class SDCard – secure digital memory card

SD cards are one of the most common small form factor removable storage media. SD cards come in a variety of sizes and phsyical form factors. MMC cards are similar removable storage devices while eMMC devices are electically similar storage devices designed to be embedded into other systems. All three form share a common protocol for communication with their host system and high-level support looks the same for them all. As such in MicroPython they are implemented in a single class called `machine.SDCard` .

Both SD and MMC interfaces support being accessed with a variety of bus widths. When being accessed with a 1-bit wide interface they can be accessed using the SPI protocol. Different MicroPython hardware platforms support different widths and pin configurations but for most platforms there is a standard configuation for any given hardware. In general constructing an `SDCard`` object with without passing any parameters will initialise the interface to the default card slot for the current hardware. The arguments listed below represent the common arguments that might need to be set in order to use either a non-stanard slot or a non-standard pin assignment. The exact subset of arguments suported will vary from platform to platform.

---

class **machine.SDCard**(*slot=1, width=1, cd=None, wp=None, sck=None, miso=None, mosi=None, cs=None*)

This class provides access to SD or MMC storage cards using either a dedicated SD/MMC interface hardware or through an SPI channel. The class implements the block protocol defined by `uos.AbstractBlockDev`. This allows the mounting of an SD card to be as simple as:

```
uos.mount(machine.SDCard(), "/sd")
```

The constrcutor takes the following paramters:

- *slot* selects which of the available interfaces to use. Leaving this unset will select the default interface.
- *width* selects the bus width for the SD/MMC interface.
- *cd* can be used to specify a card-detect pin.
- *wp* can be used to specify a write-protect pin.
- *sck* can be used to specify an SPI clock pin.
- *miso* can be used to specify an SPI miso pin.
- *mosi* can be used to specify an SPI mosi pin.
- *cs* can be used to specify an SPI chip select pin.

Implementation-specific details

Different implementations of the `SDCard` class on different hardware support varying subsets of the options above.

PyBoard

The standard PyBoard has just one slot. No arguments are necessary or supported.

ESP32

The ESP32 provides two channels of SD/MMC hardware and also supports access to SD Cards through either of the two SPI ports that are generally available to the user. As a result the *slot* argument can take a value between 0 and 3, inclusive. Slots 0 and 1 use the built-in SD/MMC hardware while slots 2 and 3 use the SPI ports. Slot 0 supports 1, 4 or 8-bit wide access while slot 1 supports 1 or 4-bit access; the SPI slots only support 1-bit access.

> **Note**
>
> Slot 0 is used to communicate with on-board flash memory on most ESP32 modules and so will be unavailable to the user.

> **Note**
>
> Most ESP32 modules that provide an SD card slot using the dedicated hardware only wire up 1 data pin, so the default value for *width* is 1.

The pins used by the dedicated SD/MMC hardware are fixed. The pins used by the SPI hardware can be reassigned.

Note

If any of the SPI signals are remapped then all of the SPI signals will pass through a GPIO multiplexer unit which can limit the performance of high frequency signals. Since the normal operating speed for SD cards is 40MHz this can cause problems on some cards.

The default (and preferred) pin assignment are as follows:

Slot	**0**	**1**	**2**	**3**
Signal	**Pin**	**Pin**	**Pin**	**Pin**
sck	6	14	18	14
cmd	11	15		
cs			5	15
miso			19	12
mosi			23	13
D0	7	2		
D1	8	4		
D2	9	12		
D3	10	13		
D4	16			
D5	17			
D6	5			
D7	18			

cc3200

You can set the pins used for SPI access by passing a tuple as the *pins* argument.

Note: The current cc3200 SD card implementation names the this class `machine.SD` rather than `machine.SDCard` .