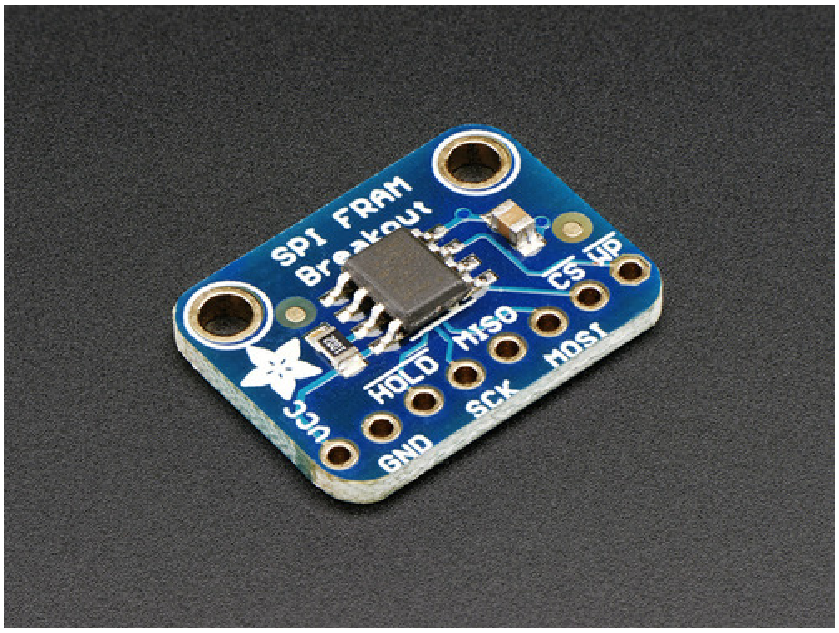
Adafruit SPI FRAM Breakout

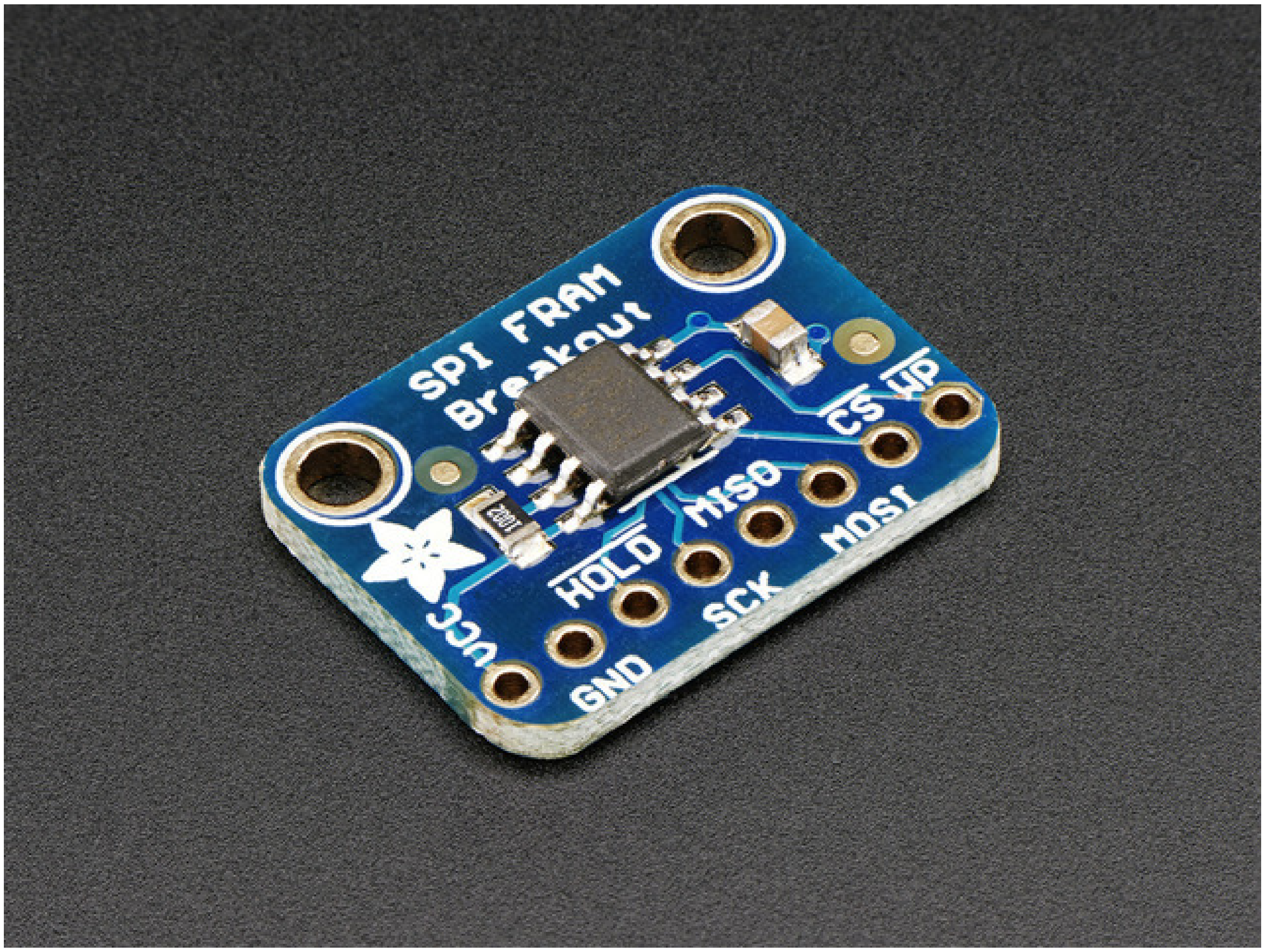
Created by lady ada



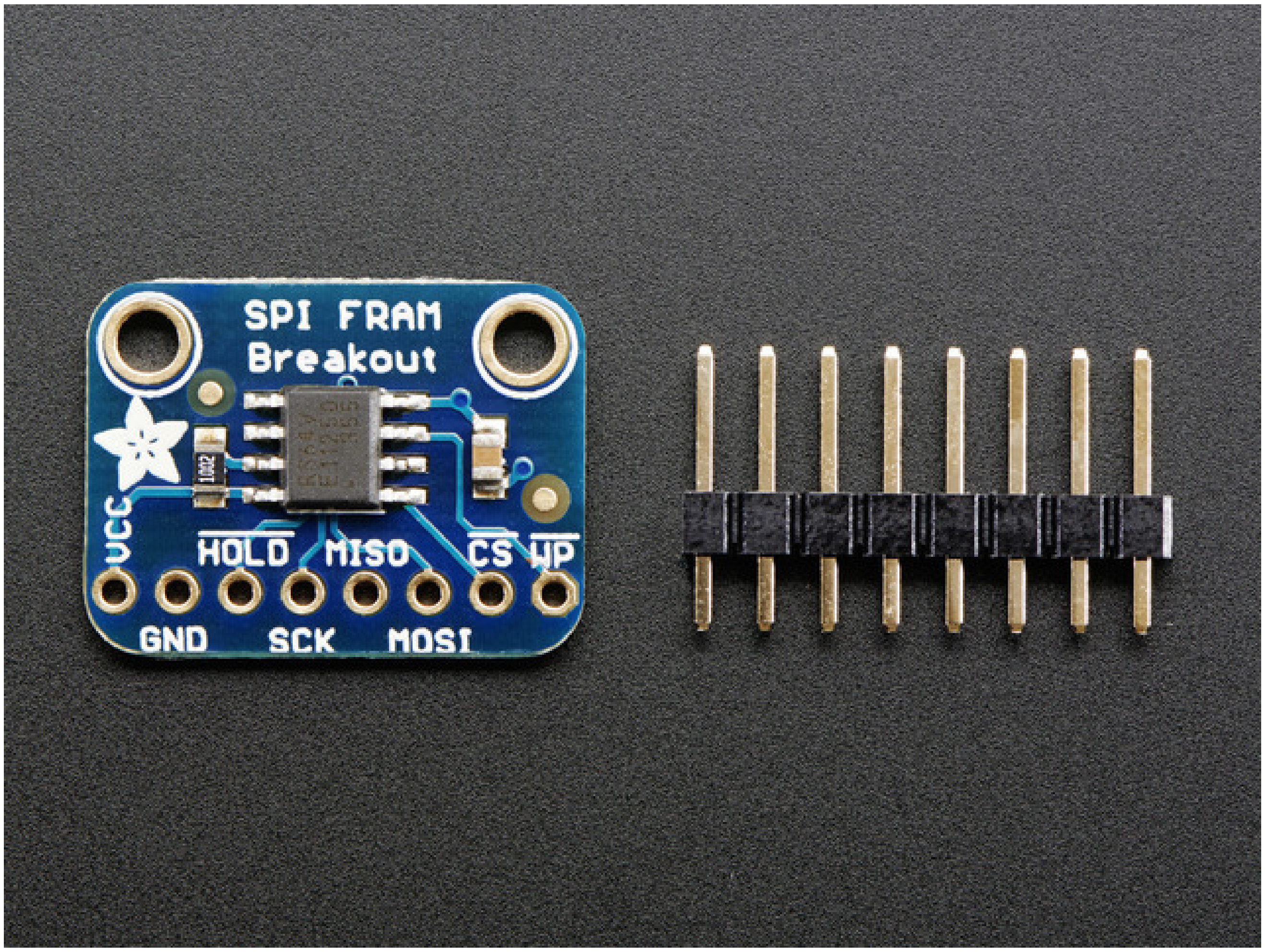
Last updated on 2016-08-24 05:35:38 PM UTC

Guide Contents

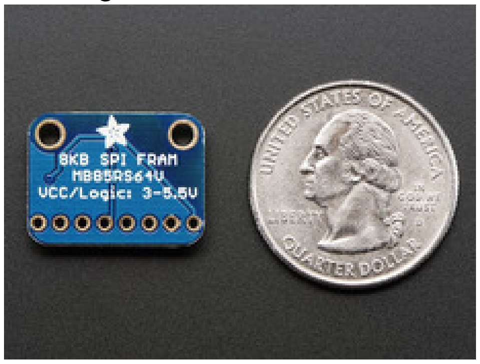
Overview



You're probably familiar with SRAM, DRAM, EEPROM and Flash but what about FRAM? FRAM is 'ferroelectric' RAM, which has some very interesting and useful properties. Unlike SRAM, FRAM does not lose the data when power is lost. In that sense it's a durable storage memory chip like Flash. However, it is much faster than Flash - and you don't have to deal with writing or erasing pages.

