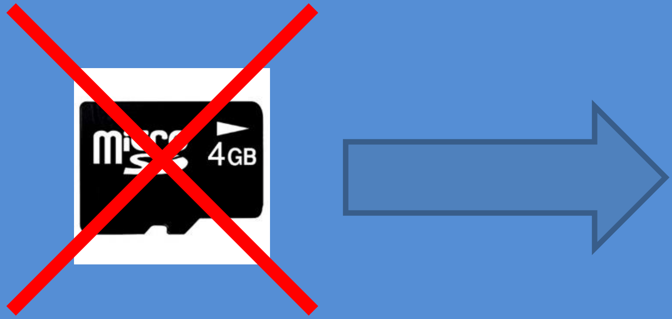
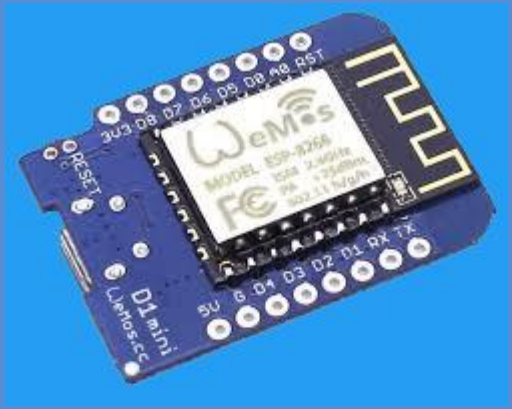


Serial Peripheral Interface Flash File System	3M Byte SPIFFS Flash Storage
Program Memory	1M Byte Program

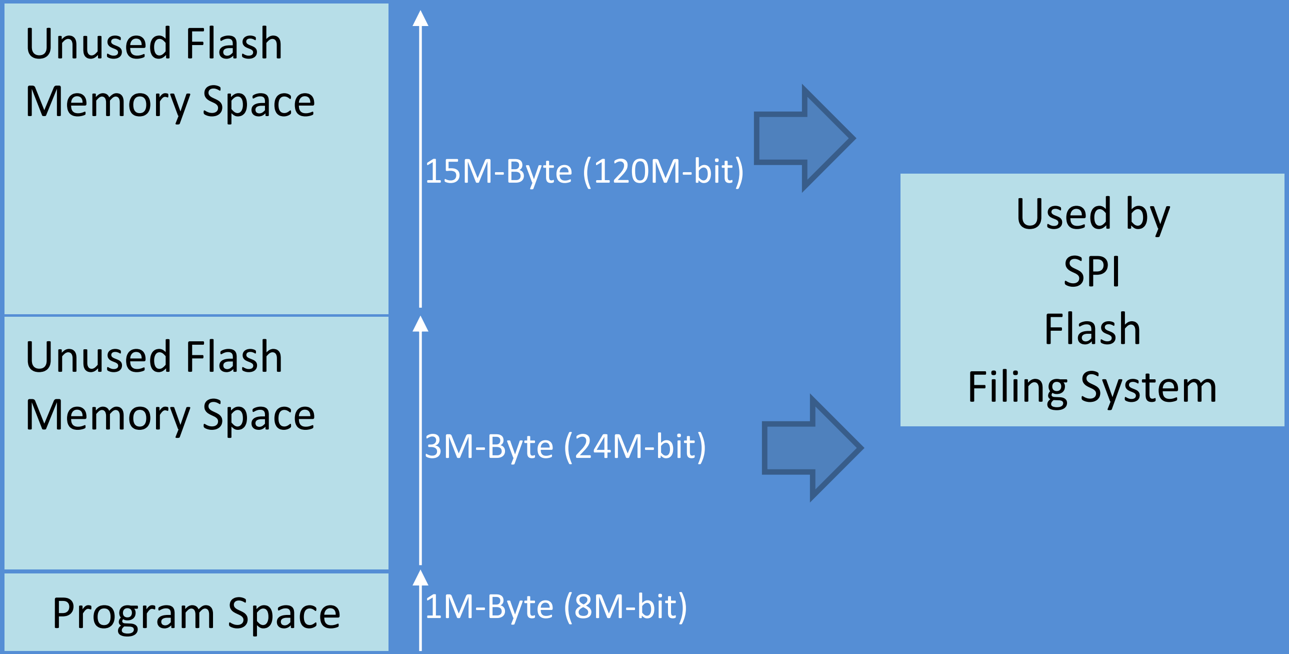
ESP8266 and using SPIFFS in-place of SD-Card storage



Filing System Implemented in Flash Storage

ESP8266
PRO

ESP8266



Unused Flash Memory Space

15M-Byte (120M-bit)

