Supercharge Your Hardware

(old and new)

w/CircuitPython

Hi

Scott Shawcroft aka *tannewt*

Freelance software engineer

Project lead on CircuitPython for Adafruit

Plan

- CircuitPlayground Demo
- Vertical slice of CircuitPython
  - Software
  - Hardware
- GameBoy Demo

CircuitPlayground Demo

CircuitPython

Code + Community

Code + Community

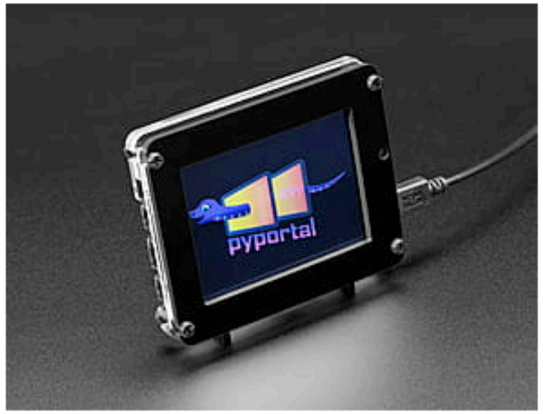
- Python is the easiest way to iterate on software
- CircuitPython code and toolchain travels with the device for ultimate hackability
- `https://github.com/adafruit/circuitpython`
- Built on MicroPython

Code **+ Community**

- Code of Conduct
- Active community on Discord and GitHub
- 190+ CircuitPython-compatible libraries
- 83+ Supported boards