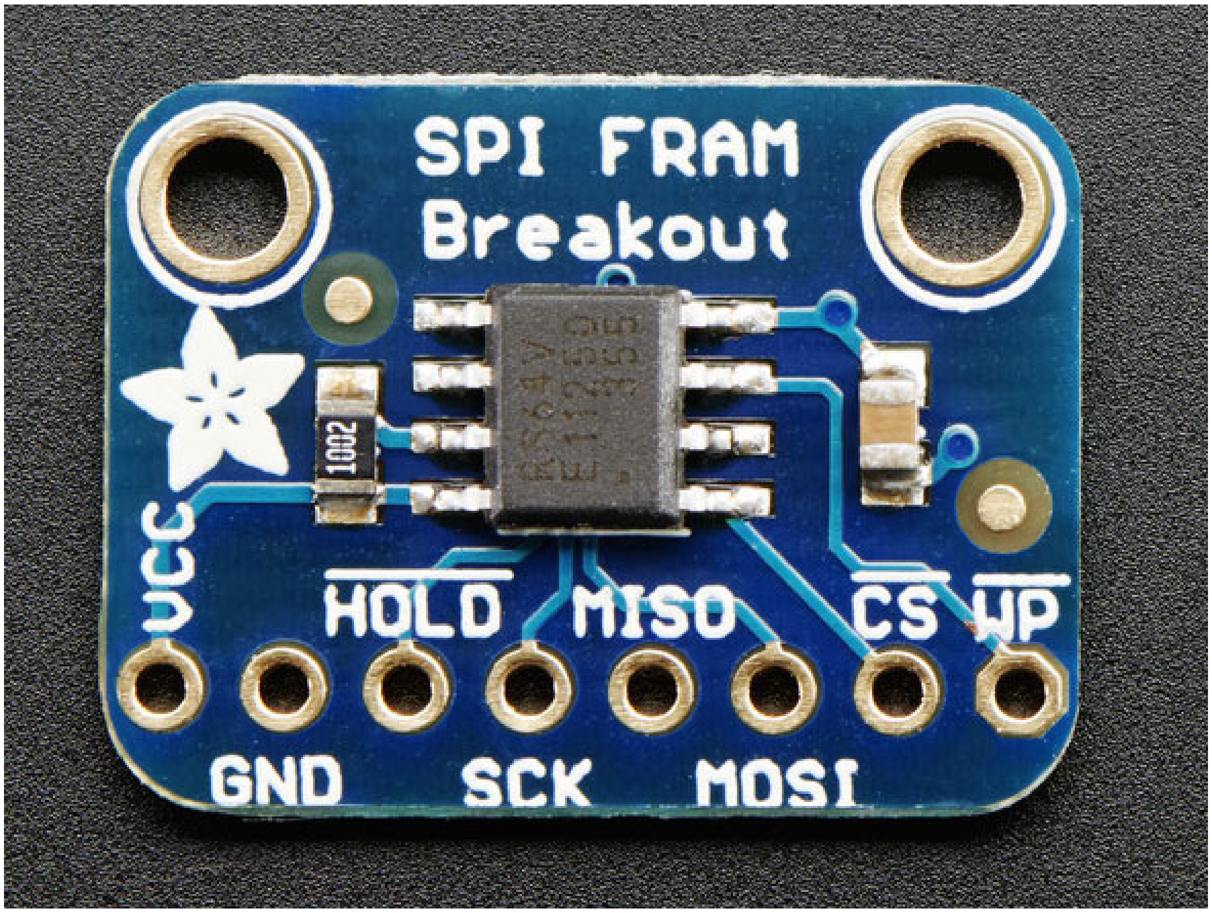


This particular FRAM chip has 64 Kbits (8 KBytes) of storage, interfaces using SPI, and can run at up to 20MHz SPI rates. Each byte can be read and written instantaneously (like SRAM) but will keep the memory for 95 years at room temperature. Each byte can be read/written 10,000,000,000,000 times so you don't have to worry too much about wear leveling.



With the best of SRAM and Flash combined, this chip can let you buffer fairly-high speed data without worrying about data-loss.

Pinouts



The FRAM chip is the little guy in the middle. On the bottom we have the power and interface pins

(http://adafru.it/dui)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.
- **GND** - common ground for power and logic

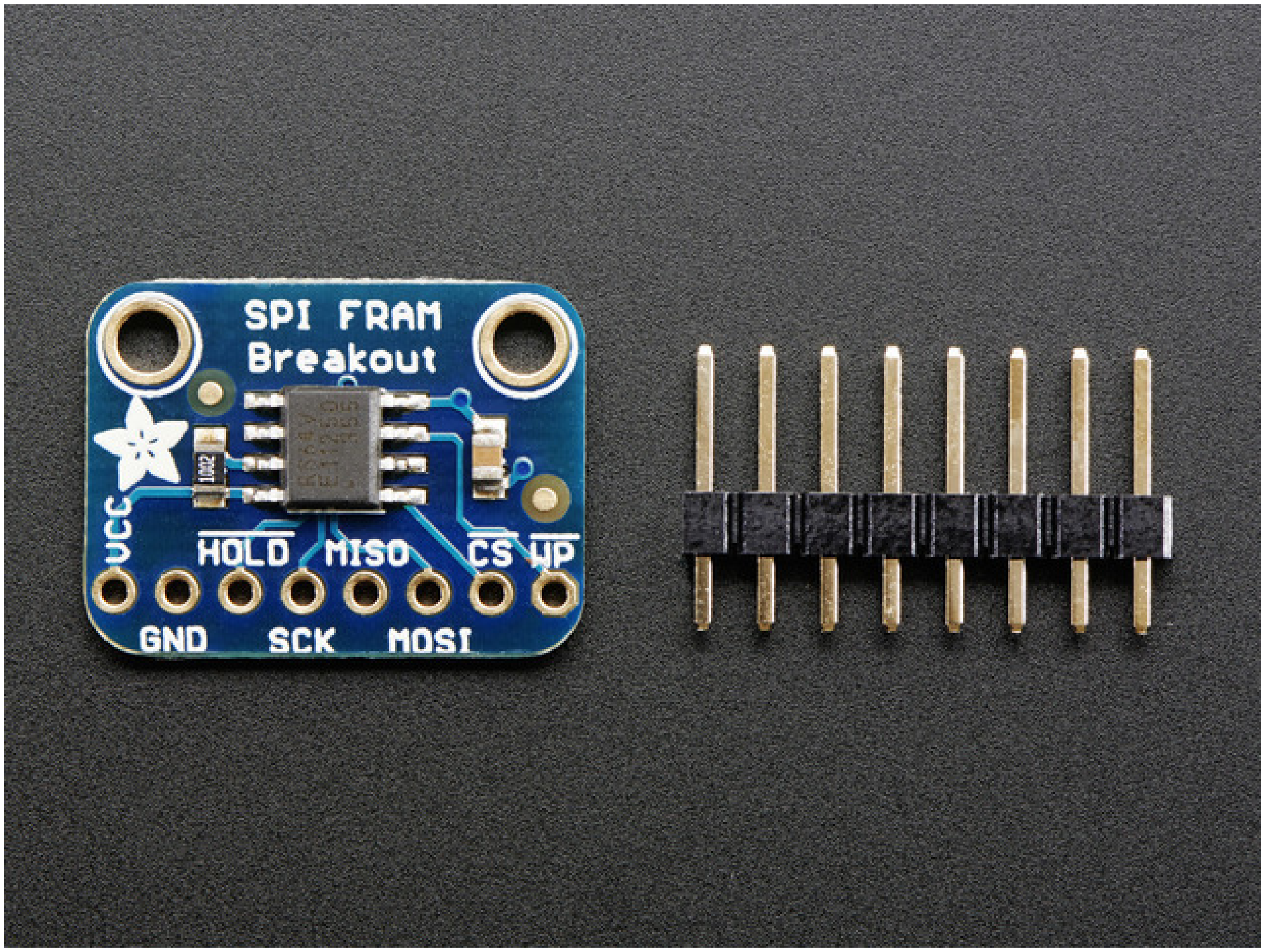
SPI Logic pins:

