Adafruit 24LC32 I2C EEPROM Breakout - 32Kbit / 4 KB

Created by Liz Clark

https://learn.adafruit.com/adafruit-24lc32-i2c-eeprom-breakout-32kbit-4-kb

Last updated on 2023-08-29 04:51:14 PM EDT
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Pinouts</td>
<td>5</td>
</tr>
<tr>
<td>• Power Pins</td>
<td></td>
</tr>
<tr>
<td>• I2C Logic Pins</td>
<td></td>
</tr>
<tr>
<td>• Address Pins</td>
<td></td>
</tr>
<tr>
<td>• Write-Protect Pin</td>
<td></td>
</tr>
<tr>
<td>• Power LED</td>
<td></td>
</tr>
<tr>
<td>Python &amp; CircuitPython</td>
<td>7</td>
</tr>
<tr>
<td>• CircuitPython Microcontroller Wiring</td>
<td></td>
</tr>
<tr>
<td>• Python Computer Wiring</td>
<td></td>
</tr>
<tr>
<td>• Python Installation of 24LC32 Library</td>
<td></td>
</tr>
<tr>
<td>• CircuitPython Usage</td>
<td></td>
</tr>
<tr>
<td>• Python Usage</td>
<td></td>
</tr>
<tr>
<td>• Example Code</td>
<td></td>
</tr>
<tr>
<td>• Read and Write</td>
<td></td>
</tr>
<tr>
<td>Python Docs</td>
<td>11</td>
</tr>
<tr>
<td>Arduino</td>
<td>12</td>
</tr>
<tr>
<td>• Wiring</td>
<td></td>
</tr>
<tr>
<td>• Library Installation</td>
<td></td>
</tr>
<tr>
<td>• Example Code</td>
<td></td>
</tr>
<tr>
<td>• Read and Write</td>
<td></td>
</tr>
<tr>
<td>Arduino Docs</td>
<td>17</td>
</tr>
<tr>
<td>Downloads</td>
<td>17</td>
</tr>
<tr>
<td>• Files</td>
<td></td>
</tr>
<tr>
<td>• Schematic and Fab Print</td>
<td></td>
</tr>
</tbody>
</table>
Overview

If you want to store calibration values, MAC addresses, non-secure access tokens, or other unique identifiers, EEPROM storage is a great option. EEPROM is long lasting, and doesn't need to be written in pages - a single byte can be written at once (unlike with flash memory!) EEPROM storage persists even when the power goes out, and can be over-written literally one million times.

Some microcontrollers, like the ATmega328, have built in EEPROM, usually around 64 to 1024 bytes of it. But some, especially ARM Cortex's, don't! What then? that's where this petite Adafruit 24LC32 I2C EEPROM Breakout comes in to help! With 32 Kbit (4...
KByte) of storage, and handy chainable Stemma QT connectors, it's just the right amount of simple I2C-controllable storage. Since it's external to your microcontroller or microcomputer, uploading new flash memory won't wipe the data from this chip.

We use the CAT24C32 (or equivalent) EEPROM, internally organized as 4096 words of 8 bits each. It features a 32 byte page write buffer (if you want to write faster than one byte at a time). Use 2 to 5V power/logic and up to 1 MHz clocked I2C. The default I2C address is 0x50 and there are three external address pins / jumpers to allow addressing of up to eight EEPROM devices on the same bus.

To make things easier and a bit more flexible, we've also included SparkFun Qwiic compatible STEMMA QT connectors for the I2C bus so you don't even need to
solder! Just plug in a compatible cable and attach it to your MCU of choice, and you’re ready to store and retrieve from a reliable and durable EEPROM. **QT Cable is not included, but we have a variety in the shop**.

For Arduino usage, **we recommend our FRAM/EEPROM library**.

---

**Pinouts**

![Image of pinouts](image)

The default I2C address is 0x50.

**Power Pins**

- **VIN** - this is the power pin. Since the EEPROM chip may use 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V.
- **3V** - 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.

**I2C Logic Pins**

- **SCL** - I2C clock pin, connect to your microcontroller I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- **SDA** - I2C data pin, connect to your microcontroller I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- **STEMMA QT** - These connectors allow you to connect your device to dev boards with STEMMA QT connectors or to other things with various associated accessories.
Address Pins

On the back of the board are three address jumpers, labeled A0, A1, and A2. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

On the front of the board are three address pins, labeled A0, A1 and A2. Just like the jumpers, these pins allow you to change the I2C address to connect multiple boards by connecting them to VIN.

The default I2C address is 0x50. The other address options can be calculated by “adding” the A0/A1/A2 to the base of 0x50.