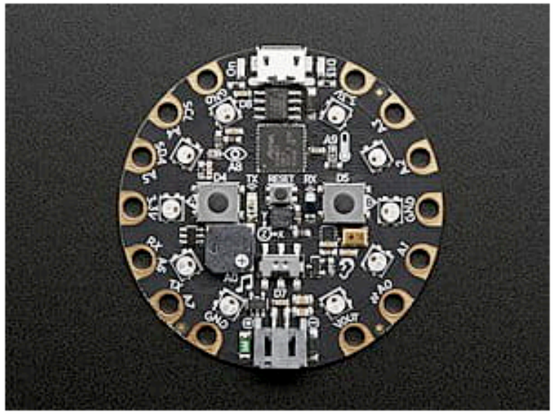
Downloads

Search for CircuitPython boards



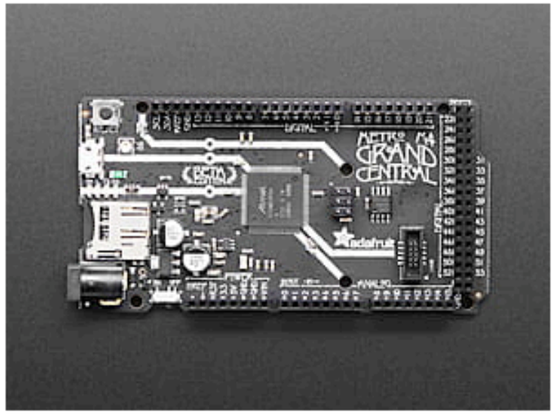
PyPortal

By Adafruit



Circuit Playground Express

By Adafruit

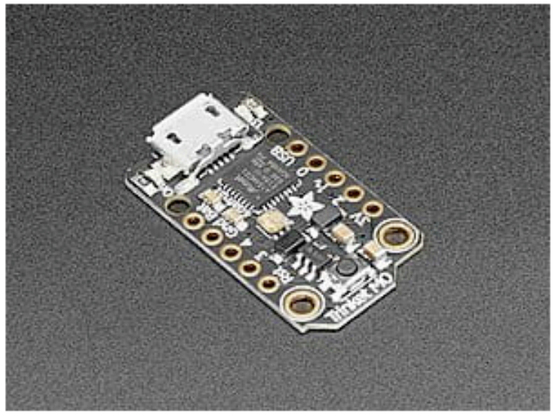


Grand Central M4 Express

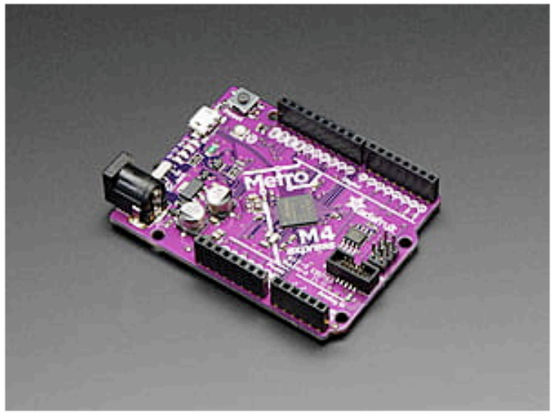
By Adafruit

NeoTrellis M4

By Adafruit

