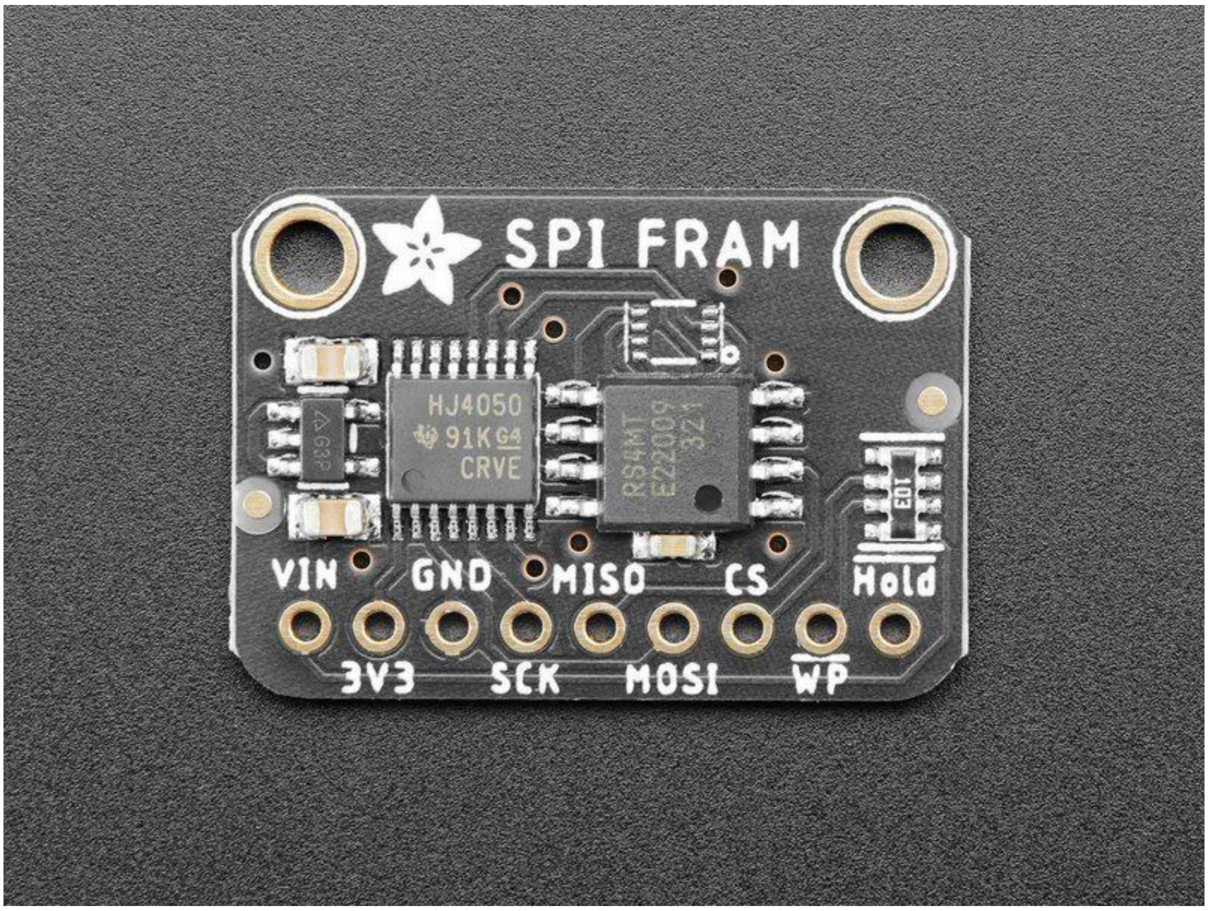
Adafruit SPI FRAM Breakout - 2 or 4 Mbit

Created by Kattni Rembor



https://learn.adafruit.com/spi-fram-2mbit-4mbit

Last updated on 2021-11-15 08:08:55 PM EST

Table of Contents

Overview 3

Pinouts 5

- Power Pins 6
- SPI Logic pins: 6

Arduino Test 6

- Arduino Wiring 7
- Download Adafruit_FRAM_SPI 8
- Load Demo 9
- Library Reference 10
- Hardware vs Software SPI 10
- Begin 11
- Writing 11
- Block Protection 11

