S3	SDIO @40MHz	4.5 MB/s	7.1 MB/s	1489 l/s

Feather RP2350 + Adalogger FeatherWing



The Feather RP2350 + Adalogger FeatherWing setup will always be limited to SPI, unlike the Feather RP2040 Adalogger which has SDIO potential in the future. The logged lines per second roughly doubled versus the RP2040 Adalogger, a notable difference.

Board	Bus	Write	Read	Logging
Feather RP2350 + Adalogger FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2181 l/s
Feather RP2350 + 3.5" TFT FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2182 l/s
PyPortal M4	SPI @24MHz	2.5 MB/s	2.4 MB/s	2763 l/s
Metro RP2350	SPI @25MHz	2.3 MB/s	2.2 MB/s	2581 l/s
ESP32-S3	SDIO @40MHz	4.5 MB/s	7.1 MB/s	1489 l/s

Feather RP2040 Adalogger

The Feather RP2040 Adalogger has bulk throughput just under the 2 MB/s mark. Longer term, CircuitPython may add PIO SDIO support (https://adafru.it/1aEv) as well as asyncio (https://adafru.it/1aEw). Should these be included, it would allow this board to use 4-bit SDIO instead of SPI. The RP2040 Adalogger board could have much faster performance with software updates.

Board	Bus	Write	Read	Logging
Feather RP2040 Adalogger	SPI (built-in)	1.9 MB/s	1.8 MB/s	1065 l/s
Feather RP2350 +3.5" TFT FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2182 l/s
PyPortal M4	SPI @24MHz	2.5 MB/s	2.4 MB/s	2763 l/s
Metro RP2350	SPI @25MHz	2.3 MB/s	2.2 MB/s	2581 l/s
ESP32-S3	SDIO @40MHz	4.5 MB/s	7.1 MB/s	1489 l/s

Memento Camera

The Memento Camera is significantly slower for writes and reads than other configurations. The chip itself is capable, but the card slot is wired for SPI not SDIO, and the frequency is set to 20 MHz to stay within spec. The next speed jump is 26.7 MHz (over spec) as the display is sharing the SPI bus the SD card is on.

Board	Bus	Write	Read	Logging
Memento ESP32-S3	SPI @20MHz	0.8 MB/s	1.1 MB/s	1511 l/s
Feather RP2040 Adalogger	SPI (built-in)	1.9 MB/s	1.8 MB/s	1065 l/s
Feather RP2350 + Adalogger FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2181 l/s
Feather RP2350 +3.5" TFT FeatherWing	SPI @24MHz	1.8 MB/s	1.7 MB/s	2182 l/s
PyPortal M4	SPI @24MHz	2.5 MB/s	2.4 MB/s	2763 l/s

Metro RP2350	SPI @25MHz	2.3 MB/s	2.2 MB/s	2581 I/s
ESP32-S3	SDIO @40MHz	4.5 MB/s	7.1 MB/s	1489 I/s

Benchmark Setup

These are the parameters and the code used in measuring the data for this guide. Jeff Epler's [Benchmarking SD card access (https://adafru.it/1aEt)](https://adafru.it/1aEt) code was used with an identical **code.py** and **mount_sd.py** varied based on board type.

Other considerations:

- All benchmarks were run on CircuitPython 10.3.0-alpha.2
- Freshly formatted exFAT SD card was used.
- Disabled host OS pass-through
  - **settings.toml** : `CIRCUITPY_SDCARD_USB = false`
- Best-of-3 runs
- The file is not closed or flushed after each line
- SD card used - 64 GB A2/U3 SD card (minimum guarantee on card ability)
  - U3 - 30 MB/s
  - A2 - Random read performance: 4000 IOPS
  - A2 - Random write performance: 2000 IOPS
- Clock is fixed on boards with build in SD card slots
- Memento shares the SPI bus with its TFT. Released the display and deselect the **TFT_CS** pin

Metro RP2350 **mount_sd.py**

Nothing needs to be done on the RP2040 and RP2350 boards with a built-in SD card slot. The card is automatically mounted.

```
pass  # /sd is already mounted by firmware at 25 MHz SPI
```

Feather ESP32-S3 **mount_sd.py**

Set up the 40MHz clock and 4-bit SDIO data pins.

```
import board, sdioio, storage

sd = sdioio.SDCard(
    clock=board.SCK,
    command=board.MOSI,
    data=[board.MISO, board.D9, board.D10, board.D11],  # DAT0..DAT3 (4-bit bus)
    frequency=40_000_000,                               # 40 MHz — API max, verifies clean
)
storage.mount(storage.VfsFat(sd), "/sd")
```

PyPortal M4 **mount_sd.py**

```
import board, sdcardio, storage
spi = board.SPI()
sd = sdcardio.SDCard(spi, board.SD_CS, baudrate=24_000_000)
storage.mount(storage.VfsFat(sd), "/sd")
```

Feather RP2350 + 3.5" TFT FeatherWing **mount_sd.py**

```
import board, sdcardio, storage
spi = board.SPI()
sd = sdcardio.SDCard(spi, board.D5, baudrate=24_000_000)  # TFT Wing V2 SD CS = D5
storage.mount(storage.VfsFat(sd), "/sd")
```

Feather RP2350 + Adalogger FeatherWing **mount_sd.py**

```
import board, sdcardio, storage
spi = board.SPI()
sd = sdcardio.SDCard(spi, board.D10, baudrate=24_000_000)  # Adalogger Wing SD CS = D10
storage.mount(storage.VfsFat(sd), "/sd")
```

Memento ESP32-S3 **mount_sd.py**

Disable the TFT before initializing the SD card.

```
import board, sdcardio, storage, digitalio, displayio
displayio.release_displays()              # stop the camera UI driving the bus
tft = digitalio.DigitalInOut(board.TFT_CS)    # park TFT CS high so it releases the bus
tft.direction = digitalio.Direction.OUTPUT
tft.value = True
spi = board.SPI()
sd = sdcardio.SDCard(spi, board.CARD_CS, baudrate=20_000_000)  # 20 MHz to stay in spec
storage.mount(storage.VfsFat(sd), "/sd")
```