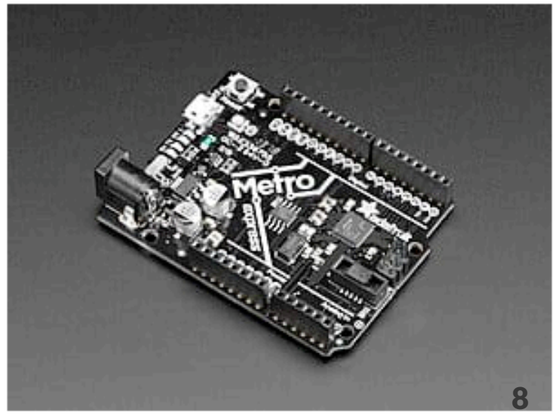


Trinket M0



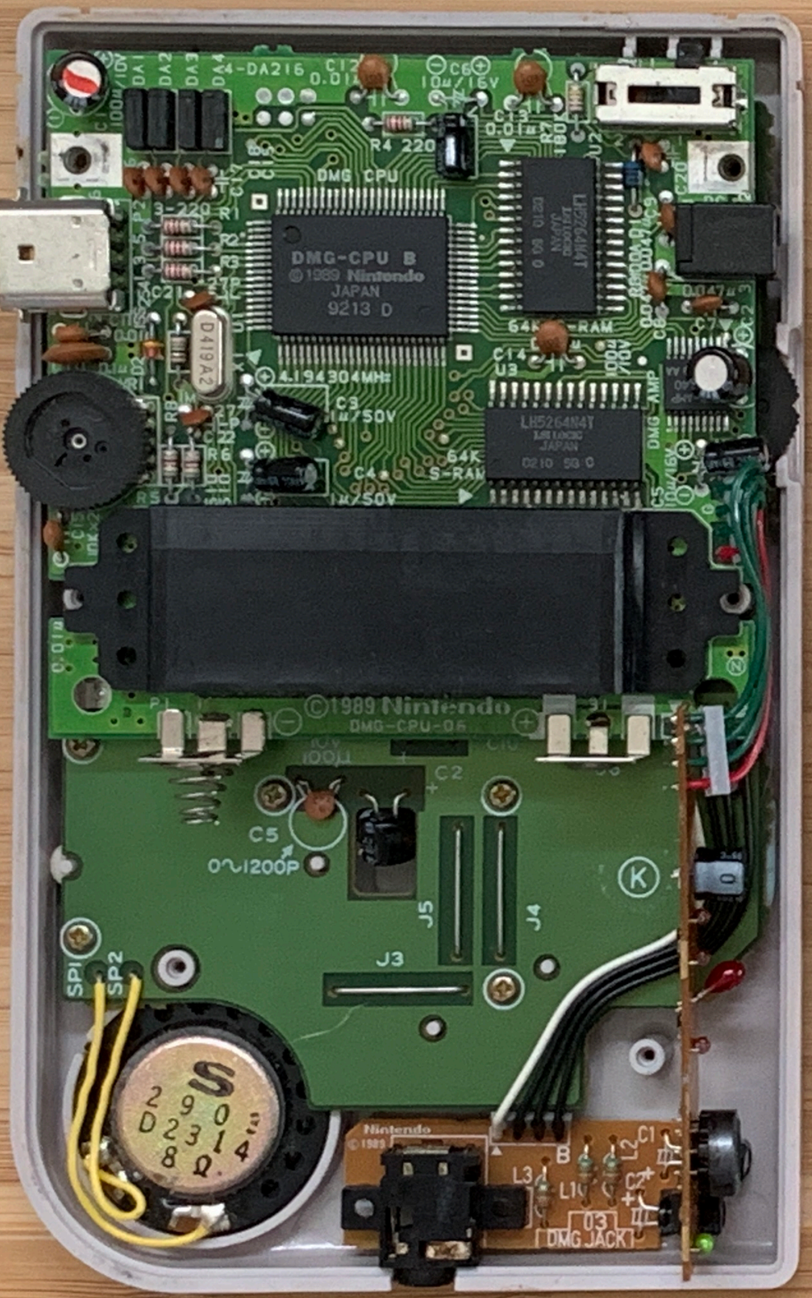
Metro M4 Express

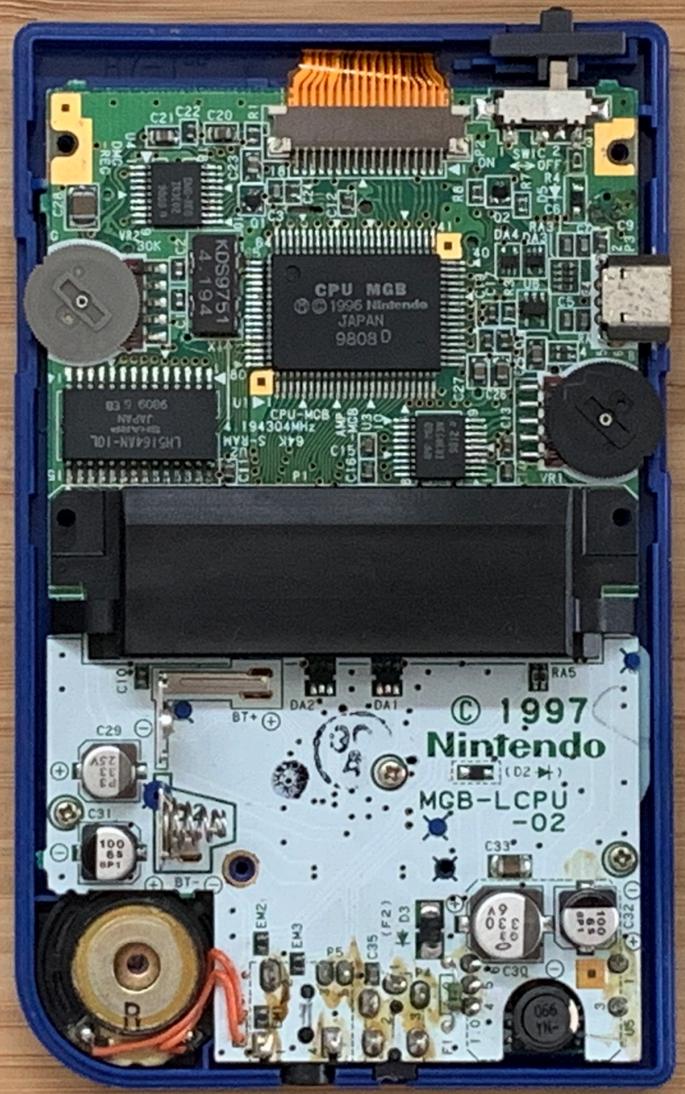
PyBadge



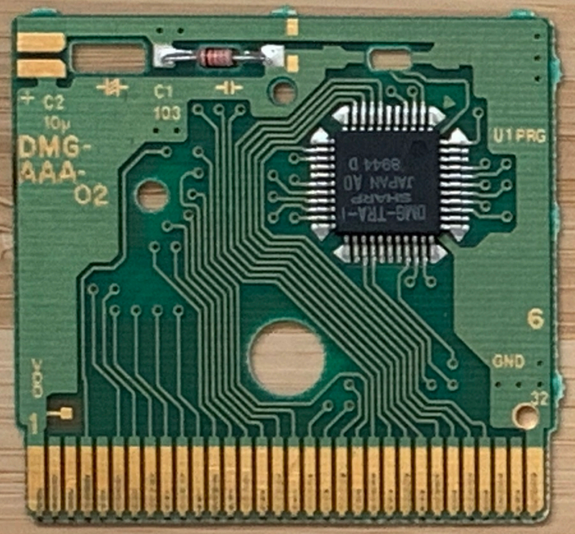
Metro M0 Express


DOT MATRIX WITH STEREO SOUND
BATTERY
Nintendo GAME BOY™
SELECT
START
B
A
POWER
GAME BOY pocket
SELECT START
POWER
GAME BOY COLOR