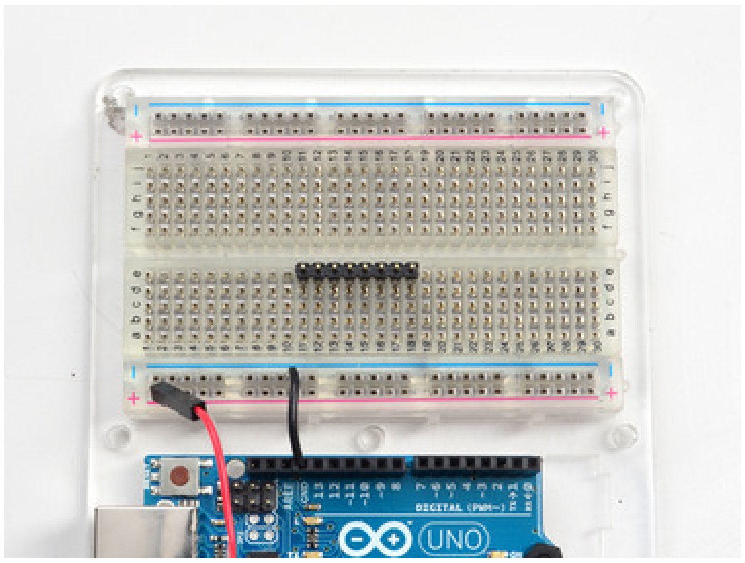
All pins are 3-5V compliant and use whatever logic level is on**VCC**

- **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.

- **SCK** - This is the SPI clock pin, its an input to the chip
- **MISO** - this is the Master In Slave Out pin, for data sent from the FRAM to your processor
- **MOSI** - this is the Master Out Slave 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)

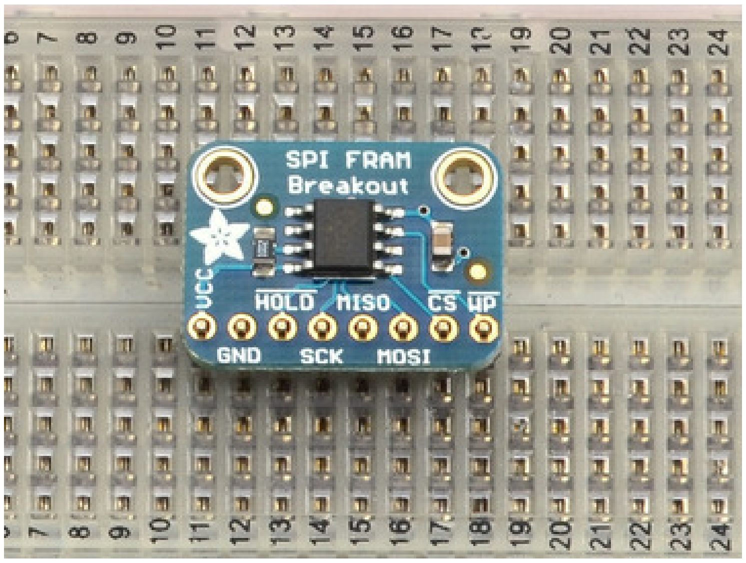
Assembly



- 

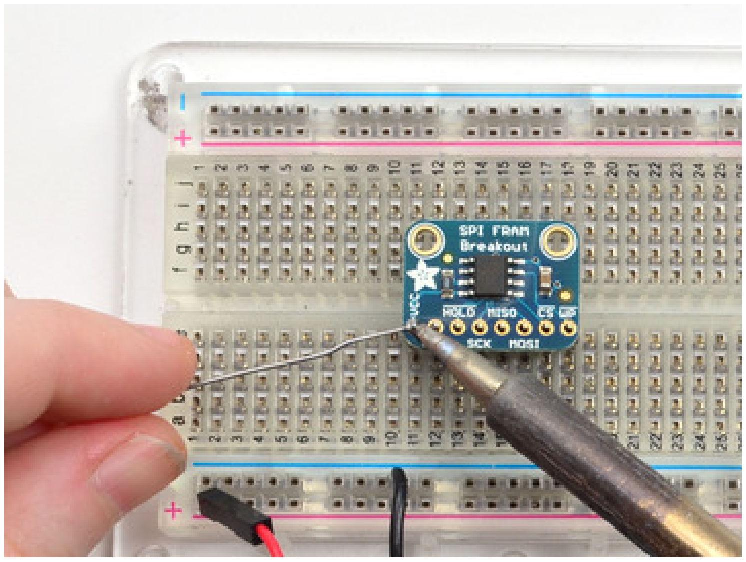
  ### Prepare the header strip:

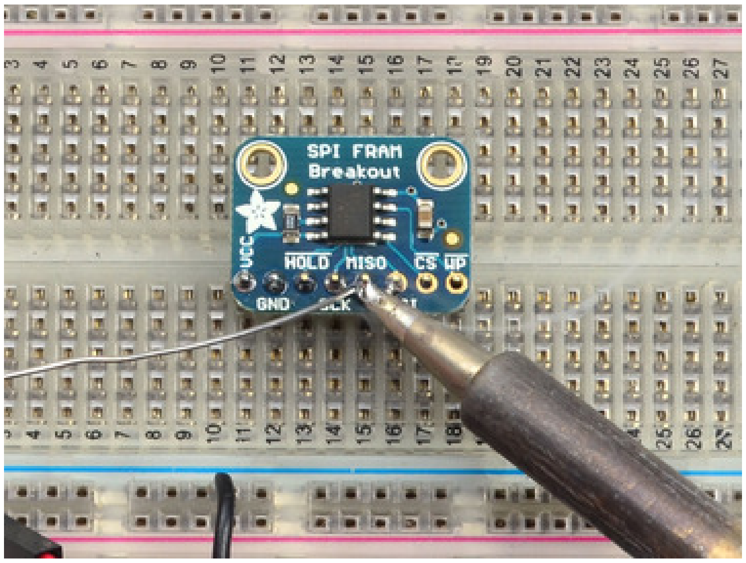
  Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

