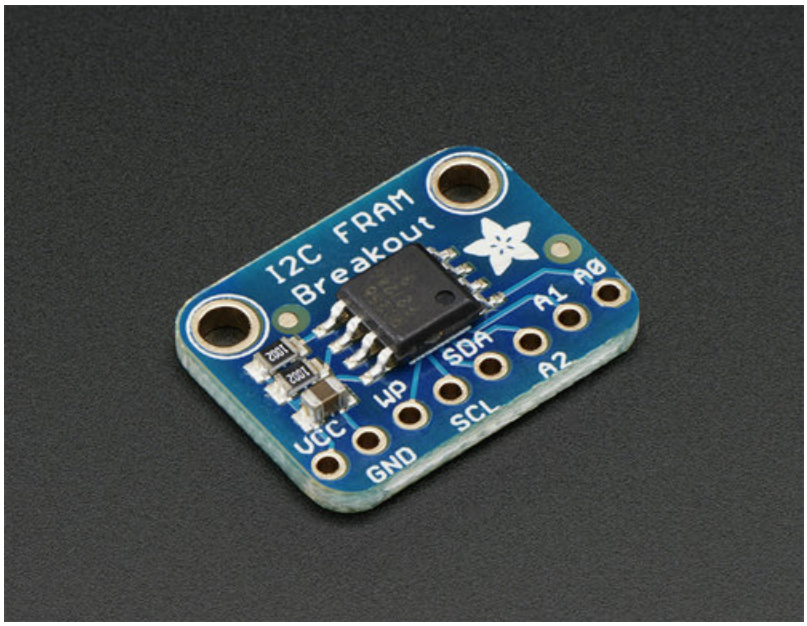
adafruit learning system

Adafruit I2C FRAM Breakout

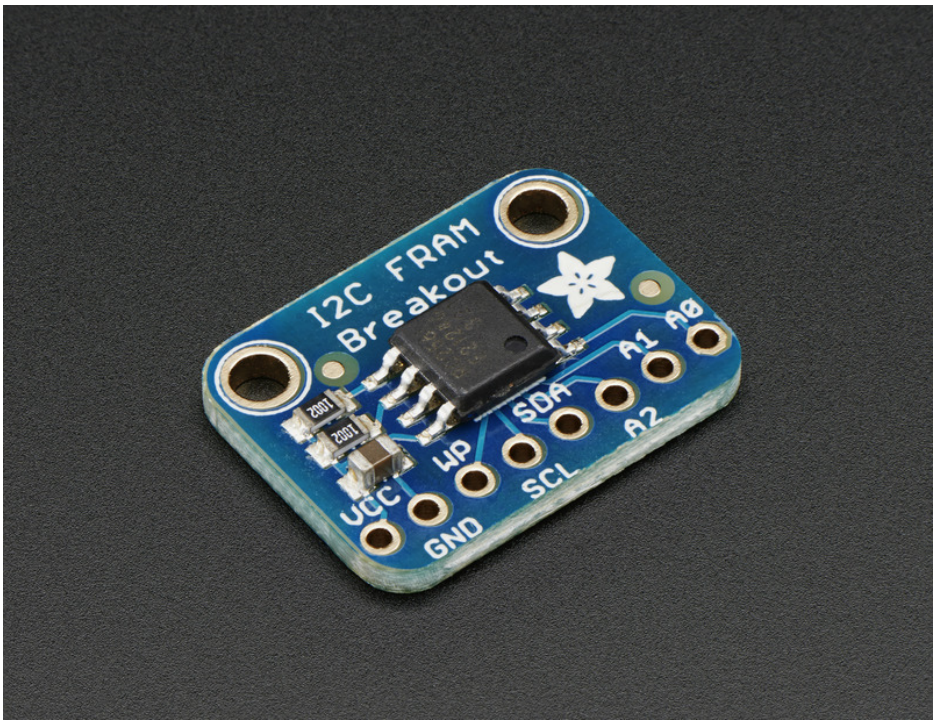
Created by lady ada



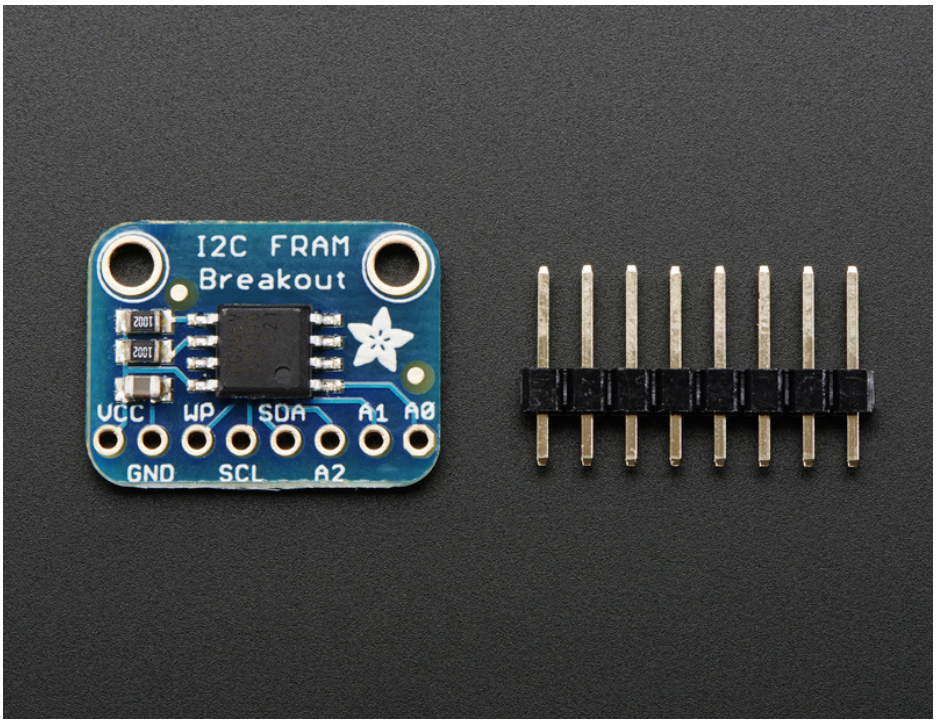
Last updated on 2018-12-21 09:33:51 PM UTC

Guide Contents

Overview

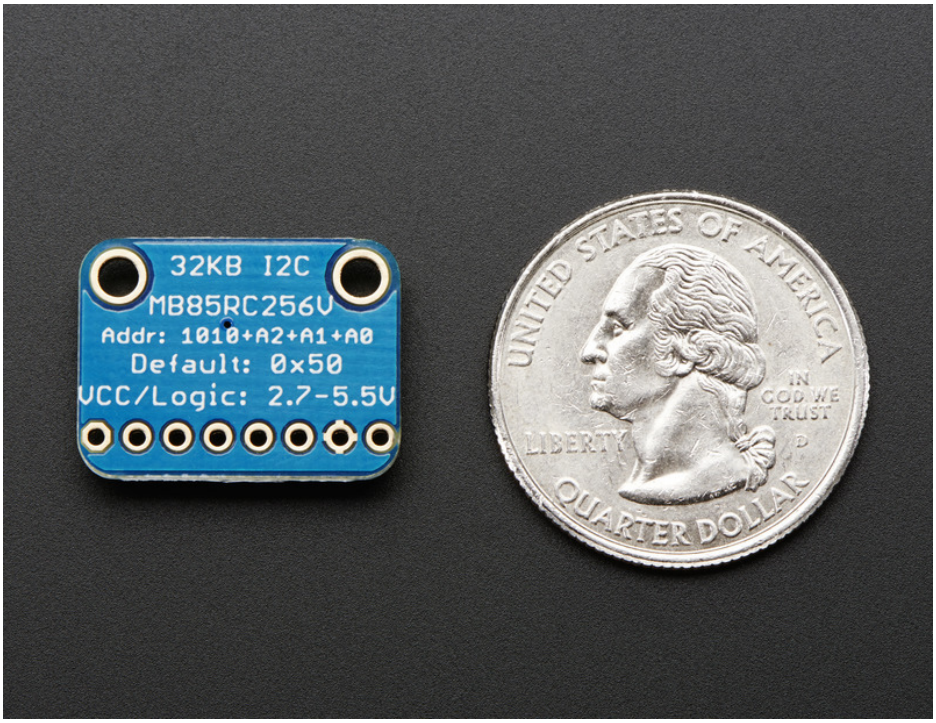


You're probably familiar with SRAM, DRAM, EEPROM and Flash but what about FRAM? FRAM is 'ferroelectric' RAM, which has some very interesting and useful properties. Unlike SRAM, FRAM does not lose the data when power is lost. In that sense it's a durable storage memory chip like Flash. However, it is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