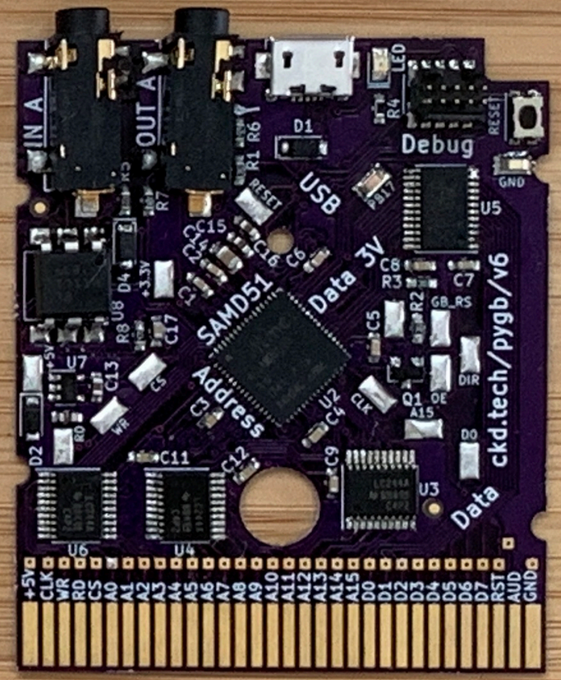

DMG CPU
DMG-CPU B
©1989 Nintendo
JAPAN
9213 D
4.194304MHz
64K S-RAM
©1989 Nintendo
DMG-CPU-06
C5
0~1200P
J5
J4
J3
SP1
SP2
DMG JACK


CPU MGB
©1996 Nintendo
JAPAN
9808 D
CPU-MGB
4.194304MHz
64K S-RAM
© 1997
Nintendo
MGB-LCPU
-02