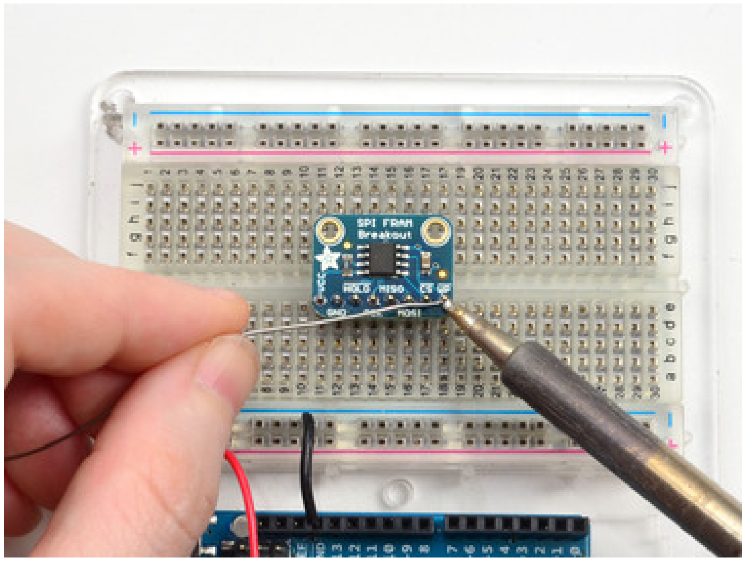
- 

Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads

- 

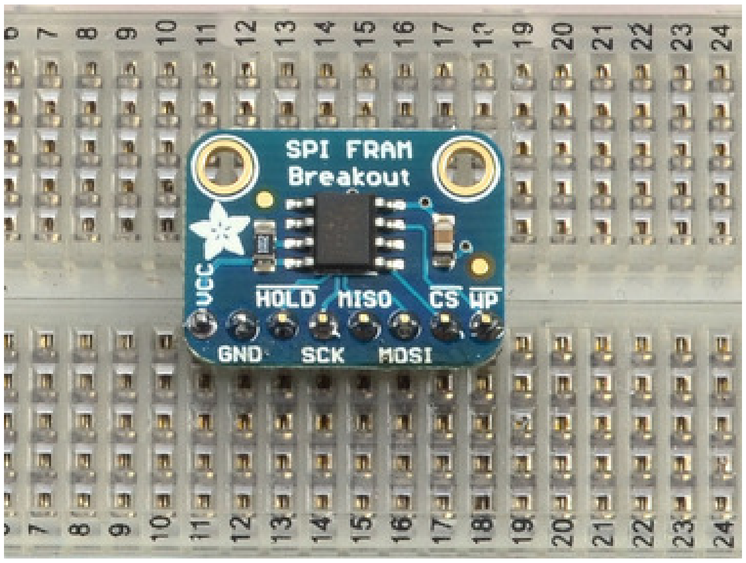
- 

- 

And Solder!

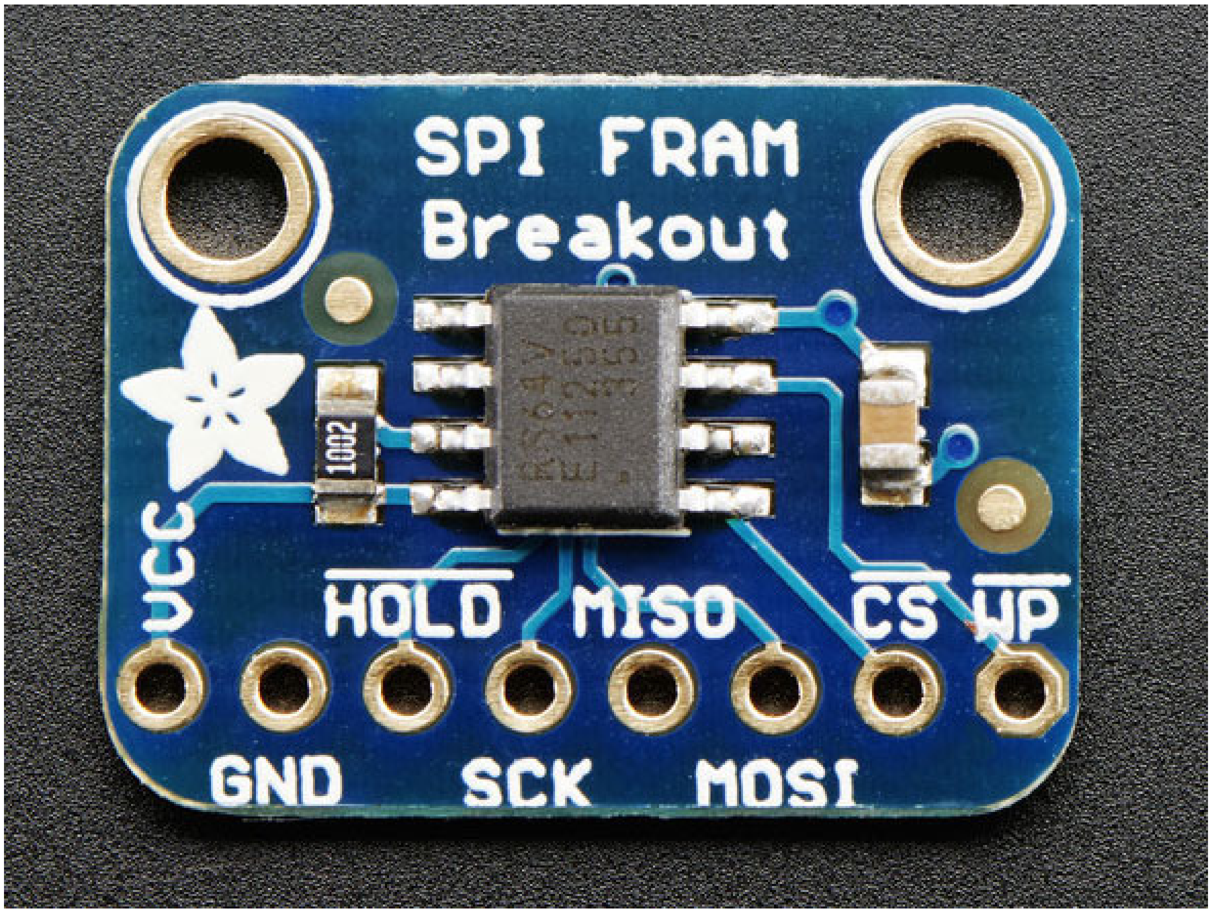
Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](http://adafru.it/aTk) (http://adafru.it/aTk)).