Arduino Docs 12

CircuitPython 12

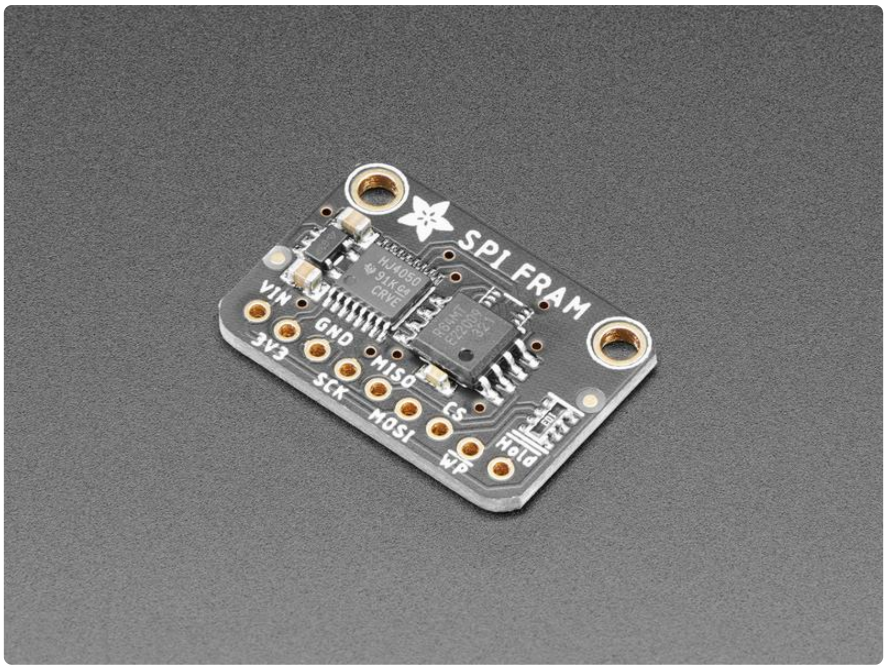
- CircuitPython Microcontroller Wiring 12
- CircuitPython Installation of FRAM Library 14
- CircuitPython Usage 15
- Full Example Code 16

Python Docs 16

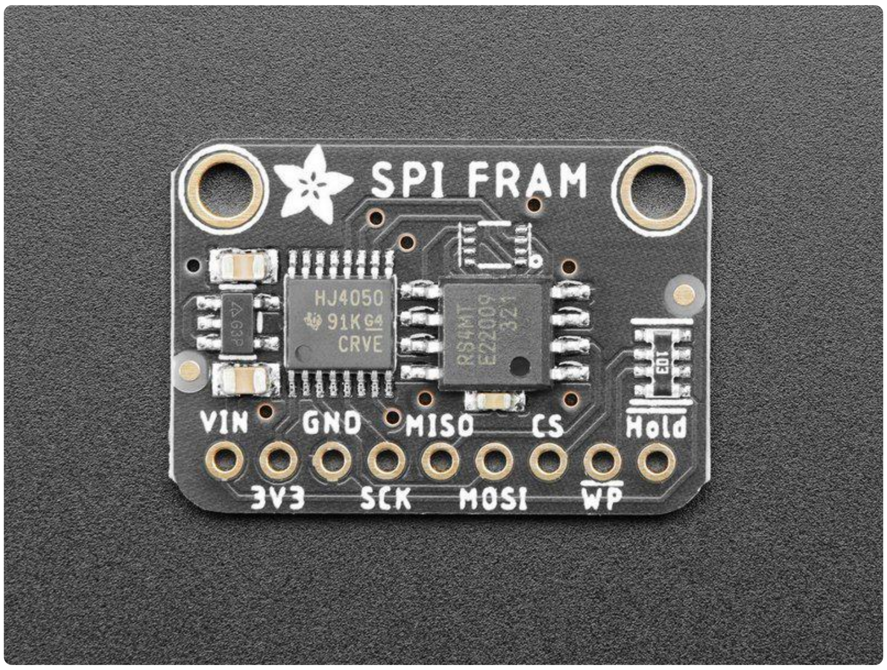
Downloads 17

- Datasheets & Files 17
- Schematic 17
- Fab Print 18

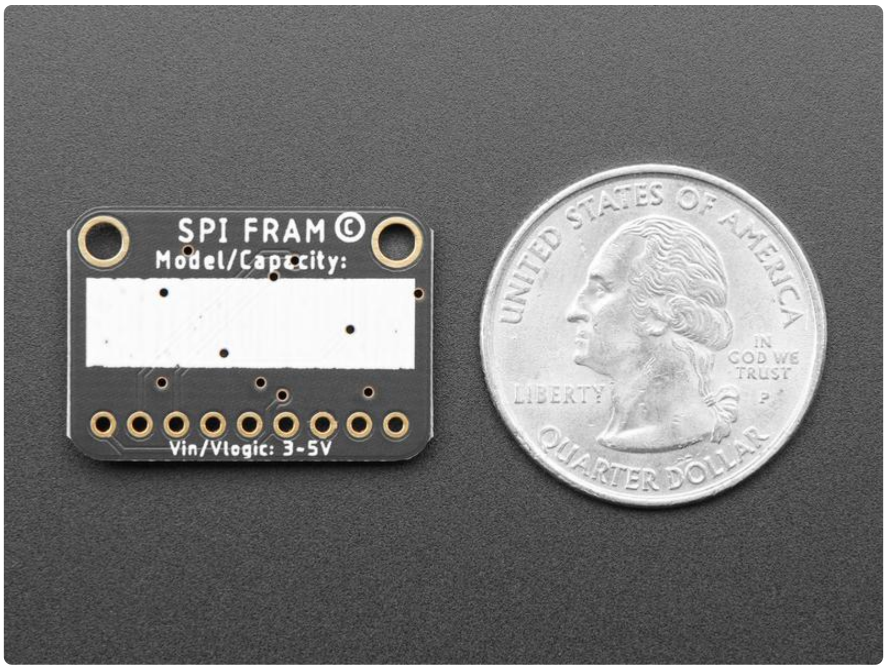
Overview



FRAM, or Ferroelectric Ram, (https://adafru.it/NdQ) is the coolest new data storage method that all the fashion magazines are talking about. Oh wait, no that's quilted handbags. But FRAM is pretty damn cool too! It's similar to Dynamic random-access memory, (https://adafru.it/NdR) only with a ferroelectric layer instead of a dielectric layer. This gives it stable handling (the bytes you write are non-volatile) with dynamic responsiveness (you can write them very fast!)

