AdafruitFRAM Library Documentation

Release 1.0

Michael Schroeder

Jun 07, 2021

Contents

CircuitPython/Python library to support the I2C and SPI FRAM Breakouts.

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-fram
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-fram
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-fram
```

CHAPTER 3

Usage Example

See simpletest examples in the `/examples/` directory.

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/fram_i2c_simpletest.py

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

## 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
```

(continues on next page)

(continued from previous page)

```

## 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])
```

Listing 2: examples/fram_spi_simpletest.py

```
# 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])
```

6.2 adafruit_fram

CircuitPython/Python library to support the I2C and SPI FRAM Breakouts.

- Author(s): Michael Schroeder

6.2.1 Implementation Notes

Hardware:

- Adafruit I2C Non-Volatile FRAM Breakout (Product ID: 1895)
- Adafruit SPI Non-Volatile FRAM Breakout (Product ID: 1897)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_fram.**FRAM**(*max_size*, *write_protect=False*, *wp_pin=None*)
Driver base for the FRAM Breakout.

__getitem__(*address*)
Read the value at the given index, or values in a slice.

```
# read single index
fram[0]

# read values 0 thru 9 with a slice
fram[0:9]
```

__len__()
The size of the current FRAM chip. This is one more than the highest address location that can be read or written to.

```
fram = adafruit_fram.FRAM_xxx() # xxx = 'I2C' or 'SPI'

# size returned by len()
len(fram)

# can be used with range
for i in range(0, len(fram))
```

__setitem__(*address*, *value*)
Write the value at the given starting index.

```
# write single index
fram[0] = 1

# write values 0 thru 4 with a list
fram[0] = [0,1,2,3]
```

write_protected
The status of write protection. Default value on initialization is `False`.

When a `WP` pin is supplied during initialization, or using `write_protect_pin`, the status is tied to that pin and enables hardware-level protection.

When no `WP` pin is supplied, protection is only at the software level in this library.

write_wraparound
Determines if sequential writes will wrapaound highest memory address (`len(FRAM) - 1`) address. If `False`, and a requested write will extend beyond the maximum size, an exception is raised.

class adafruit_fram.**FRAM_I2C**(*i2c_bus*, *address=80*, *write_protect=False*, *wp_pin=None*)
I2C class for FRAM.

Param ~busio.I2C i2c_bus: The I2C bus the FRAM is connected to.

Param int address: I2C address of FRAM. Default address is `0x50`.

Param bool write_protect: Turns on/off initial write protection. Default is `False`.

Param wp_pin: (Optional) Physical pin connected to the `WP` breakout pin. Must be a `digitalio.DigitalInOut` object.

write_protected

The status of write protection. Default value on initialization is `False`.

When a `WP` pin is supplied during initialization, or using `write_protect_pin`, the status is tied to that pin and enables hardware-level protection.

When no `WP` pin is supplied, protection is only at the software level in this library.

class adafruit_fram.**FRAM_SPI**(*spi_bus*, *spi_cs*, *write_protect=False*, *wp_pin=None*, *baudrate=100000*, *max_size=8192*)

SPI class for FRAM.

Param ~busio.SPI spi_bus: The SPI bus the FRAM is connected to.

Param ~digitalio.DigitalInOut spi_cs: The SPI CS pin.

Param bool write_protect: Turns on/off initial write protection. Default is `False`.

Param wp_pin: (Optional) Physical pin connected to the `WP` breakout pin. Must be a `digitalio.DigitalInOut` object.

Parameters

- **baudrate** (*int*) – SPI baudrate to use. Default is `1000000`.
- **max_size** (*int*) – Size of FRAM in Bytes. Default is `8192`.

write_protected

The status of write protection. Default value on initialization is `False`.

When a `WP` pin is supplied during initialization, or using `write_protect_pin`, the status is tied to that pin and enables hardware-level protection.

When no `WP` pin is supplied, protection is only at the software level in this library.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Symbols

A

F

W