Unused Flash Memory Space

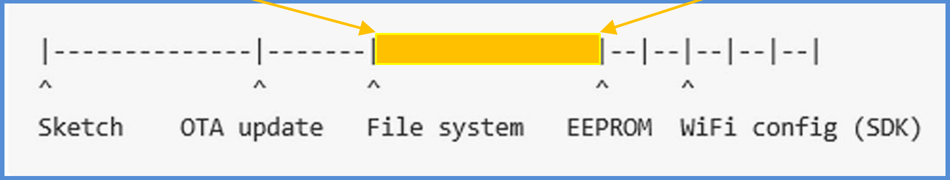
3M-Byte (24M-bit)

Program Space

1M-Byte (8M-bit)

Used by SPI Flash Filing System

SPIFFS – the SPI bus Flash memory File System



```
|--------------|-------|               |--|--|--|--|--|
^              ^       ^               ^     ^
Sketch    OTA update   File system   EEPROM  WiFi config (SDK)
```

Filing System Comparison SD v SPIFFS

Filing System Objects

SD-CARD

- begin
- exists
- open
- remove
- rmdir
- mkdir

SPIFFS

- begin
- exists
- open
- remove
- exists
- format
- openDir
- rename
- info

File Objects

SD-CARD

- seek
- position
- size
- close
- print
- println
- read
- write
- available
- flush
- peek
- isDirectory
- openNextFile
- rewindDirectory

SPIFFS

- seek
- position
- size
- close
- print
- println
- read
- write
- name

See here for SPIFFS Documentation

Filing System Comparison SD v SPIFFS

```
// SD Card Write then Read to a file example
#include <SD.h>;
void setup() {
  SD.begin(D8);
}

void loop() {
  char filename [] = "datalog.txt";
  File myDataFile = SD.open(filename, FILE_WRITE);
  myDataFile.println("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
  myDataFile.println(3.141592654);
  myDataFile.close();

  myDataFile = SD.open(filename, FILE_READ);
  while (myDataFile.available()) {
    Serial.write(myDataFile.read());
   }
   myDataFile.close();
}
```

Program output/results (File is 34-bytes)

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
3.141592654
```

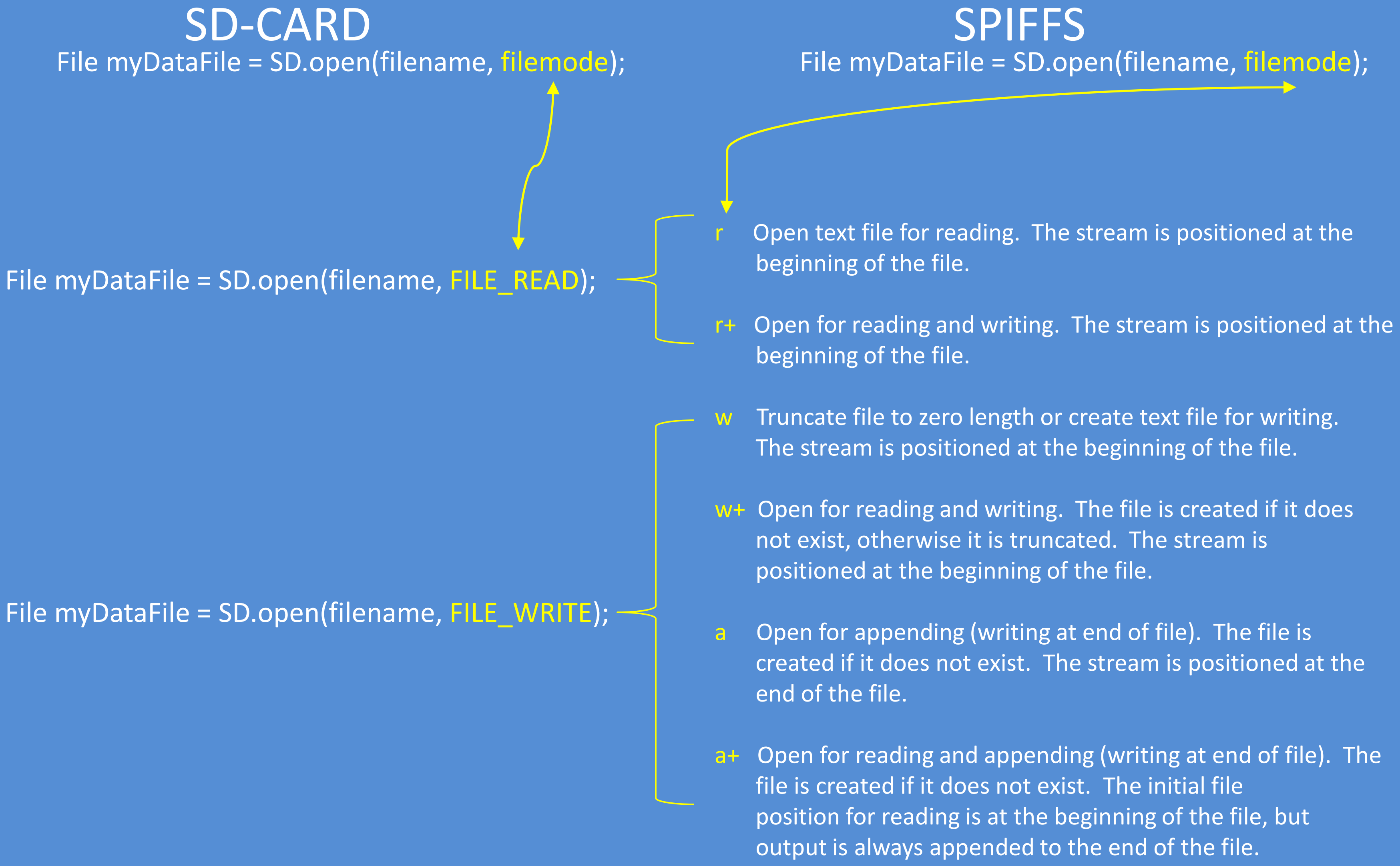
```
// SPIFFS Write then Read to a file example
#include <FS.h>
void setup() {
  SPIFFS.begin();
}