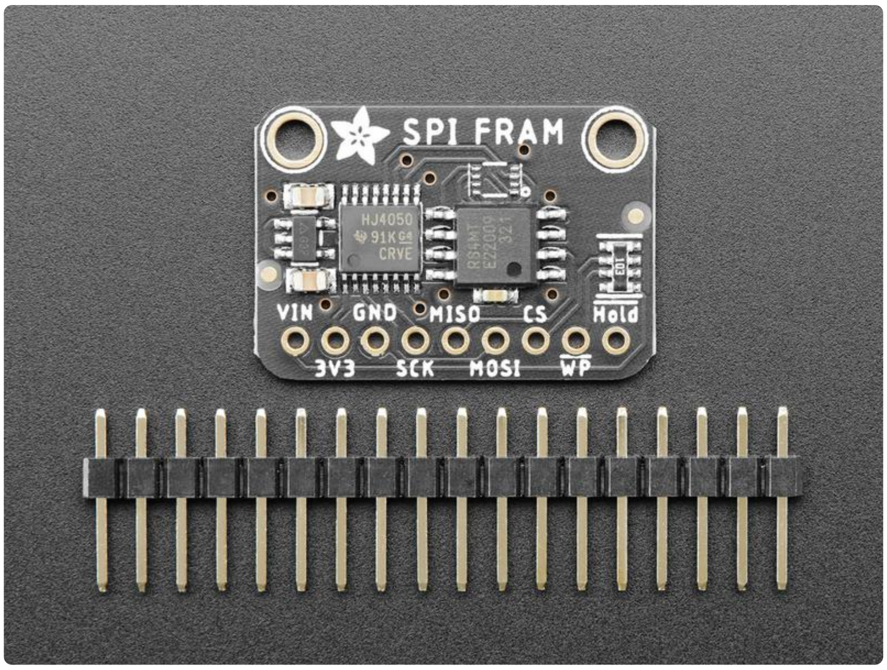


Now, with our SPI FRAM breakout board you can add some FRAM storage to your next DIY project. FRAM allows for a lower power usage and a faster write performance. It's excellent for low-power or inconsistent-power datalogging or data buffering where you want to stream data fast while also keeping the data when there's no power. Unlike Flash or EEPROM there's no pages to worry about. Each byte can be read/written 10,000,000,000,000 times so you don't have to worry too much about wear leveling.

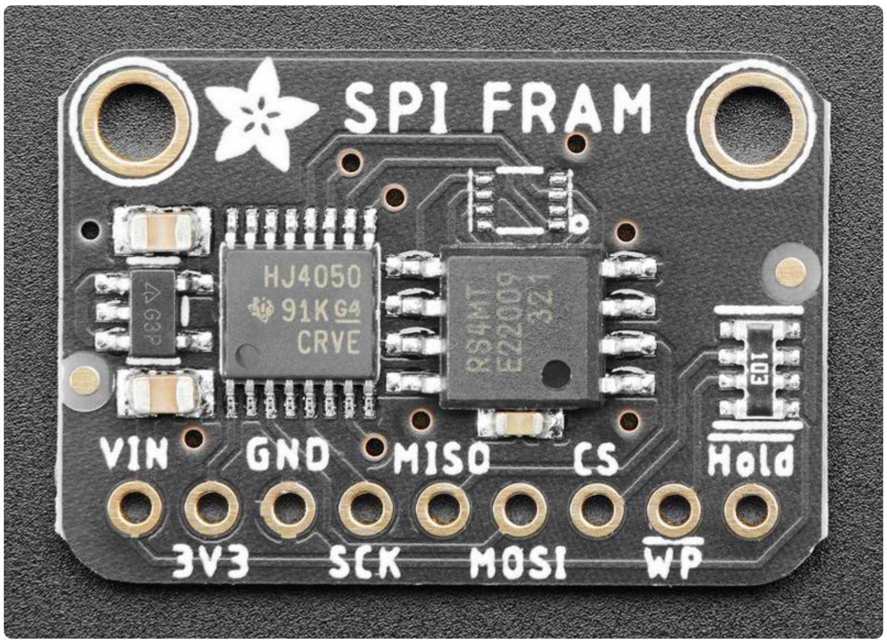


These particular FRAM chips have 2 or 4 Megabits (256 or 512 KBytes) of storage, interfaces using SPI, and can run at up to 40 MHz clock rates. Each byte can be read and written instantaneously (like SRAM) but will keep the memory for 95 years at room temperature.

We put this handy marvel onto a breakout board with 3.3V logic level shifting and regulator so that you can use this chip with either 3V or 5V power and logic. It comes in a breadboard-friendly breakout and bit of standard 0.1" header.



Pinouts



The FRAM is the chonky chip to the right of the center of the board. Along the bottom we have power and interface pins.

The set of 8 tiny pads above the chip are for a different form-factor of RAM, it's normal for there not to be a chip there.

If you have the 2MBit version, you'll see RS2MT on the top of the chip. If you have the 4MBit version, you'll see RS4MT, and the chip will be a little wider.