A0 sets the lowest bit with a value of 1, A1 sets the next bit with a value of 2 and A2 sets the next bit with a value of 4. The final address is 0x50 + A2 + A1 + A0 which would be 0x57.

So for example if A2 is soldered closed and A0 is soldered closed, the address is 0x50 + 4 + 1 = 0x55.

If only A0 is soldered closed, the address is 0x50 + 1 = 0x51

If only A1 is soldered closed, the address is 0x50 + 2 = 0x52

If only A2 is soldered closed, the address is 0x50 + 4 = 0x54

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

<table>
<thead>
<tr>
<th>ADDR</th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
<th>ADDR</th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x50</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>0x54</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>0x51</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>0x55</td>
<td>H</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>0x52</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>0x56</td>
<td>L</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>0x53</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>0x57</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>
Write-Protect Pin

- WP - Write-protect pin. Connect to VCC to make the EEPROM read-only.

Power LED

- Power LED - In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled on. It is the green LED.

Python & CircuitPython

It's easy to use the 24LC32 with Python or CircuitPython, and the Adafruit_CircuitPython_on_24LC32() module. This module allows you to easily write Python code that allows you to read and write to the 24LC32 EEPROM. You can use this EEPROM with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library().

CircuitPython Microcontroller Wiring

First, wire up a 24LC32 to your board exactly as shown below. Here's an example of wiring a Feather M4 to the EEPROM with I2C using one of the handy STEMMA QT() connectors:

- Board 3V to EEPROM VIN (red wire)
- Board GND to EEPROM GND (black wire)
- Board SCL to EEPROM SCL (yellow wire)
- Board SDA to EEPROM SDA (blue wire)

You can also use standard 0.100" pitch headers to wire it up on a breadboard:
Python Computer Wiring

Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported.

Here's the Raspberry Pi wired to the EEPROM using I2C and a STEMMA QT

Pi 3V to EEPROM VIN (red wire)
Pi GND to EEPROM GND (black wire)
Pi SCL to EEPROM SCL (yellow wire)
Pi SDA to EEPROM SDA (blue wire)

Finally here is an example of how to wire up a Raspberry Pi to the EEPROM using a solderless breadboard:
Python Installation of 24LC32 Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready! Once that's done, from your command line run the following command:

```
    pip3 install adafruit-circuitpython-24lc32
```

If your default Python is version 3, you may need to run `pip` instead. Make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython Usage

To use with CircuitPython, you need to first install the 24LC32 library, and its dependencies, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders and file:

- `adafruit_bus_device/`
  - `adafruit_register/`
Python Usage

Once you have the library pip3 installed on your computer, copy or download the following example to your computer, and run the following, replacing code.py with whatever you named the file:

```
python3 code.py
```

Example Code

```python
# SPDX-FileCopyrightText: Copyright (c) 2021 Tim Cocks for Adafruit Industries
# # SPDX-License-Identifier: Unlicense

import board
import adafruit_24lc32

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a microcontroller
eeprom = adafruit_24lc32.EEPROM_I2C(i2c)

print("length: {}".format(len(eeprom)))

# eeprom[0] = 4
# print(eeprom[0])

# eeprom[0:4] = [9, 3, 8, 1]
# print(eeprom[0:4])
```

If running CircuitPython: Once everything is saved to the CIRCUITPY drive, connect to the serial console () to see the data printed out!

If running Python: The console output will appear wherever you are running Python.
In the example, the EEPROM is instantiated on I2C. Then, the EEPROM's available storage length is printed to the REPL. The EEPROM is internally organized as 4096 words of 8 bits each, or 4 Kbytes total.

**Read and Write**

In the example, uncomment the following lines:

```python
eeprom[0] = 4
print(eeprom[0])
eeprom[0:4] = [9, 3, 8, 1]
print(eeprom[0:4])
```

Then save and rerun the code. 4 is written to the EEPROM's index 0. The EEPROM's index 0 is then read and printed to the REPL in bytes. Then, a sequence of integers are written to the EEPROM and then read and printed to the REPL as bytes.

**Python Docs**

[Python Docs]()
Arduino

Using the 24LC32 EEPROM with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_FRAM_I2C library and running the provided example code.

Wiring

Wire as shown for a 5V board like an Uno. If you are using a 3V board, like an Adafruit Feather, wire the board's 3V pin to the 24LC32 VIN.

Here is an Adafruit Metro wired up to the 24LC32 using the STEMMA QT connector:

Board 5V to EEPROM VIN (red wire)
Board GND to EEPROM GND (black wire)
Board SCL to EEPROM SCL (yellow wire)
Board SDA to EEPROM SDA (blue wire)

Here is an Adafruit Metro wired up using a solderless breadboard:

Board 5V to EEPROM VIN (red wire)
Board GND to EEPROM GND (black wire)
Board SCL to EEPROM SCL (yellow wire)
Board SDA to EEPROM SDA (blue wire)
Library Installation

You can install the Adafruit EEPROM & FRAM I2C library for Arduino using the Library Manager in the Arduino IDE.

