

Adafruit I2C FRAM Breakout

Created by lady ada



Last updated on 2017-07-14 05:38:45 AM UTC

Guide Contents

Guide Contents	2
Overview	3
Pinouts	6
Power Pins:	6
I2C Logic pins:	6
Assembly	8
Prepare the header strip:	8
Add the breakout board:	8
And Solder!	9
Wiring and Test	11
Arduino Wiring	11
Download Adafruit_FRAM_I2C	12
Load Demo	13
Library Reference	14
Downloads	16
Schematics	16
Fabrication Print	16



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**

* * * * * * * * * * * * * * * * * * * *	A A A A A A A A A A A A A A A A A A A	5 A A A A A A A A A A A A A A A A A A A
	D I2C FRAM Breakout	
	UCC UP SDA A1 A8 O O O O O O O O GND SCL A2	
6 8 8 8 9 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	11 12 13 13 13 15 15 15 15 15 15 15 15 15 15 15 15 15	194 204 214 224 234

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 <u>Guide to Excellent</u> <u>Soldering</u> (http://adafru.it/aTk)).



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



Wiring and 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 clockSCL 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 dataSDA 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<u>download</u> <u>Adafruit_FRAM_I2C from our github repository</u> (http://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 <u>Download Adafruit FRAM i2c Library</u> http://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: <u>http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use</u> (http://adafru.it/aYM)

Load Demo

Open up File->Examples->Adafruit_FRAM_I2C->MB85RC256V and upload to your Arduino wired up to the sensor

	Som MB85RC256V Arduino 1.0.5 File Edit Sketch Tools Help		Adafruit_CC3000 Adafruit_CharacterOLED Adafruit_FastFloraPixel	* *	
	New	Ctrl+N	Adafruit_Fingerprint	•	
	Open	Ctrl+0	Adafruit_FloraPixel	▶	
	Sketchbook	•	Adafruit_FRAM_I2C	•	MB85RC256V
1	Examples	Chello M	Adafruit_FRAM_SPI	•	
l	Close	Ctrl+W	Adafruit_FT6206	•	
l	Save Ac	Ctrl+Shift+S	Adafruit_GP9002		
l	Upload	Ctrl+II	Adafruit_GPS		
l	Upload Using Programmer	Ctrl+Shift+U	Adafruit_GSL168x		
l			Adafruit_HMC5883_U	•	
	Page Setup	Ctrl+Shift+P	Adafruit_HTU21DF		
l	Print	Ctrl+P	Adafruit_HX8340B	1	
l	Preferences	Ctrl+Comma	Adafruit_ILI9340		
	Quit	Ctrl+0	Adatruit_ILI9541		
ł	Zuit	Cutty	Adafruit_L3GD20	•	

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.

🛃 COM7	70		4		-		-	-	,	-	ĺ			٢
													Send	
Fo	und I	2C FR	AM											-
Restart	ed 1	times												
0x0: 0	x2 0x	:00 0x	00 Ox	00 Ox	00 Ox	00 0x	KO 00	x0 00	x0 00	x0 00	x0 00	:00 0x	:00 0x	t
0x20:	0 x 00	0x00	0x00	0x00	0 x 00	0 x 00	0 x 00	(
0x40:	0 x 00	0x00	0x00	0x00	0x00	0 x 00	0 x 00	(-						
0x60:	0 x 00	0x00	0x00	0x00	0x00	0 x 00	0 x 00	(
0x80:	0 x 00	0 x 00	0 x 00	0x00	0 x 00	0 x 00	0 x 00	0x00	0x00	0x00	0x00	0 x 00	0 x 00	(
0xA0:	0x00	0 x 00	0 x 00	0x00	0x00	0 x 00	0 x 00	0x00	0x00	0x00	0x00	0 x 00	0 x 00	
0xC0:	0x00	0x00	0x00	0x00	0x00									
OxEO:	0x00	0x00	0x00	0x00	0x00									
0x100:	0x00	0 0x00	0x00	0x00	0x00	0x00	0x00							
0x120:	0x00	000	0x00	0.000	0x00	0.000	0x00	0.000	0.000	0.000	0.000	000	000	
0x140:	0x00	0x00	0x00	0x00	0x00									
0x100.	0x00		0x00	0x00	0x00	0x00	0x00		0x00			0x00	0.000	
0x120.	0x00	0x00	0x00	0x00	0x00									
0x1C0:	0x00	0x00	0x00	0x00	0x00									
0x1E0:	0x00	0x00	0x00	0x00	0x00									
0x200:	0x00	0x00	0x00	0x00	0x00									
0x220:	0x00	0x00	0x00	0x00	0x00									
0x240:	0x00	0x00	0x00	0x00	0x00									
0x260:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0x00	
0x280:	0x00	0x00	0x00	0x00	0x00									
0x2A0:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
0x2C0:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0 x 00	0x00	0x00	0 x 00	0 x 00	
0x2E0:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0 x 00	0 x 00	
0x300:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
0x320:	0x00	0x00	0x00	0 x 00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
0x340:	0x00	0x00	0x00	0x00	0x00	Ŧ								
•			1										Þ	
Autoscroll Newline 9600 baud											•]			

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.



Downloads

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

Schematics



Fabrication Print