Power Pins

- VCC - this is the power pin. Since the chip uses 3-5VDC you should pick whatever the logic voltage you're using. For most Arduino's that's 5V.
- 3v3 - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- GND - common ground for power and logic

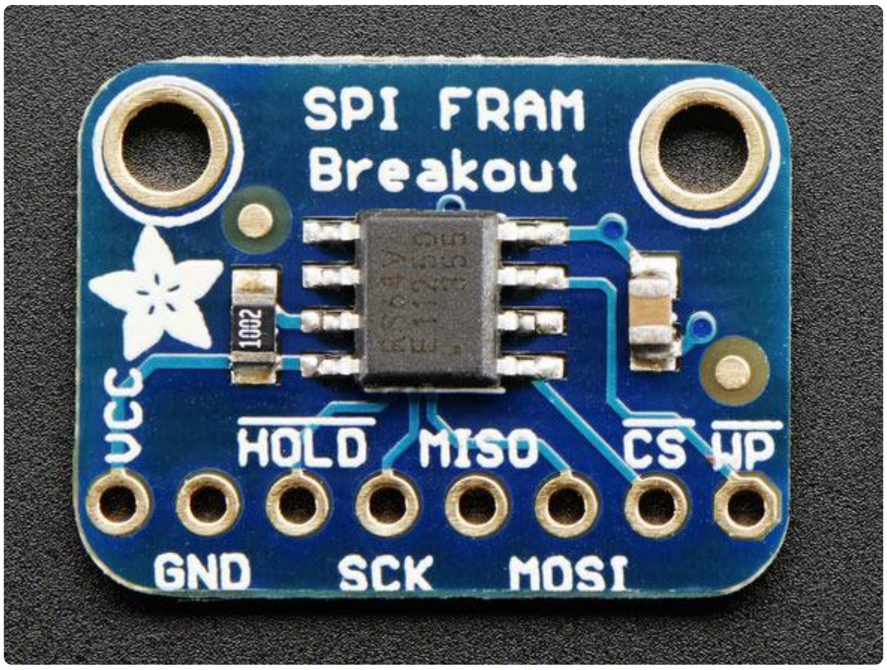
SPI Logic pins:

All pins are 3-5V compliant and use whatever logic level is on VCC

- SCK - This is the SPI clock pin, its an input to the chip
- MISO - this is the Microcontroller In Serial Out pin, for data sent from the FRAM to your processor
- MOSI - this is the Microcontroller Out Serial In pin, for data sent from your processor to the FRAM
- CS - this is the chip select pin, drop it low to start an SPI transaction. Its an input to the chip
- WP - Write Protect pin. This is used to write protect the status register only! This pin does not directly affect write protection for the entire chip. Instead, it protects the block-protect register which is configured however you want (sometimes only half the FRAM is protected)
- HOLD - this is a 'wait' pin for the SPI bus. When pulled low, it puts the SPI bus on hold. This is different than the CS pin because it doesnt stop the current transaction. Its good if you want to talk to other SPI devices and stream data back and forth without stopping and starting transactions.

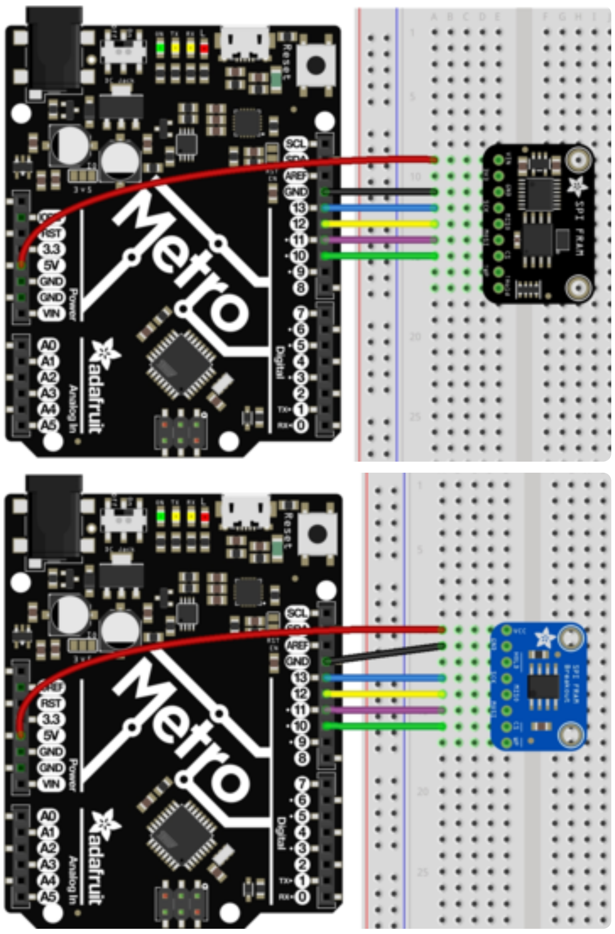
Arduino Test

You can use the same wiring and test code for our older non-level-shifted breakout, or our newer versions which have a regulator and level shifting circuitry!