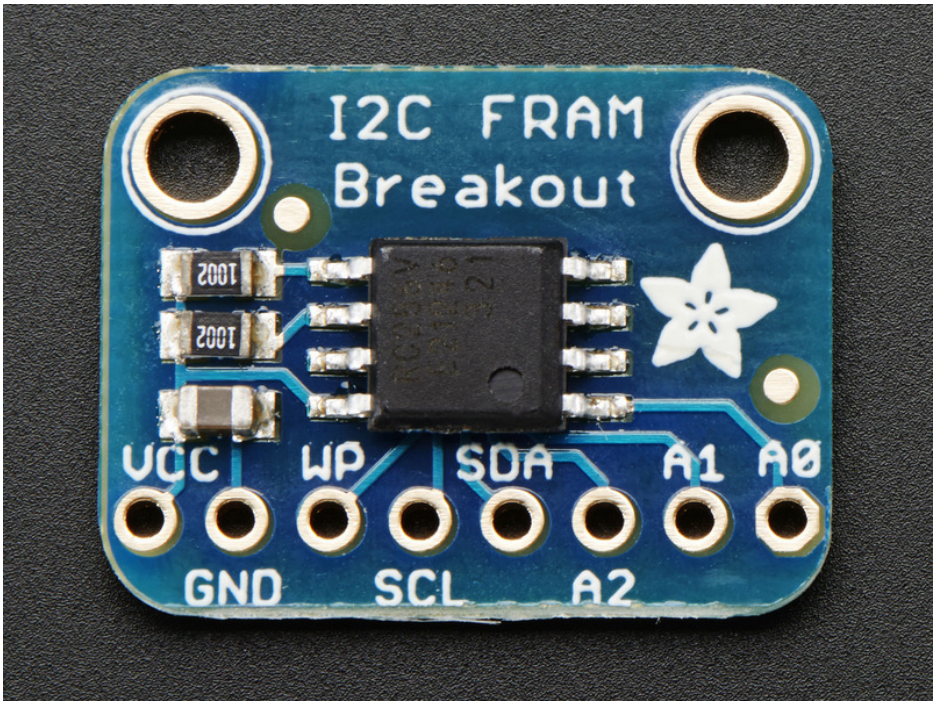
This particular FRAM chip has 256 Kbits (32 KBytes) of storage, interfaces using I2C, and can run at up to 1MHz I2C

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

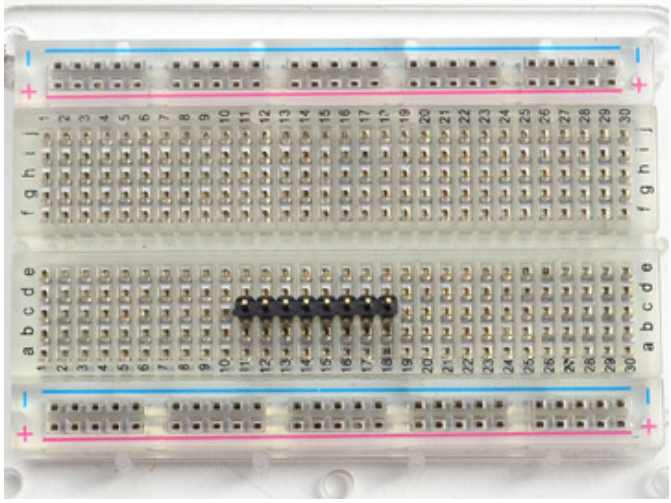
Power Pins:

- **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

I2C Logic pins:

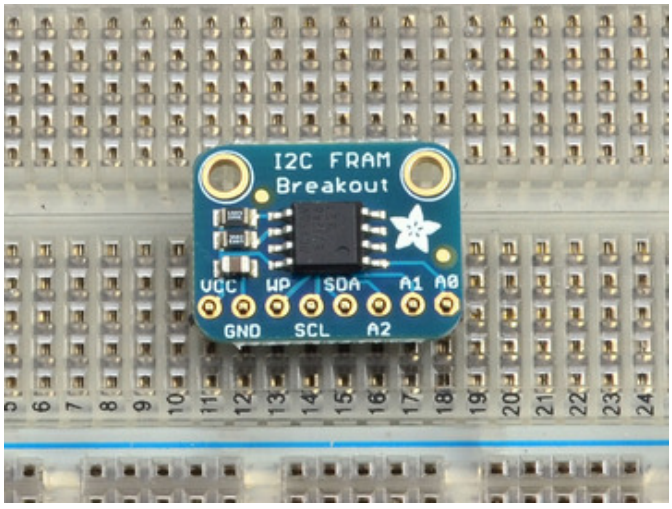
- **WP** - Write Protect pin. This is used to force write protection so you cannot write to the FRAM. It has an internal pulldown. Bring to a high voltage (VCC) to turn on WP
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.
- **A2, A1, A0** - These are the I2C address selection pins. By default the I2C address is 0x50. Connecting these pins to VCC and power cycling the chip will adjust the lower three bits of the address. For example, if A0 is high, the address is 0x51. If A1 and A2 are high, the address is 0x56