Click the Manage Libraries ... menu item, search for Adafruit FRAM I2C, and select the Adafruit FRAM I2C 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!
#include "Adafruit_EEPROM_I2C.h"

/* Example code for the Adafruit I2C EEPROM breakout */

/* Connect SCL to SCL
Connect SDA to SDA
Connect VDD to 3 - 5V DC
Connect GROUND to common ground */

Adafruit_EEPROM_I2C i2ceeprom;
#define EEPROM_ADDR 0x50  // the default address!

void setup(void) {
  Serial.begin(115200);
  if (i2ceeprom.begin(0x50)) {  // you can stick the new i2c addr in here, e.g. begin(0x51);
    Serial.println("Found I2C EEPROM");
  } else {
    Serial.println("I2C EEPROM not identified ... check your connections?\r\n");
    while (1) delay(10);
  }

  // Read the first byte
  uint8_t test = i2ceeprom.read(0x0);
  Serial.print("Restarted "); Serial.print(test); Serial.println(" times");
  // Test write ++
  test++;
  i2ceeprom.write(0x0, test);

  // Try to determine the size by writing a value and seeing if it changes the first byte
  Serial.println("Testing size!");

  uint32_t max_addr;
  for (max_addr = 1; max_addr < 0xFFFF; max_addr++) {
    if (i2ceeprom.read(max_addr) != test)
      continue; // def didnt wrap around yet

    // maybe wrapped? try writing the inverse
    if (! i2ceeprom.write(max_addr, (byte)~test)) {
      Serial.print("Failed to write address 0x");
      Serial.println(max_addr, HEX);
    }
  }

  // read address 0x0 again
  uint8_t val0 = i2ceeprom.read(0);

  // re-write the old value
  if (! i2ceeprom.write(max_addr, test)) {
    Serial.print("Failed to re-write address 0x");
    Serial.println(max_addr, HEX);
  }
}
// check if addr 0 was changed
if (val0 == (byte)~test) {
    Serial.println("Found max address");
    break;
}
Serial.print("This EEPROM can store ");
Serial.print(max_addr);
Serial.println(" bytes");

// dump the memory
uint8_t val;
for (uint16_t addr = 0; addr < max_addr; addr++) {
    val = i2ceeprom.read(addr);
    if ((addr % 32) == 0) {
        Serial.print("0x");Serial.print(addr, HEX);Serial.print(":");
    }
    Serial.print("0x");
    if (val < 0x10) Serial.print('0');
    Serial.print(val, HEX);Serial.print(" ");
}

void loop(void) {
}

Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You should see that the sketch has found your connected I2C EEPROM and the maximum storage size for the EEPROM in bytes. Then, the byte arrays stored on the EEPROM are printed to the Serial Monitor.
#include "Adafruit_EEPROM_I2C.h"
#include "Adafruit_FRAM_I2C.h"

/* Example code for the Adafruit I2C EEPROM/FRAM breakout */

/* Connect SCL to SCL
Connect SDA to SDA
Connect VDD to 3 - 5V DC
Connect GROUND to common ground */

Adafruit_EEPROM_I2C i2ceeprom;
//Adafruit_FRAM_I2C i2ceeprom;

#define EEPROM_ADDR 0x50 // the default address!

void setup(void) {
    Serial.begin(115200);
    if (i2ceeprom.begin(0x50)) { // you can stick the new i2c addr in here, e.g. begin(0x51);
        Serial.println("Found I2C EEPROM");
    } else {
        Serial.println("I2C EEPROM not identified ... check your connections?\r\n");
        while (1) delay(10);
    }

    float f = 3.141592;
    uint8_t buffer[4]; // floats are 4 bytes!
    memcpy(buffer, (void *)&f, 4);
    Serial.println("Writing float to address 0x00");
    i2ceeprom.write(0x00, buffer, 4);
    i2ceeprom.read(0x00, buffer, 4);
    memcpy((void *)&f, buffer, 4);
    Serial.print("Read back float value: ");
    Serial.println(f, 8);
}

void loop(void) {

}
Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You should see that the sketch has found your connected I2C EEPROM. Then, a float is written to the EEPROM's address 0x00 and is read back from the EEPROM.

Arduino Docs

Downloads

Files

- 24LC32 Datasheet ()
- EagleCAD PCB files on GitHub ()
- 3D Models on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()
Schematic and Fab Print