Arduino Wiring

You can easily wire this breakout to any microcontroller, we'll be using a Metro

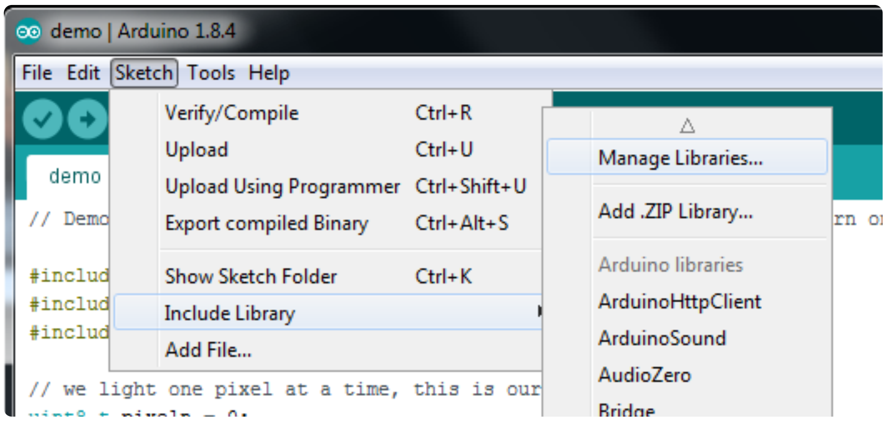


- Connect Vcc to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect GND to common power/ data ground
- Connect the SCK pin to the SPI clock pin on your Arduino. We'll be using Digital #13 which is also the hardware SPI pin on an Uno
- Connect the MISO pin to the SPI MISO pin on your Arduino. We'll be using Digital #12 which is also the hardware SPI pin on an Uno.
- Connect the MOSI pin to the SPI MOSI pin on your Arduino. We'll be using Digital #11 which is also the hardware SPI pin on an Uno.
- Connect the CS pin to the SPI CS pin on your Arduino. We'll be using Digital #10 but any pin can be used later

Download Adafruit_FRAM_SPI

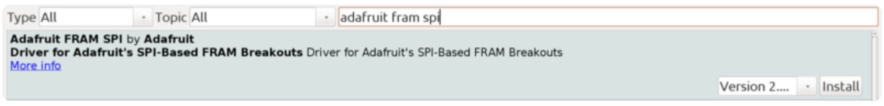
To begin reading and writing data, you will need to download Adafruit_FRAM_SPI (https://adafru.it/du5)from the Arduino Library Manager.

Open up the Arduino Library Manager:



```
// Demo
#includ
#includ
#includ
// we light one pixel at a time, this is our
```

Search for the Adafruit FRAM SPI library and install it


Type All
Topic All
adafruit fram spi
Adafruit FRAM SPI by Adafruit
Driver for Adafruit's SPI-Based FRAM Breakouts Driver for Adafruit's SPI-Based FRAM Breakouts
More info
Version 2....
Install

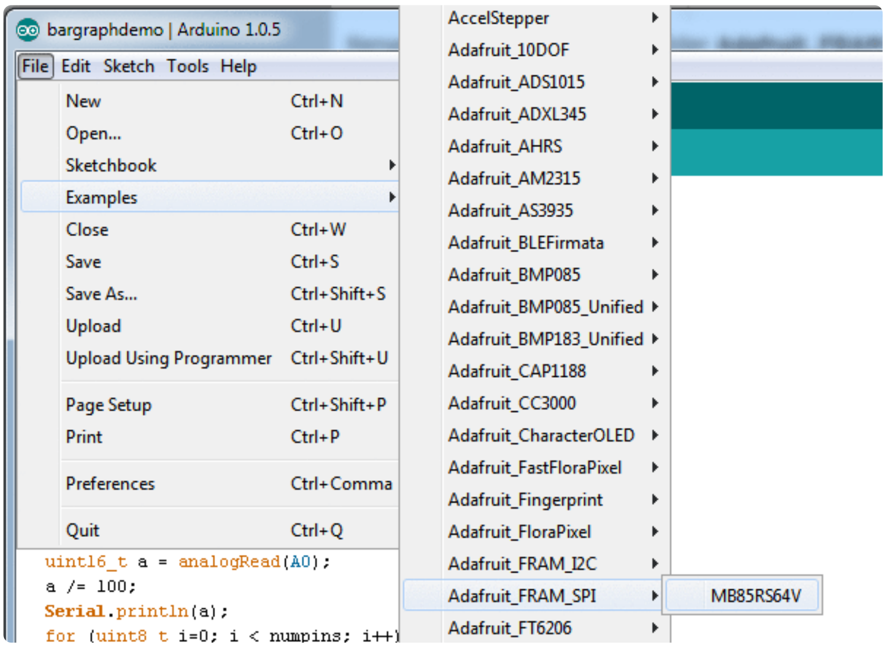
Rename the uncompressed folder Adafruit_FRAM_SPI and check that the Adafruit_FRAM_SPI folder contains Adafruit_FRAM_SPI.cpp and Adafruit_FRAM_SPI.h

Place the Adafruit_FRAM_SPI library folder your arduinosketchfolder/libraries/ folder. You may need to create the libraries subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

Load Demo

Open up File->Examples->Adafruit_FRAM_SPI->MB85RS64V and upload to your Arduino wired up to the sensor