Assembly



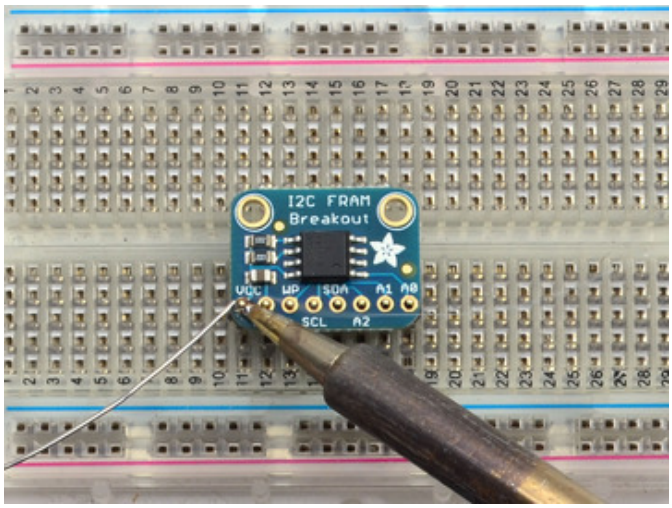
Prepare the header strip:

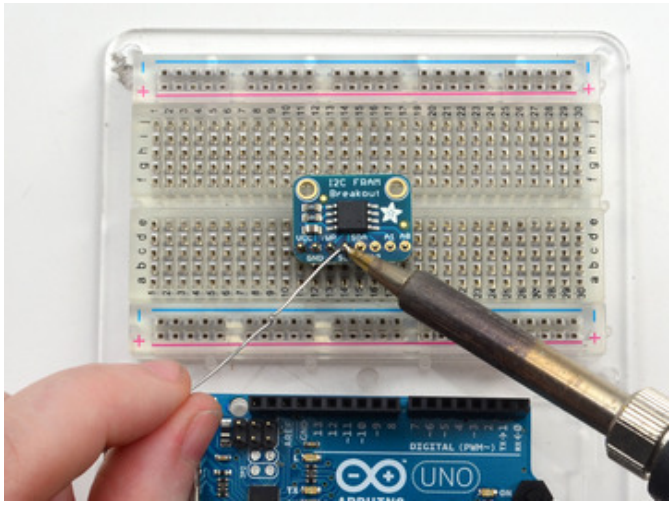
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

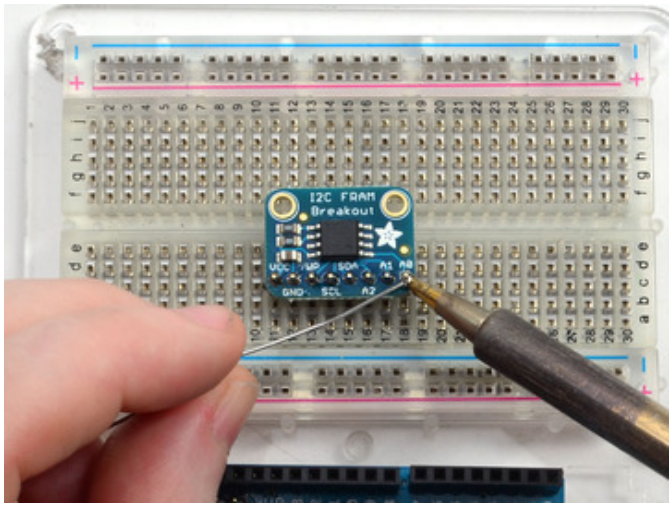


Add the breakout board:

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



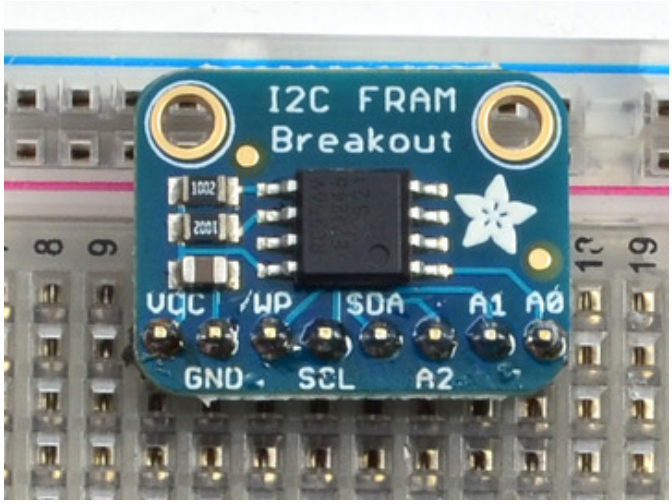




And Solder!

Be sure to solder all pins for reliable electrical contact.

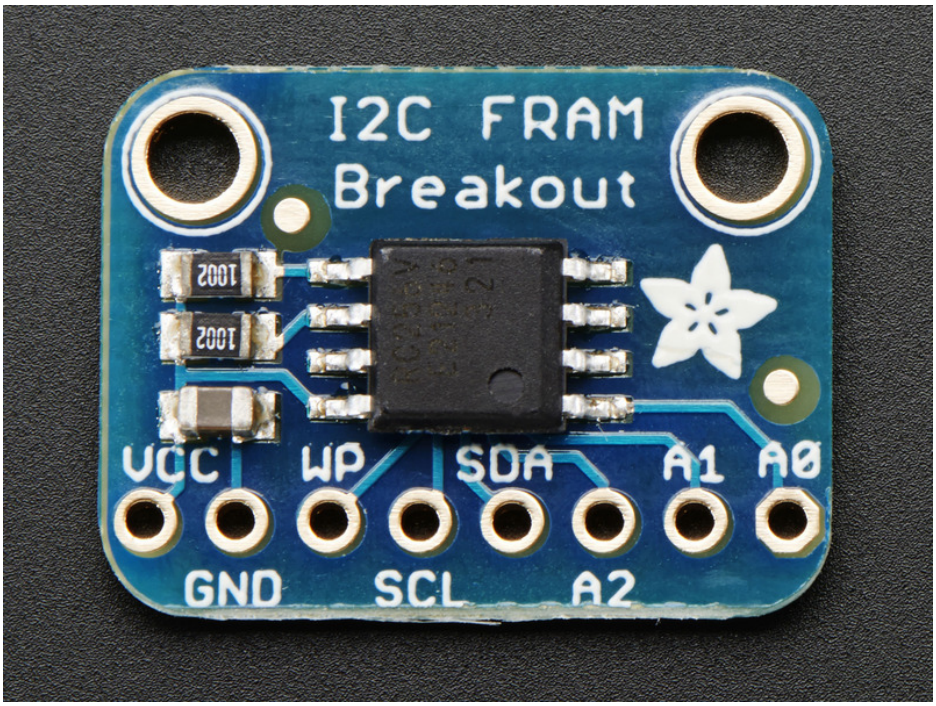
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk).



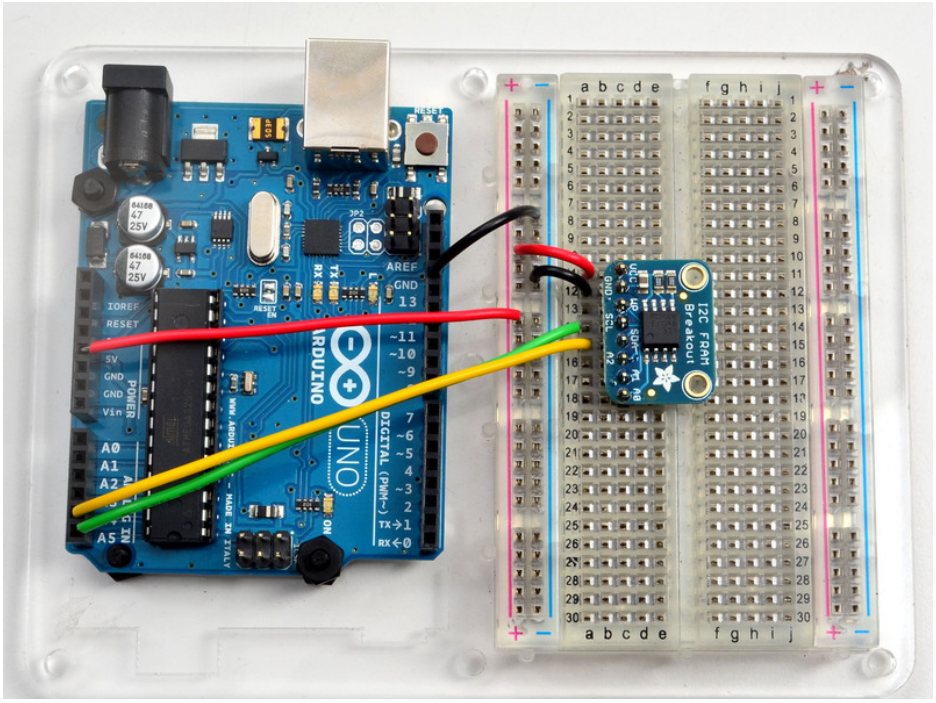
You're done! Check your solder joints visually and continue onto the next steps