CPU CGB D
©1998 Nintendo
JAPAN
0030 I
©1998
Nintendo
CGB-CPU-05

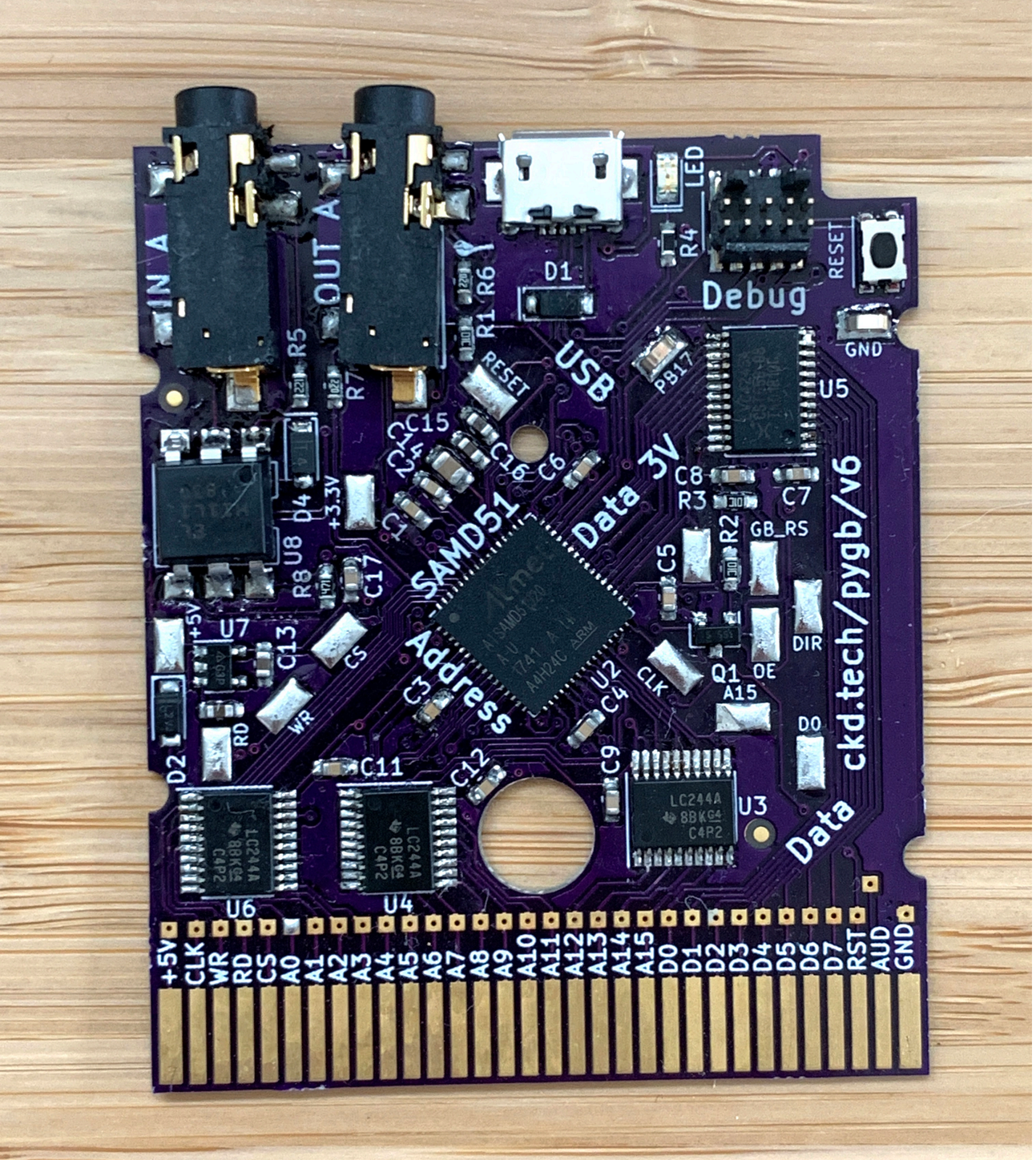

DMG-AAA-02
U1 PRG
GND


DMG-KFCN-01
Batt CR2025
MBC3A
U3 RAM
VDD
GND


IN A
OUT A
Debug
USB
SAMD51
Address
Data 3V
Data
ckd.tech/pygb/v6


Nintendo GAME BOY
TETRIS
DMG-TR-USA
THIS SIDE OUT
MADE IN JAPAN


mary-kateandashley
Pocket Planner
DMG-BMAE-USA
THIS SIDE OUT



MCU as Cart

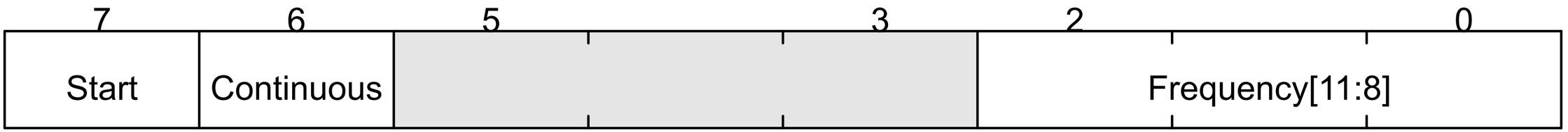
- DHole in late 2014 used an STM32F4 as a GameBoy cart.[1]
- Respond with a byte on the data bus for every 1 MHz clock where the address is in cart range (`0x0000-0x7fff`)
- Chose to use the SAMD51 because it is 120 MHz and already had CircuitPython support

[1] `https://dhole.github.io/post/gameboy_cartridge_emu_1/`

Layers

- Python Libraries - Expands the lowest level to simplify it
- Lowest Python - Barest Python that has no dependencies
- C <-> Python - hooks uniform C API to uniform Python API
- Lowest C - does time critical, chip-specific hardware interfacing
- The wire to the GameBoy CPU