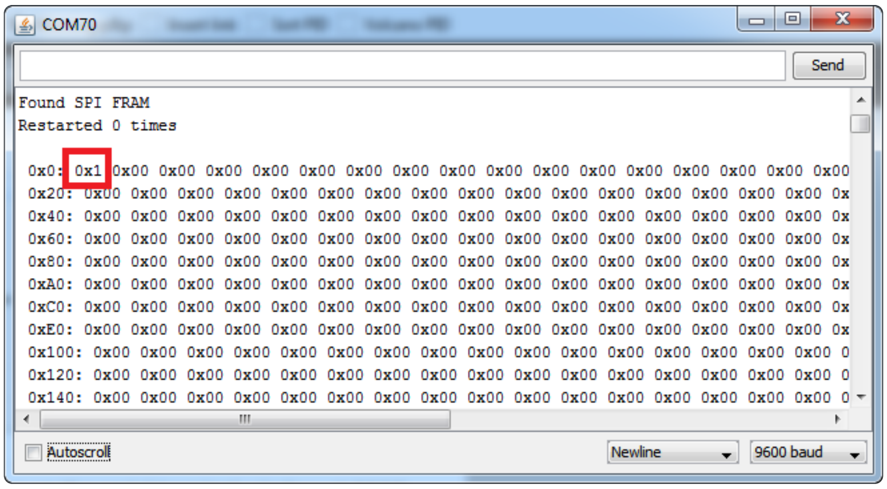


```
uint16_t a = analogRead(A0);
a /= 100;
Serial.println(a);
for (uint8_t i=0; i < numpins; i++)
```

Thats it! Now open up the serial terminal window at 9600 speed to begin the test.

The test is fairly simple - It first verifies that the chip has been found. Then it reads the value written to location #0 in the memory, prints that out and write that value + 1 back to location #0. This acts like a restart-meter: every time the board is reset the value goes up one so you can keep track of how many times its been restarted.

Afterwards, the Arduino prints out the value in every location (all 8KB!)



```
Found SPI FRAM
Restarted 0 times

 0x0: 0x1 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
 0x20: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0x40: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0x60: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0x80: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0xA0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0xC0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0xE0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
 0x100: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0
 0x120: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0
 0x140: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0
```

Library Reference

The library we have is simple and easy to use

Hardware vs Software SPI

You can create the FRAM object using software-SPI (each pin can be any I/O) with

> Adafruit_FRAM_SPI fram = Adafruit_FRAM_SPI(FRAM_SCK, FRAM_MISO, FRAM_MOSI, FRAM_CS);

or use hardware SPI

> Adafruit_FRAM_SPI fram = Adafruit_FRAM_SPI(FRAM_CS);

which means the other 3 pins are the hardware SPI defined pins for your chip. Check the SPI Reference page for details on which pins are which for your Arduino! (https://adafru.it/d5h)

Hardware SPI is faster (the chip can handle up to 20MHz), but you have to use fixed pins. Software SPI is not as fast (maybe 1MHz max on an UNO), but you can switch pins around.

Begin

You can initialize the SPI interface and chip with begin()

```
fram.begin()
```

It will return true or false depending on whether a valid FRAM chip was found

For the 4Mbit version, you should change this to:

```
fram.begin(3)
```

Writing

Then to write a value, call

```
fram.writeEnable(true);
fram.write8(address, byte-value);
fram.writeEnable(false);
```

to write an 8-bit value to the address location
Later on of course you can also read with

```
fram.read8(address);
```

which returns a byte reading. For writing, you must enable writing before you send data to the chip, its for safety! However you can write as much as you want between the writeEnable calls

Block Protection

We dont cover how to protect subsections of the FRAM chip. It's covered a bit more inside the Datasheet.
For advanced users, we have two functions to set/get the Status Register. IF you want to set the status register dont forget that WP must be logical high!

```
uint8_t getStatusRegister();
setStatusRegister(uint8_t value);
```

Arduino Docs

Arduino Docs (https://adafru.it/Nfg)

CircuitPython

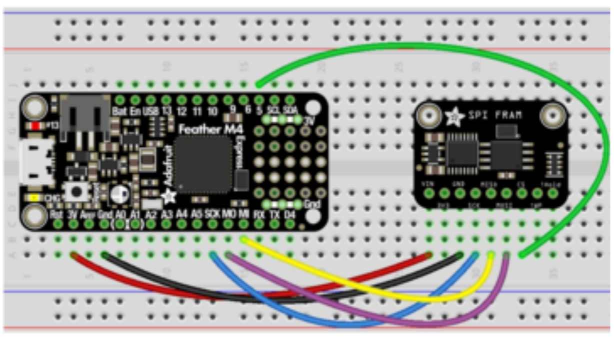
You can use the same wiring and test code for our older non-level-shifted breakout, or our newer versions which have a regulator and level shifting circuitry!