Arduino Test

Arduino Wiring



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 **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The I2C FRAM has a default I2C address of **0x50** but you can set the address to any of 8 values between 0x50 and 0x57 so you can have up to 8 of these chips all sharing the same SCL/SDA pins. We suggest just connecting up the power and SDA/SCL pins for now. Once you have things working, change up the address as desired.

Download Adafruit_FRAM_I2C

To begin reading and writing data to the chip, you will need to download Adafruit_FRAM_I2C from our github repository (https://adafru.it/dtq). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

https://adafru.it/dtr

https://adafru.it/dtr

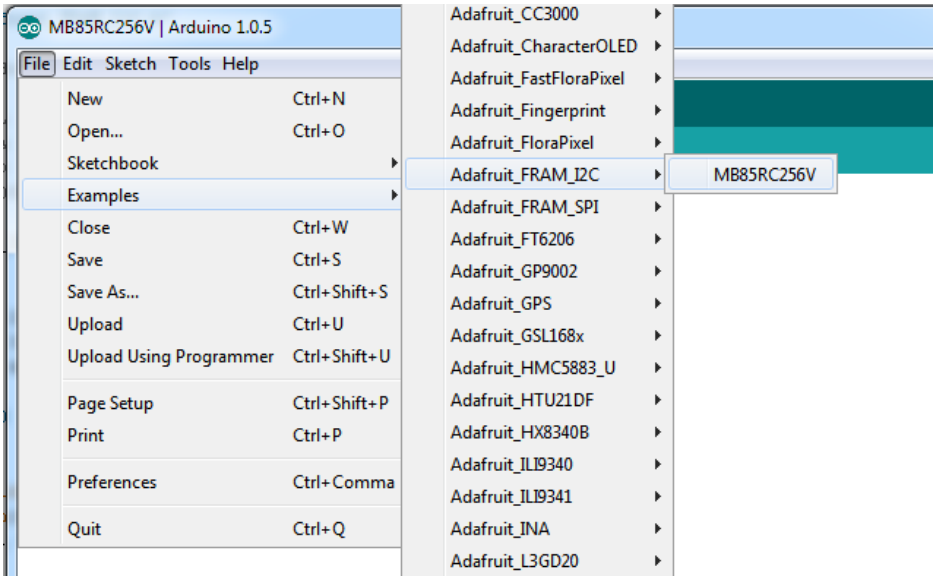
Rename the uncompressed folder **Adafruit_FRAM_I2C** and check that the **Adafruit_FRAM_I2C** folder contains **Adafruit_FRAM_I2C.cpp** and **Adafruit_FRAM_I2C.h**

Place the **Adafruit_FRAM_I2C** 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_I2C->MB85RC256V** and upload to your Arduino wired up to the sensor