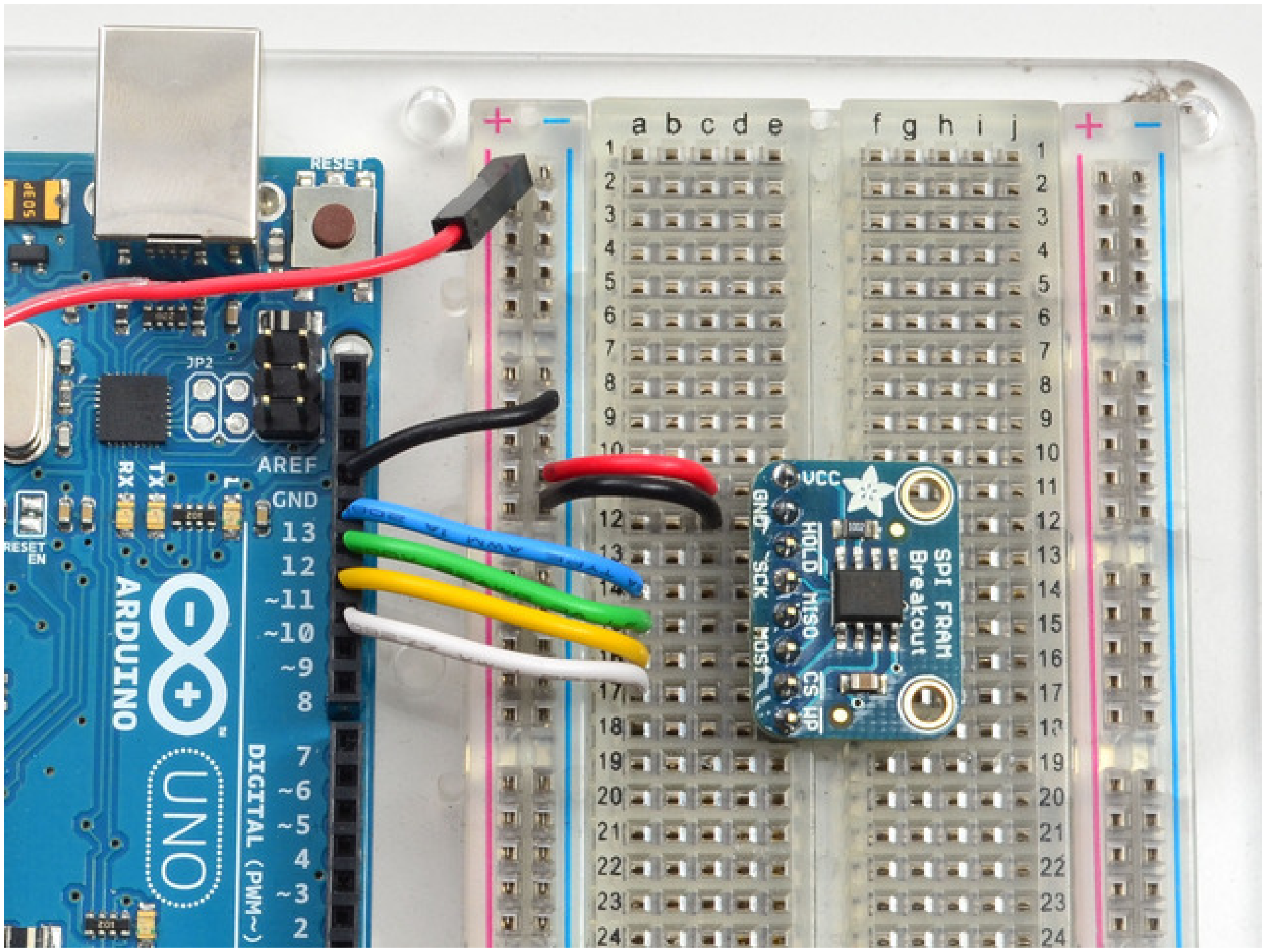
- 

You're done! Check your solder joints visually and continue onto the next steps

Wiring and Test



Arduino Wiring


Woops, the above has VCC and GND both connected to ground - Instead, connect VCC to the red power rail!

You can easily wire this breakout to any microcontroller, we'll be using an Arduino