void loop() {
  char filename [] = "datalog.txt";
  File myDataFile = SPIFFS.open(filename, "a");
  myDataFile.println("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
  myDataFile.println(3.141592654);
  myDataFile.close();

  myDataFile = SPIFFS.open(filename, "r");
 while (myDataFile.available()) {
    Serial.write(myDataFile.read());
  }
  myDataFile.close();
}
```

Program output/results (File is 34-bytes)

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
3.141592654
```

Filing System Comparison SD v SPIFFS

Implementation Examples/Differences



SD-CARD

File myDataFile = SD.open(filename, filemode);

File myDataFile = SD.open(filename, FILE_READ);

File myDataFile = SD.open(filename, FILE_WRITE);

SPIFFS

File myDataFile = SD.open(filename, filemode);

r Open text file for reading. The stream is positioned at the beginning of the file.

r+ Open for reading and writing. The stream is positioned at the beginning of the file.

w Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

w+ Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

a Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.

a+ Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

Filing System Comparison SD v SPIFFS

SPIFFS Things to Note

- SPIFFS is not a FAT filing system

- Filenames are ‘absolute’ and contain the full address
  - e.g. “/DATA/filename.dat”

- Directories are NOT implemented, however they are simulated
  - SD-Card system allows a Directory containing files, SPIFFS does not
  - E.g. an SD-Card directory container file called [DATA] can exist and contain a file called ‘filename.dat’
  - SPIFFS uses absolute filenames and is implemented as myFile.OPEN(“/DATA/filename.dat”,”r”)

- Using Dir dir = SPIFFS.openDir("/DATA"); effectively appends a path filter to files
  - Using the example above, File myFile = dir.openFile("r"); then myFile.openFIle(“filename.dat”,”r”); actually opens the file “/DATA/filename.dat”

See here for SPIFFS Documentation

Filing System Comparison SD v SPIFFS

SPIFFS Things to Note

- Filenames path are not case-sensitive and limited to 32 chars. One character is reserved for C string termination ('\0'), leaving 31 usable characters for filenames (and path)
- SPIFFS does not warn you when the 31-character filename limit is exceeded
- Flash memory will fail after ~10,000 writes! So keeping writing to a minimum

  e.g. read many, write few!

  Example: If data logging temp and humidity readings every 10-mins (tt.t,hh.h,dd/mm/yy,hh,mm 25-bytes) then the 3MB flash would be full when 3.1024.1024/25=125829 readings had been recorded. At 10-mins per reading that's 2.4 years when the data could be overwritten again ~10,000 times, which is more than enough life (2.4x10,000 years) for most applications!
- Overall SPIFFS is more versatile than the SD filing system

See here for SPIFFS Documentation

GitHub Example Files

- Example-1   Writing and Reading from SD-Card file system
- Example-2   Writing and Reading from SPIFFS file system
- Example-3   Combined SD and SPIFFS Writing and Reading

See here for GitHub SPI File Repository