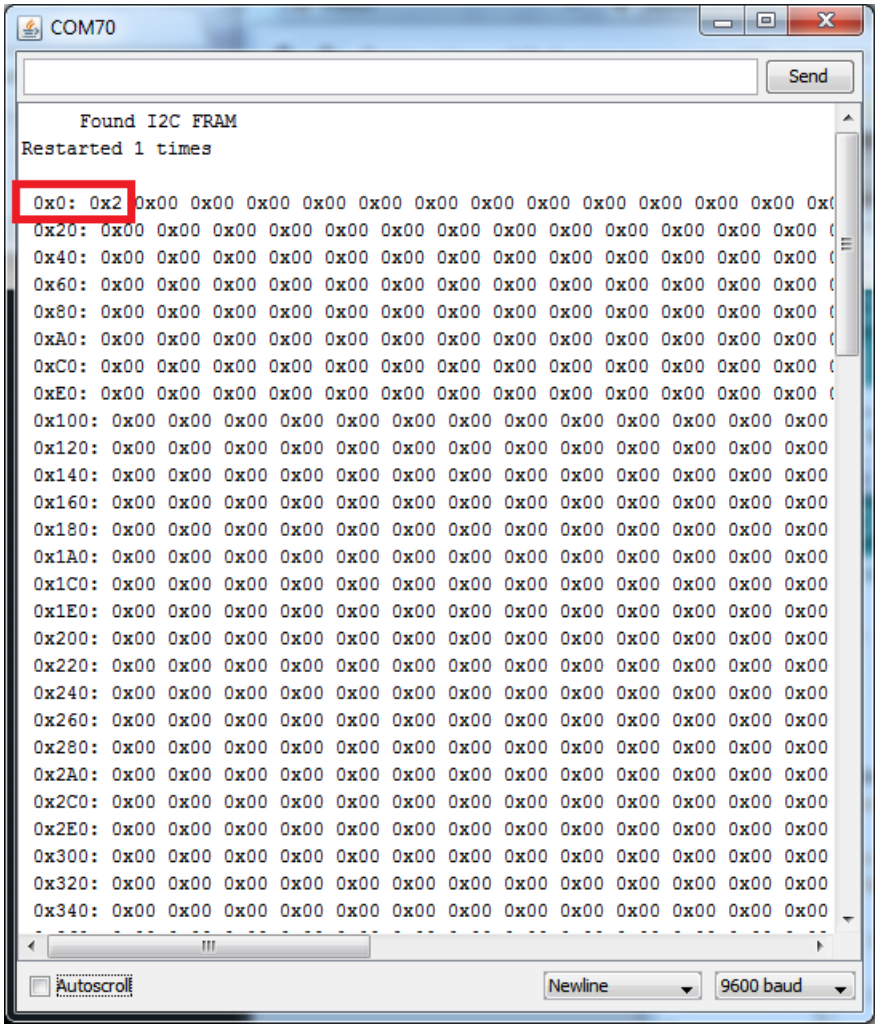


Thats it! Now open up the serial terminal window at 9600 speed to begin the test.

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 256KB!)



Library Reference

The library we have is simple and easy to use

You can create the FRAM object with

```
Adafruit_FRAM_I2C fram = Adafruit_FRAM_I2C();
```

Then when you **begin**(), pass in the i2c address. The default is 0x50 so if you don't put any value in the default is used. If you have different addresses, call something like

```
fram.begin(0x53)
```

for example.

Then to write a value, call

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

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.

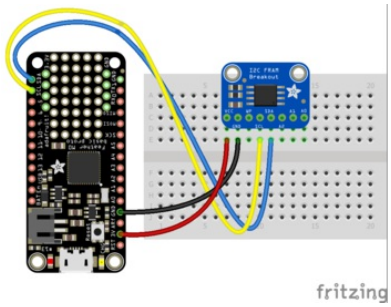
CircuitPython

It's easy to use the SPI FRAM Breakout with 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

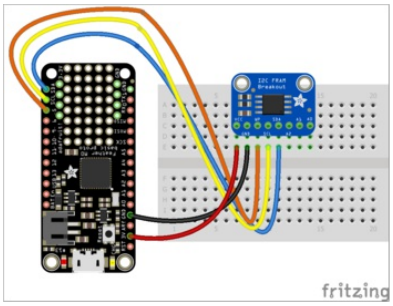
First we'll wire up a SPI FRAM Breakout to a microcontroller, as was shown in the Arduino Test page (https://adafru.it/DpH).

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



- **Board 3V** to **sensor VIN**
- **Board GND** to **sensor GND**
- **Board SDA** to **sensor SDA**
- **Board SCL** to **sensor SCL**

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 SDA** to **sensor SDA**
- **Board SCL** to **sensor SCL**
- **Board D5** to **sensor WP**

The CircuitPython library includes write protection internally, so using the hardware write protection isn't necessary. However, if using an external source like a separate microcontroller, hardware write protection will likely be necessary.