- 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

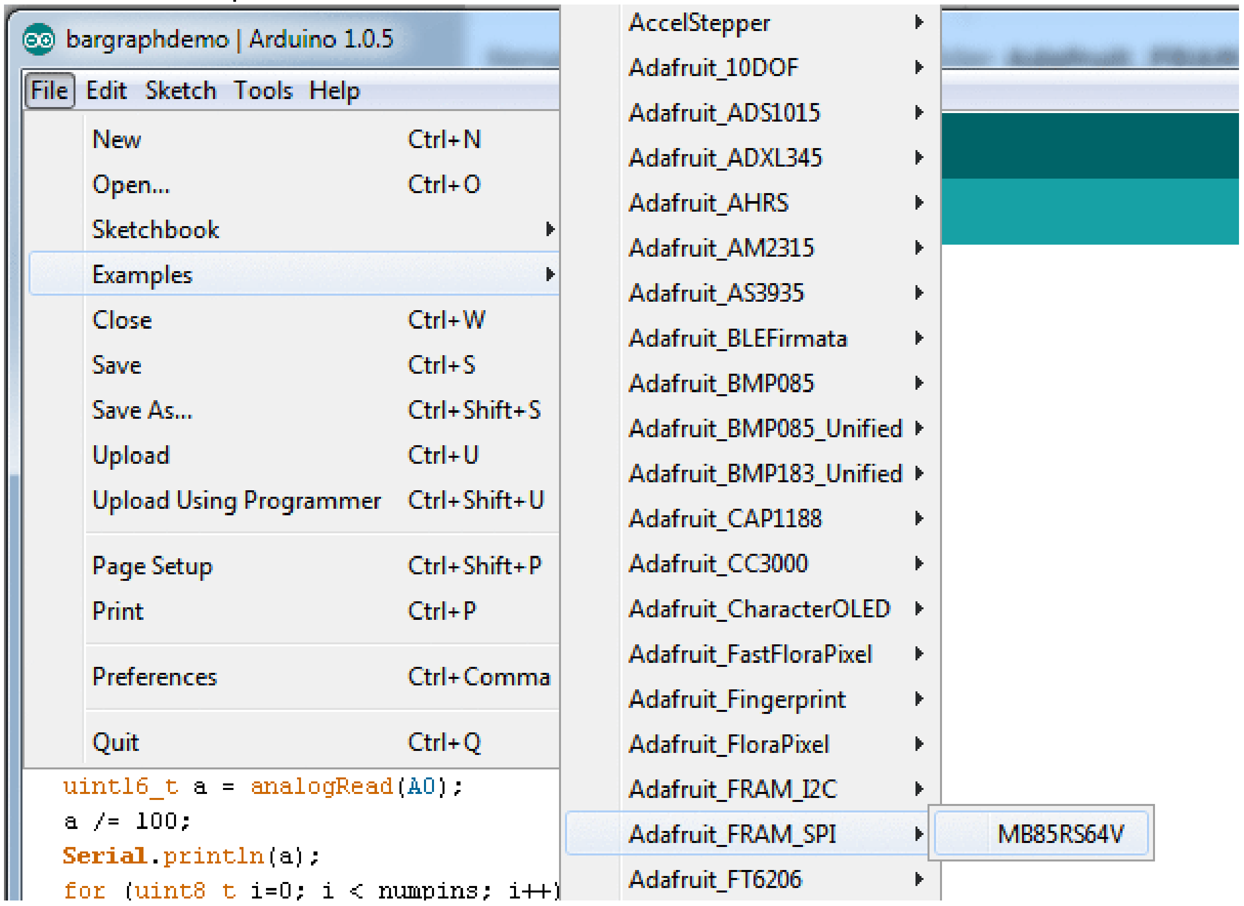
To begin reading and writing data, you will need todownload Adafruit_FRAM_SPI from our github repository (http://adafru.it/du5). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip
Download Adafruit_FRAM_SPI Library
http://adafru.it/du6
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 (http://adafru.it/aYM)

Load Demo

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

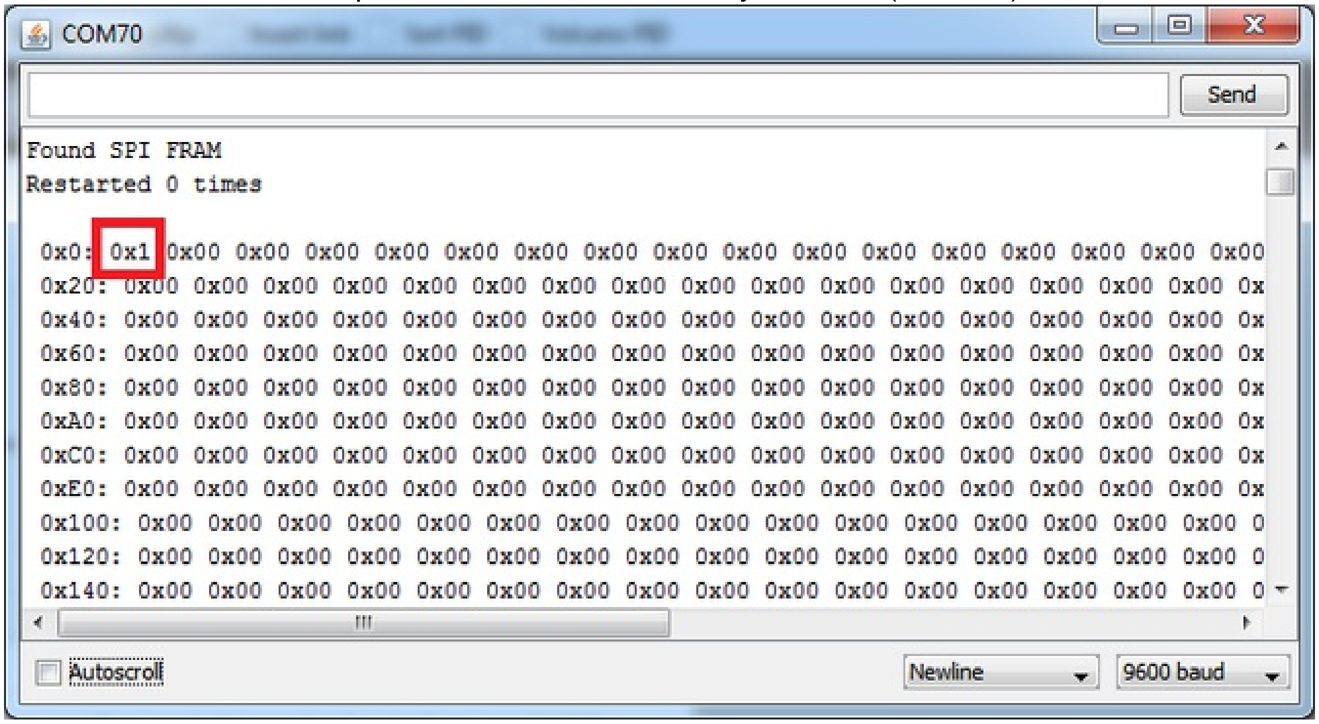


```
uint16_t a = analogRead(A0);
a /= 100;
Serial.println(a);
for (uint8_t i=0; i < numpins; i++)
```

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! (http://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

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!