The magic bit in `0xff14`



code.py

```
from adafruit_gameboy import gb

# Register's are documented here: http://marc.rawer.de/Gameboy/Docs/GBCPUman.pdf

offset = 0xff10 # Voice 1

# Voice 1
#   - Bit 7 - Start
#   - Bit 6 - Counter/consecutive
#   - Bit 2-0 - Top 3 frequency bits
gb[offset + 4] = 0b10000111 # 0x87
```

code.py

```
from adafruit_gameboy import gb

# Register's are documented here: http://marc.rawer.de/Gameboy/Docs/GBCPUman.pdf

offset = 0xff10 # Voice 1

# Voice 1
#   - Bit 7 - Start
#   - Bit 6 - Counter/consecutive
#   - Bit 2-0 - Top 3 frequency bits
gb[offset + 4] = 0b10000111 # 0x87
```

Some layers later...

ports/atmel-samd/common-hal/gbio/__init__.c

```
void common_hal_gbio_queue_commands(const uint8_t* buf, uint32_t len) {
    // Wait for a previous sequence to finish.

    uint32_t total_len = 0;

    memcpy(command_cache + 2, buf, len);
    total_len += len;

    // Start DMA and wait for it.
    DmacDescriptor* descriptor_out = dma_descriptor(dma_out_channel);
    descriptor_out->BTCTRL.reg |= DMAC_BTCTRL_VALID;
    descriptor_out->BTCNT.reg = total_len;
    descriptor_out->SRCADDR.reg = ((uint32_t) command_cache) + total_len;
    descriptor_out->DSTADDR.reg = (uint32_t)&PORT->Group[0].OUT.reg + 2;

    dma_enable_channel(dma_out_channel);

    // Wait for DMA
}
```

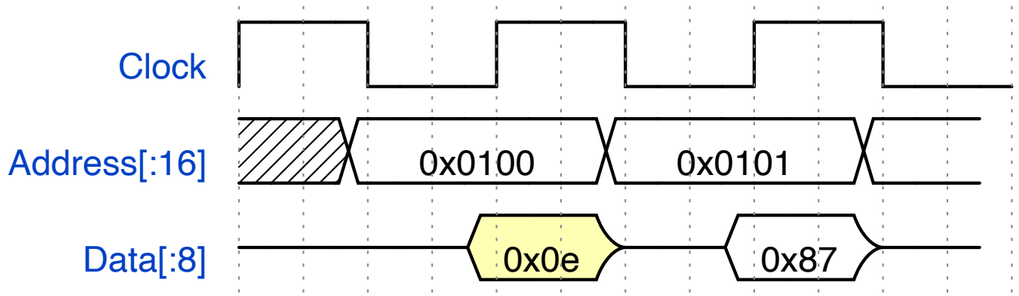
DMA Trigger

Use the SAMD51 CCLUT to do:

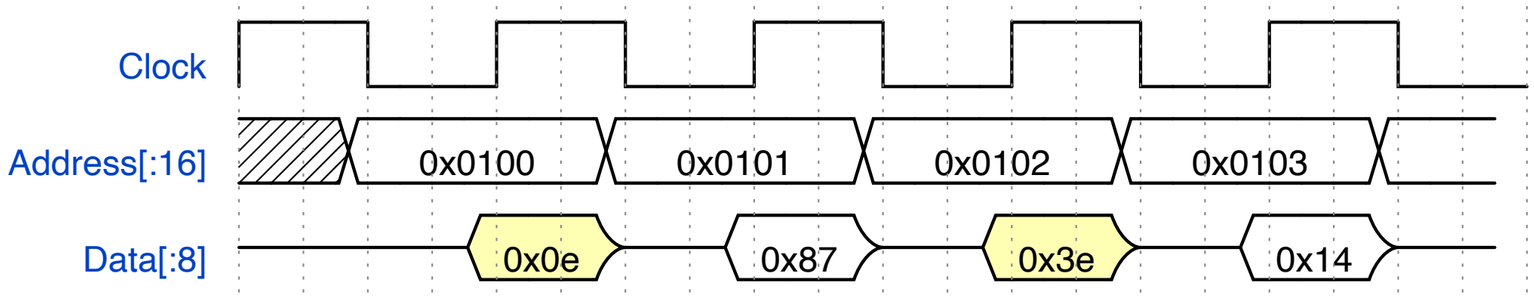
A15 | Read | Clock

Then drive the DMA and the level shifter output enable based on it.

