AdafruitAVRprog Library Documentation

Release 1.0

ladyada

Jun 07, 2021

Contents

Program your favorite AVR chips directly from CircuitPython with this handy helper class that will let you make stand-alone programmers right from your REPL. Should work with any/all AVR chips, via SPI programming. Tested with ATmega328, ATtiny85 and ATmega2560

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython

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-avrprog
```

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

```
sudo pip3 install adafruit-circuitpython-avrprog
```

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-avrprog
```

CHAPTER 3

Usage Example

See examples folder for full examples that progam various bootloaders onto chips.

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/avrprog_read_signature_simpletest.py

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

"""
Read Signature Test - All this does is read the signature from the chip to
check connectivity!
"""

import board
import busio
import pwmio
import adafruit_avrprog

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
avrprog = adafruit_avrprog.AVRprog()
avrprog.init(spi, board.D5)

# pylint: disable-msg=no-member
# we can generate an 6 MHz clock for driving bare chips too!
clock_pwm = pwmio.PWMOut(board.D9, frequency=6000000, duty_cycle=65536 // 2)
# pylint: enable-msg=no-member

avrprog.begin()
print("Signature bytes: ", [hex(i) for i in avrprog.read_signature()])
avrprog.end()
```

Listing 2: examples/avrprog_program_trinket85.py

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

"""
Trinket/Gemma (ATtiny85) programming example, be sure you have the '85 wired up so:
  Trinket Ground to CircuitPython GND
  Trinket USB to CircuitPythong USB or make sure the Trinket is powered by USB
  Pin 2 -> CircuitPython SCK
  Pin 1 -> CircuitPython MISO
  Pin 0 -> CircuitPython MOSI
  RESET  -> CircuitPython D5 (or change the init() below to change it!)
Drag "trinket_boot.hex" onto the CircuitPython disk drive, then open REPL!
"""

import board
import busio
import adafruit_avrprog

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
avrprog = adafruit_avrprog.AVRprog()
avrprog.init(spi, board.D5)

# Each chip has to have a definition so the script knows how to find it
attiny85 = avrprog.Boards.ATtiny85


def error(err):
    """ Helper to print out errors for us and then halt """
    print("ERROR: " + err)
    avrprog.end()
    while True:
        pass


while input("Ready to GO, type 'G' here to start> ") != "G":
    pass

if not avrprog.verify_sig(attiny85, verbose=True):
    error("Signature read failure")
print("Found", attiny85["name"])

avrprog.write_fuses(attiny85, low=0xF1, high=0xD5, ext=0x06, lock=0x3F)
if not avrprog.verify_fuses(attiny85, low=0xF1, high=0xD5, ext=0x06, lock=0x3F):
    error("Failure verifying fuses!")

print("Programming flash from file")
avrprog.program_file(attiny85, "trinket_boot.hex", verbose=True, verify=True)

print("Done!")
```

Listing 3: examples/avrprog_program_uno328.py

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

```

(continues on next page)

(continued from previous page)

```
"""
UNO Optiboot programming example, be sure you have the UNO wired up so:
  UNO Ground to CircuitPython GND
  UNO 5V to CircuitPython USB or make sure the UNO is powered by USB
  UNO Pin 13 -> CircuitPython SCK
  UNO Pin 12 -> CircuitPython MISO
  UNO Pin 11 -> CircuitPython MOSI
  UNO RESET  -> CircuitPython D5 (or change the init() below to change it!)
Drag "optiboot_atmega328.hex" onto the CircuitPython disk drive, then open REPL!
"""

import board
import busio
import pwmio
import adafruit_avrprog

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
avrprog = adafruit_avrprog.AVRprog()
avrprog.init(spi, board.D5)

# pylint: disable-msg=no-member
# we can generate an 6 MHz clock for driving bare chips too!
clock_pwm = pwmio.PWMOut(board.D9, frequency=6000000, duty_cycle=65536 // 2)
# pylint: enable-msg=no-member

# Each chip has to have a definition so the script knows how to find it
atmega328p = avrprog.Boards.ATmega328p


def error(err):
    """ Helper to print out errors for us and then halt """
    print("ERROR: " + err)
    avrprog.end()
    while True:
        pass


while input("Ready to GO, type 'G' here to start> ") != "G":
    pass

if not avrprog.verify_sig(atmega328p, verbose=True):
    error("Signature read failure")
print("Found", atmega328p["name"])

# Since we are unsetting the lock fuse, an erase is required!
avrprog.erase_chip()

avrprog.write_fuses(atmega328p, low=0xFF, high=0xDE, ext=0x05, lock=0x3F)
if not avrprog.verify_fuses(atmega328p, low=0xFF, high=0xDE, ext=0x05, lock=0x3F):
    error(
        "Failure programming fuses: "
        + str([hex(i) for i in avrprog.read_fuses(atmega328p)])
    )

print("Programming flash from file")
avrprog.program_file(atmega328p, "optiboot_atmega328.hex", verbose=True, verify=True)

```

(continues on next page)

(continued from previous page)

```
avrprog.write_fuses(atmega328p, lock=0x0F)
if not avrprog.verify_fuses(atmega328p, lock=0x0F):
    error("Failure verifying fuses!")

