# Table of Contents

## Overview

## Pinouts
- Power Pins
- SPI Logic Pins

## Arduino
- Wiring
- Library Installation
- Example Code

## Arduino Docs

## Downloads
- Files
- Schematic and Fab Print
Overview

For many modern and powerful chips like the RP2040, ESP32, RT10xx and STM32 series microcontrollers, designers can save money and reduce the number of chip options by not including building in the Flash memory used to store code and resources. Instead, an external QSPI Flash memory chip is wired up that can provide up to 16 Megabytes (a.k.a 128 Megabits) of memory. It's not as fast as if it were on the microcontroller internal bus but with Quad SPI I/O and some clever caching by the chip designer, it's pretty effective!

To make prototyping and designing with QSPI flash a little easier, we designed these breakouts that convert the wide 8-SOIC packages to a cute 0.3" wide DIP. We find these handy when testing out different flash sizes, or if we just want to add more storage memory to a project.
There are three variants for these breakouts:

- **W25Q16JVSSIQ**, a 3.3V power and logic 16-Megabit / 2 Mega-byte chip.
- **W25Q64JVSSIQ**, a 3.3V power and logic 64-Megabit / 8 Mega-byte chip.
- **W25Q128JVSSIQ**, a 3.3V power and logic 128-Megabit / 16 Mega-byte chip.

Note that the Q at the end means that the Quad Enable bit is permanently set in the status register. If you're using this in QSPI mode, then it'll work right out of the box. If you're using this in SPI mode, the 'hold' and 'write protect' pins don't do anything, so just connect them to 3.3V.

**In Arduino**, we have the Adafruit_SPIFlash library that can be used to interface with this chip - but don't forget it's only good for 3.3V logic and power!

In CircuitPython, this chip has a TOML definition file so you could use it in a board definition.

Comes with a bit of header that can be used to solder in and plug into a breadboard or to replace an existing socketed SPI flash DIP chip.
Pinouts

Power Pins

- **3V** - this is the 3V power input pin for the flash chip. These chips can only be used with 3.3V power and logic!
- **G** - common ground for power and logic.

Only use these breakouts with 3.3V logic and power!

SPI Logic Pins

- **CS** - Chip Select input. Enables and disables device operation. Set low to select the device to begin a SPI transaction. Marked on the front of the board with a white square.
- **IO0** - Data Input/Data Input Output 0 (MOSI). Used for serially writing instructions, addresses or data to the device on the rising edge of CLK.
- **IO1** - Data Output/Data Input Output 1 (MISO). Used for reading data or status from the device on the falling edge of CLK.
- **IO2** - Write Protect input/Data Input Output 2. Used to prevent the status register from being written to. This pin is active low.
- **IO3** - Hold Input/Data Input Output 3. Allows the device to be paused while it is actively selected.
- **CLK** - Serial Clock Input (SCK). Provides the timing for serial input and output operations.
Arduino

Using a QSPI DIP breakout board with Arduino involves wiring up the QSPI chip to your Arduino-compatible microcontroller, installing the Adafruit_SPIFlash() library and running the provided example code.

Wiring

Wire as shown for a 3.3V board like a Feather. These breakouts are only good for 3.3V logic and power! Do not use them with a 5V board, like an Arduino Uno!

Here is an Adafruit Feather M4 wired up to the QSPI DIP breakout board on a solderless breadboard:

Only use these breakouts with 3.3V logic and power!

Library Installation

You can install the Adafruit SPIFlash library for Arduino using the Library Manager in the Arduino IDE.

Click the Manage Libraries ... menu item, search for Adafruit SPIFlash, and select the Adafruit SPIFlash library:
If asked about dependencies, click "Install all".

If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

/*
 * SD card read/write
 * This example shows how to read and write data to and from an SD card file
 * The circuit:
 * * SD card attached to SPI bus as follows:
 * ** MOSI - pin 11
 * ** MISO - pin 12
 * ** CLK - pin 13
 * created Nov 2010
 * by David A. Mellis
 * modified 9 Apr 2012
 * by Tom Igoe
 * This example code is in the public domain.
 */
*/
#include <SPI.h>
#include "SdFat.h"
#include "Adafruit_SPIFlash.h"
// for flashTransport definition
#include "flash_config.h"
Adafruit_SPIFlash flash(&flashTransport);
// file system object from SdFat
FatVolume fatfs;
File32 myFile;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    delay(10); // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Initializing Filesystem on external flash...");
  // Init external flash
  flash.begin();
  // Open file system on the flash
  if (!fatfs.begin(&flash)) {
    Serial.println("Error: filesystem is not existed. Please try SdFat_format "
    "example to make one.");
    while (1) {
      yield();
      delay(1);
    }
  }
  Serial.println("initialization done.");
  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = fatfs.open("test.txt", FILE_WRITE);
  // if the file opened okay, write to it:
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.");
    // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  // re-open the file for reading:
  myFile = fatfs.open("test.txt");
  if (myFile) {
    Serial.println("test.txt");
    // read from the file until there's nothing else in it:
    while (myFile.available()) {
      Serial.write(myFile.read());
    }
    // close the file:

myFile.close();
} else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
}

void loop() {
    // nothing happens after setup
}

Upload the read/write sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You'll see the initialization of the QSPI flash module over SPI. Then, a `text.txt` file is written to the flash chip. Finally, the contents of the `text.txt` file are read back and printed to the Serial Monitor.

---

**Arduino Docs**

[Arduino Docs](https://www.arduino.cc)

**Downloads**

**Files**

- [Flash Module Datasheet](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
- [W25Q16 Technical Documentation](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
- [W25Q64 Technical Documentation](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
- [W25Q128 Technical Documentation](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
- [EagleCAD PCB files on GitHub](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
- [Fritzing object in the Adafruit Fritzing Library](https://www.arduino.cc/en/Main/Products/FlashMemoryDatasheets)
Schematic and Fab Print

These apply to all versions of the board.