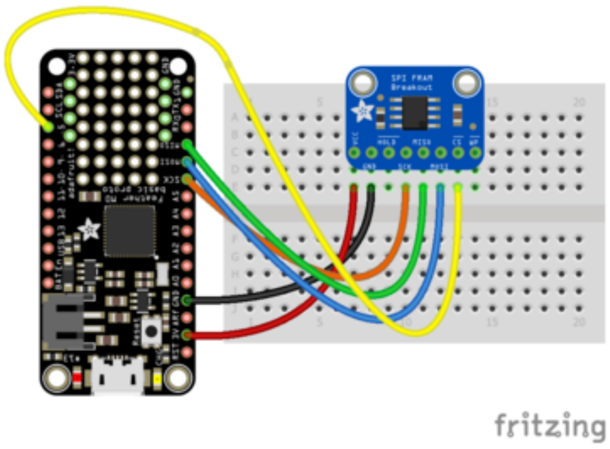
It's easy to use the SPI FRAM Breakout with Python or CircuitPython and the Adafruit CircuitPython FRAM (https://adafru.it/Dhi) module. This module allows you to easily write Python code that reads the humidity, temperature, pressure, and more from the sensor.

CircuitPython Microcontroller Wiring

First we'll wire up a SPI FRAM Breakout to a microcontroller.

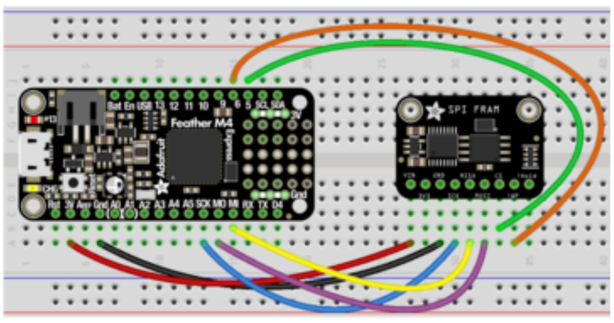
Here is an example of wiring the breakout to a Feather M0 Basic or a Feather M4:



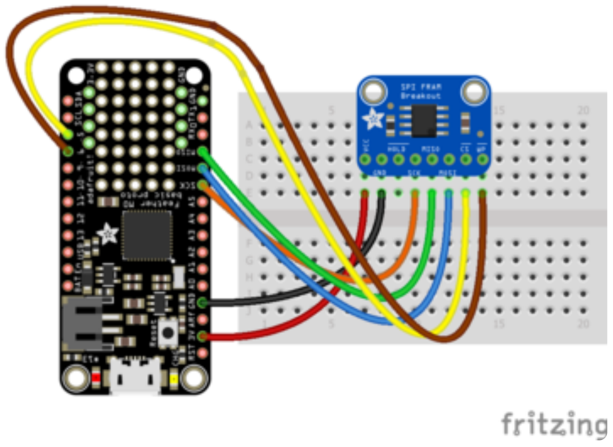


- Board 3V to sensor VIN
- Board GND to sensor GND
- Board SCK to sensor SCK
- Board MOSI to sensor MOSI
- Board MISO to sensor MISO
- Board D5 to sensor CS

If you'd like to use the hardware write protection, connect another GPIO to the sensor's WP pad, like so:



- Board 3V to sensor VIN
- Board GND to sensor GND
- Board SCK to sensor SCK
- Board MOSI to sensor MOSI
- Board MISO to sensor MISO
- Board D5 to sensor CS
- Board D6 to sensor WP



The CircuitPython library takes advantage of the software level write protection on the SPI FRAM chip, so using the hardware write protection isn't necessary. However, the hardware write protection could be useful with an external source of control, like a separate microcontroller.

CircuitPython Installation of FRAM Library

You'll need to install the Adafruit CircuitPython FRAM (https://adafru.it/Dhi) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- adafruit_fram.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder or root filesystem has the adafrui t_fram.mpy, and adafruit_bus_device files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython `>>>` prompt.

CircuitPython Usage

To demonstrate the usage of the breakout we'll initialize it, write data to the FRAM, and read that data from the board's Python REPL.

Run the following code to import the necessary modules and initialize the SPI connection with the breakout:

```
import board
import busio
import digitalio
import adafruit_fram
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(board.D5)
fram = adafruit_fram.FRAM_SPI(spi, cs)
```

Or, if you're using the hardware write protection:

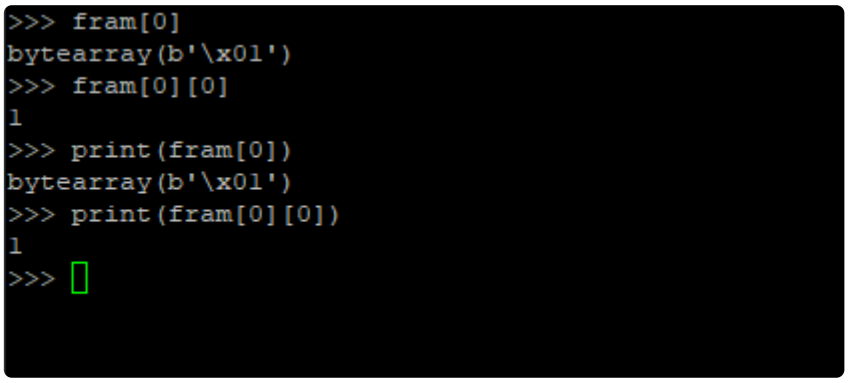
```
import board
import busio
import digitalio
import adafruit_fram
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(board.D5)
wp = digitalio.DigitalInOut(board.D6)
fram = adafruit_fram.FRAM_SPI(spi, cs, wp_pin=wp)
```

Now you can write or read to any address locations:

```
fram[0] = 1

fram[0]
```