print("Done!")
```

Listing 4: examples/avrprog_program_mega2560.py

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

"""
Arduino Mega 2560 programming example, be sure you have the Mega/2560 wired up so:
  Mega Ground to CircuitPython GND
  Mega 5V to CircuitPython USB or make sure the Trinket is powered by USB
  Pin 52 -> CircuitPython SCK
  Pin 50 -> CircuitPython MISO  - Note this is backwards from what you expect
  Pin 51 -> CircuitPython MOSI  - Note this is backwards from what you expect
  RESET  -> CircuitPython D5 (or change the init() below to change it)
Drag "stk500boot_v2_mega2560.hex" onto the CircuitPython disk drive, then open REPL
"""

import board
import busio
import adafruit_avrprog

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
avrprog = adafruit_avrprog.AVRprog()
avrprog.init(spi, board.D5)

# To program a chip, you'll need to find out the signature, size of the flash,
# flash-page size and fuse mask. You can find this in the datasheet or in
# avrdude.conf located at:
# http://svn.savannah.nongnu.org/viewvc/*checkout*/avrdude/trunk/avrdude/avrdude.conf.
↪in
# You can also use the predefined values in AVRprog.Boards
atmega2560 = {
    "name": "ATmega2560",
    "sig": [0x1E, 0x98, 0x01],
    "flash_size": 262144,
    "page_size": 256,
    "fuse_mask": (0xFF, 0xFF, 0x07, 0x3F),
}


def error(err):
    """ Helper to print out errors for us and then halt """
    print("ERROR: " + err)
    avrprog.end()
    while True:
        pass


while input("Ready to GO, type 'G' here to start> ") != "G":
    pass

```

(continues on next page)

(continued from previous page)

```
if not avrprog.verify_sig(atmega2560, verbose=True):
    error("Signature read failure")
print("Found", atmega2560["name"])

# Since we are unsetting the lock fuse, an erase is required!
avrprog.erase_chip()

avrprog.write_fuses(atmega2560, low=0xFF, high=0xD8, ext=0x05, lock=0x3F)
if not avrprog.verify_fuses(atmega2560, low=0xFF, high=0xD8, ext=0x05, lock=0x3F):
    error(
        "Failure programming fuses: "
        + str([hex(i) for i in avrprog.read_fuses(atmega2560)])
    )

print("Programming flash from file")
avrprog.program_file(
    atmega2560, "stk500boot_v2_mega2560.hex", verbose=True, verify=True
)

avrprog.write_fuses(atmega2560, lock=0x0F)
if not avrprog.verify_fuses(atmega2560, lock=0x0F):
    error("Failure verifying fuses!")

print("Done!")
```

6.2 adafruit_avrprog

Program your favorite AVR chips directly from CircuitPython with this handy helper class that will let you make stand-alone programmers right from your REPL

- Author(s): ladyada

6.2.1 Implementation Notes

Hardware:

- See Learn Guide for supported hardware: Stand-alone programming AVRs using CircuitPython

Software and Dependencies:

- Adafruit CircuitPython firmware for the ESP8622 and M0-based boards: https://github.com/adafruit/circuitpython/releases

class adafruit_avrprog.**AVRprog**

Helper class used to program AVR chips from CircuitPython.

class Boards

Some well known board definitions.

begin(*clock=1000000*)

Begin programming mode: pull reset pin low, initialize SPI, and send the initialization command to get the AVR's attention.

end()

End programming mode: SPI is released, and reset pin set high.

erase_chip()
Fully erases the chip.

init(*spi_bus*, *rst_pin*)
Initialize the programmer with an SPI port that will be used to communicate with the chip. Make sure your SPI supports 'write_readinto' Also pass in a reset pin that will be used to get into programming mode

program_file(*chip*, *file_name*, *verbose=False*, *verify=True*)
Perform a chip erase and program from a file that contains Intel HEX data. Returns true on verify-success, False on verify-failure. If 'verify' is False, return will always be True

read(*addr*, *read_buffer*)
Read a chunk of memory from address 'addr'. The amount read is the same as the size of the bytearray 'read_buffer'. Data read is placed directly into 'read_buffer' Requires calling begin() beforehand to put in programming mode.

read_fuses(*chip*)
Read the 4 fuses and return them in a list (low, high, ext, lock) Each fuse is bitwise-&'s with the chip's fuse mask for simplicity

read_signature()
Read and return the signature of the chip as two bytes in an array. Requires calling begin() beforehand to put in programming mode.

verify_file(*chip*, *file_name*, *verbose=False*)
Perform a chip full-flash verification from a file that contains Intel HEX data. Returns True/False on success/fail.

verify_fuses(*chip*, *low=None*, *high=None*, *ext=None*, *lock=None*)
Verify the 4 fuses. If the kwarg low/high/ext/lock is not passed in or is None, that fuse is not checked. Each fuse is bitwise-&'s with the chip's fuse mask. Returns True on success, False on a fuse verification failure

verify_sig(*chip*, *verbose=False*)
Verify that the chip is connected properly, responds to commands, and has the correct signature. Returns True/False based on success

write_fuses(*chip*, *low=None*, *high=None*, *ext=None*, *lock=None*)
Write any of the 4 fuses. If the kwarg low/high/ext/lock is not passed in or is None, that fuse is skipped

adafruit_avrprog.**read_hex_page**(*file_state*, *page_addr*, *page_size*, *page_buffer*)
Helper function that does the Intel Hex parsing. Takes in a dictionary that contains the file 'state'. The dictionary should have file_state['f'] be the file stream object (returned by open), the file_state['line'] which tracks the line number of the file for better debug messages. This function will update 'line' as it reads lines. It will set 'eof' when the file has completed reading. It will also store the 'extended address' state in file_state['ext_addr'] In addition to the file, it takes the desired buffer address start (page_addr), size (page_size) and an allocated bytearray. This function will try to read the file and fill the page_buffer. If the next line has data that is beyond the size of the page_address, it will return without changing the buffer, so pre-fill it with 0xFF before calling, for sparsely-defined HEX files. Returns False if the file has no more data to read. Returns True if we've done the best job we can with filling the buffer and the next line does not contain any more data we can use.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index