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 **adafruit_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
i2c = busio.I2C(board.SDA, board.SCL)
fram = adafruit_fram.FRAM_I2C(i2c)
```

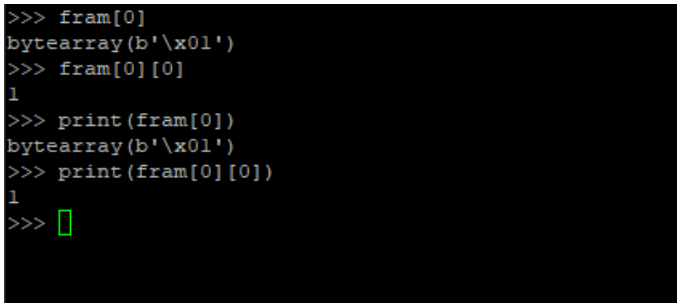
Or, if you're using the hardware write protection:

```
import board
import busio
import digitalio
import adafruit_fram
i2c = busio.I2C(board.SDA, board.SCL)
wp = digitalio.DigitalInOut(board.D6)
fram = adafruit_fram.FRAM_I2C(i2c, 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:



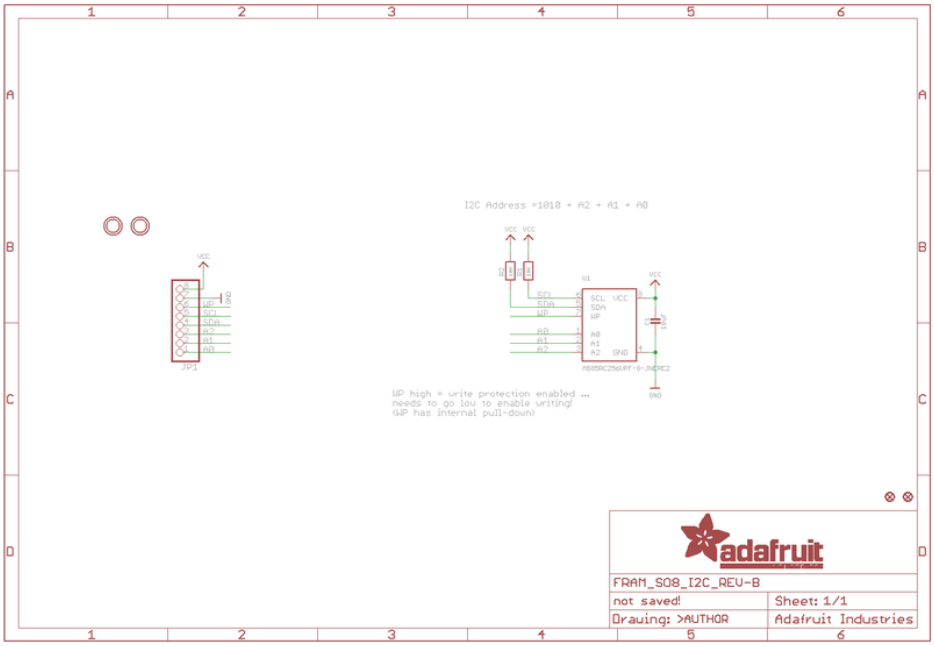
Full Example Code

```
## Simple Example For CircuitPython/Python I2C FRAM Library

import board
import busio
import adafruit_fram

## Create a FRAM object (default address used).
i2c = busio.I2C(board.SCL, board.SDA)
fram = adafruit_fram.FRAM_I2C(i2c)

## Optional FRAM object with a different I2C address, as well
## as a pin to control the hardware write protection ('WP'
## pin on breakout). 'write_protected()' can be used
## independent of the hardware pin.

#import digitalio
#wp = digitalio.DigitalInOut(board.D10)
#fram = adafruit_fram.FRAM_I2C(i2c,
#                              address=0x53,
#                              wp_pin=wp)

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

fram[0] = 1

## Read that byte to ensure a proper write.
## Note: reads return a bytearray

print(fram[0])

## Or write a sequential value, then read the values back.
## Note: reads return a bytearray. Reads also allocate
##       a buffer the size of slice, which may cause
##       problems on memory-constrained platforms.

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

Downloads

- Datasheet for the MB85RC256VPF FRAM chip (https://adafru.it/xDK)
- Fritzing object in the Adafruit Fritzing library (https://adafru.it/aP3)
- EagleCAD PCB files in GitHub (https://adafru.it/q6a)

Schematics



Fabrication Print