Reading the FRAM returns a bytearray. To get a "raw" value, use the index of the value's location. Some various ways to get values are as such:



```
>>> fram[0]
bytearray(b'\x01')
>>> fram[0][0]
1
>>> print(fram[0])
bytearray(b'\x01')
>>> print(fram[0][0])
1
>>>
```

Full Example Code

```python
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

## Simple Example For CircuitPython/Python SPI FRAM Library

import board
import busio
import digitalio
import adafruit_fram

## Create a FRAM object.
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
cs = digitalio.DigitalInOut(board.D5)
fram = adafruit_fram.FRAM_SPI(spi, cs)

## Write a single-byte value to register address '0'

fram[0] = 1

## Read that byte to ensure a proper write.
## Note: 'read()' returns a bytearray

print(fram[0])

## Or write a sequential value, then read the values back.
## Note: 'read()' returns a bytearray. It also allocates
##       a buffer the size of 'length', which may cause
##       problems on memory-constrained platforms.

# values = list(range(100)) # or bytearray or tuple
# fram[0] = values
# print(fram[0:99])
```

Python Docs

Python Docs (https://adafru.it/Dhk)

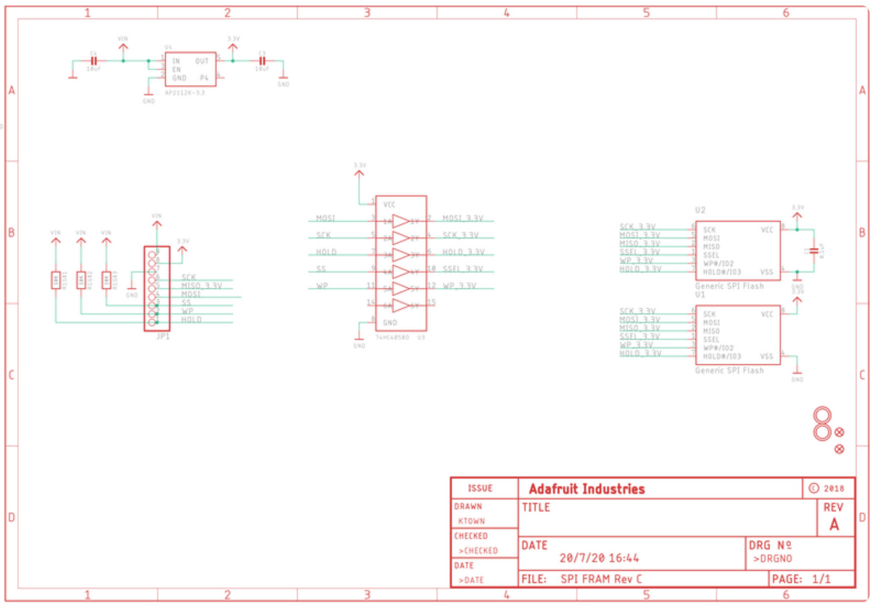
Downloads

Datasheets & Files

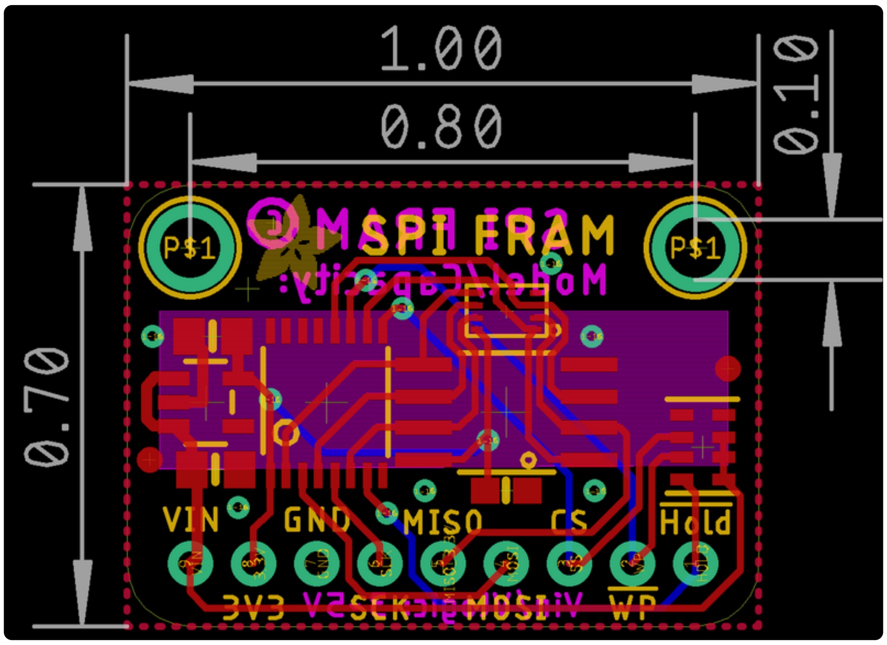
- MB85RS4MT Datasheet (https://adafru.it/NdS)
- Fritzing object in Adafruit Fritzing library (https://adafru.it/NdT)
- EagleCAD PCB files on GitHub (https://adafru.it/q6a)

Schematic

The schematic and PCB is identical for the 2 and 4 MBit versions



Fab Print


1.00
0.80
0.10
0.70
SPI FRAM
VIN
GND
MISO
CS
Hold
3V3
SCK
MOSI
WP