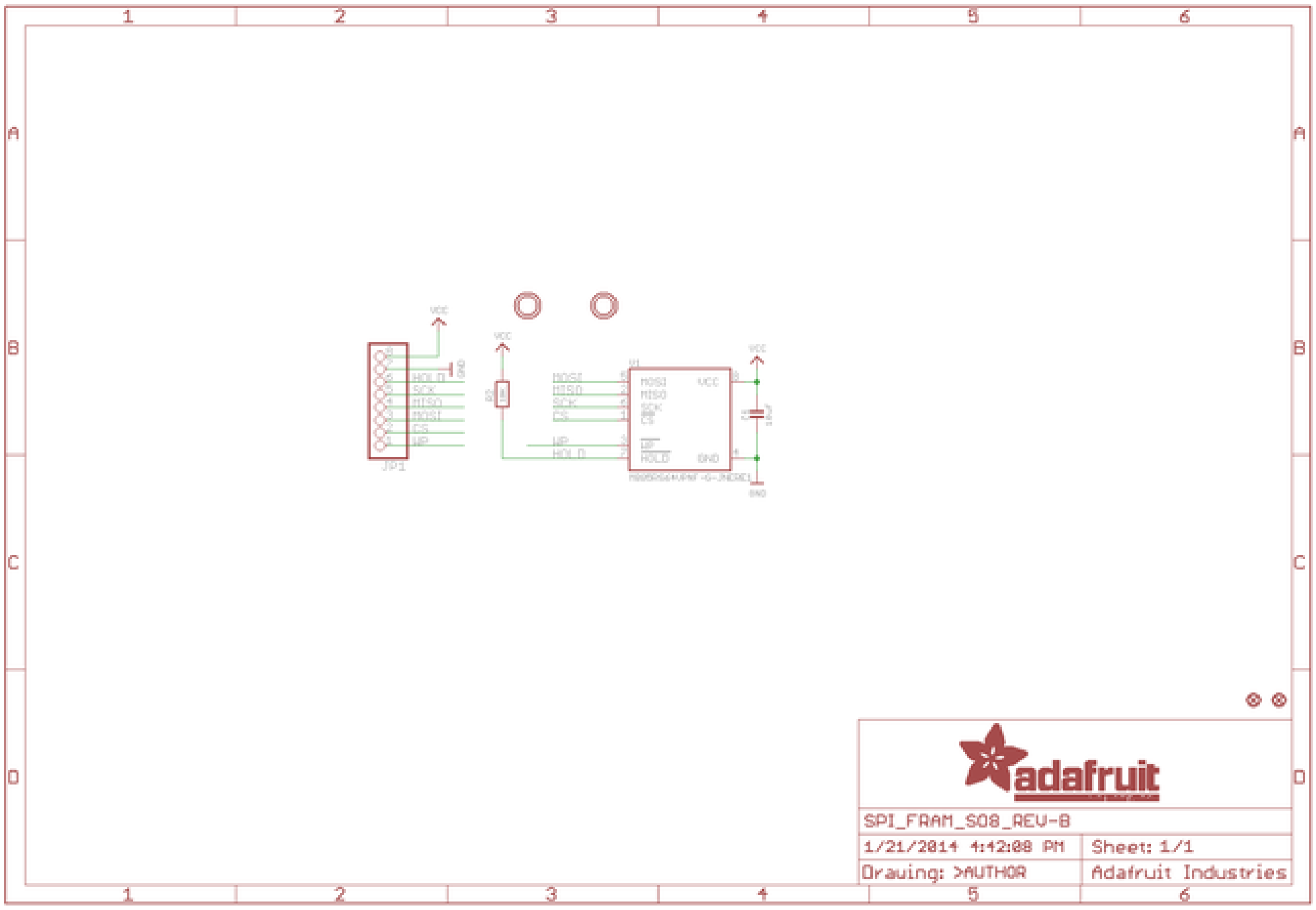
uint8_t getStatusRegister();
setStatusRegister(uint8_t value);

Downloads

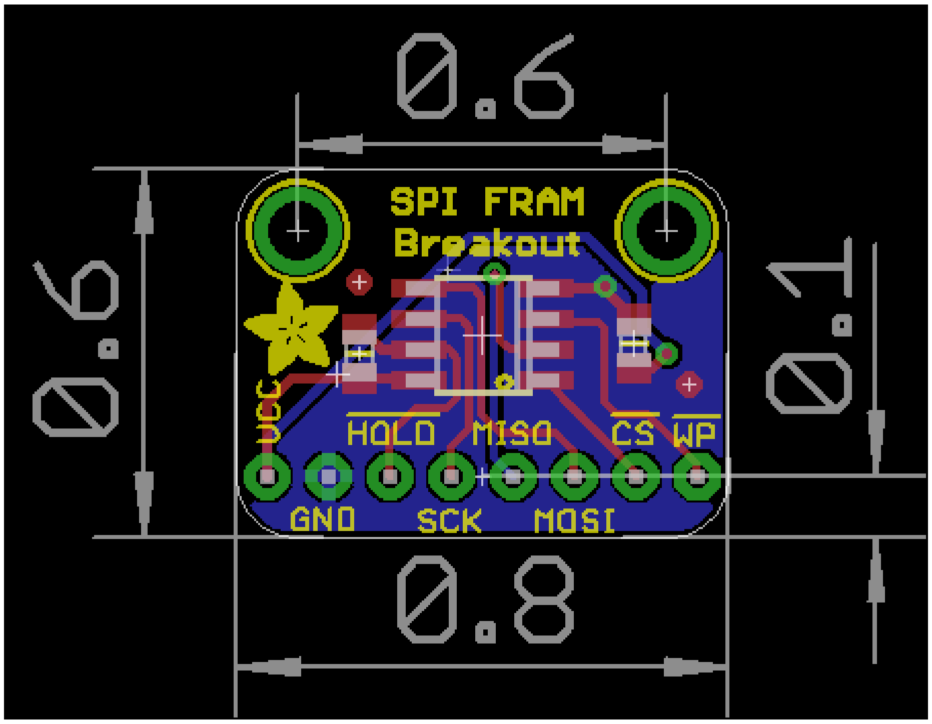
Datasheets & Files

- [MB85RS64V Datasheet](http://adafru.it/du7) (http://adafru.it/du7)
- [Fritzing object in Adafruit Fritzing library](http://adafru.it/aP3) (http://adafru.it/aP3)
- [EagleCAD PCB files](http://adafru.it/q6a) (http://adafru.it/q6a)

Schematics



Fabrication Print


0.6
0.6
0.1
0.8
SPI FRAM
Breakout
VCC
HOLD
MISO
CS
WP
GND
SCK
MOSI