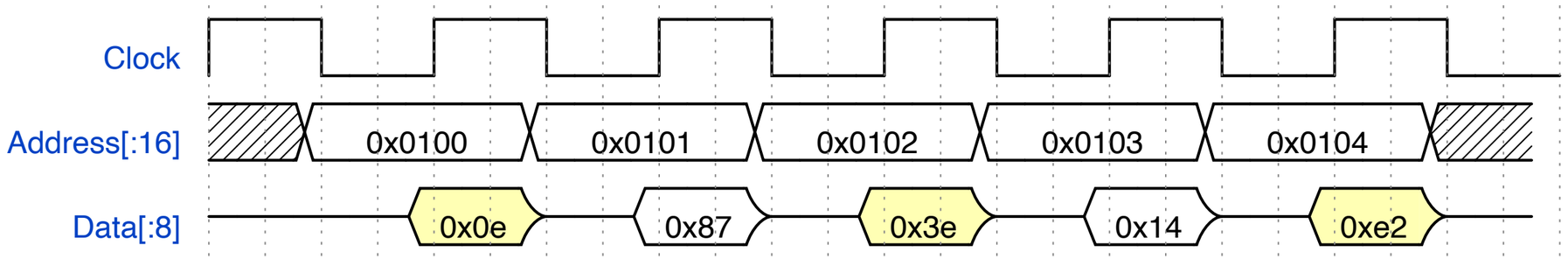
Load our new value into register A



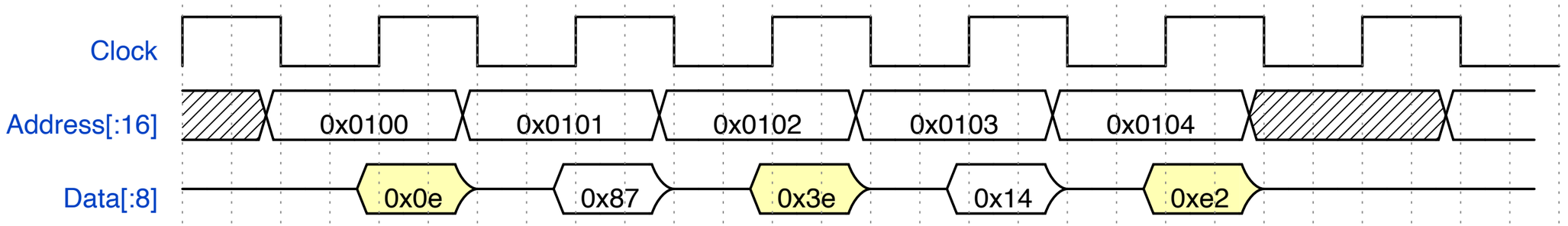
Load address into register C



Store A into 0xff00 + C

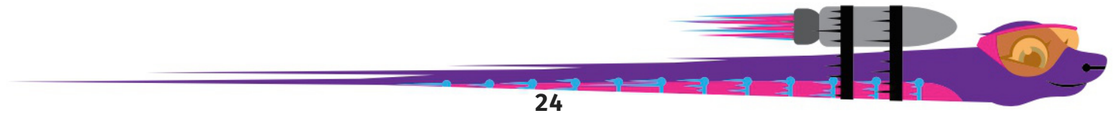


Continue


Clock
Address[:16]
Data[:8]
0x0100
0x0101
0x0102
0x0103
0x0104
0x0e
0x87
0x3e
0x14
0xe2

GameBoy Demo

https://media.ccc.de/v/33c3-8029-the_ultimate_game_boy_talk



Takeaways

- CircuitPython is the easiest way to program hardware.
- C can be used to do anything from Python.
- Existing hardware can benefit from CircuitPython.

Get started!

- We can help you get CircuitPython on **your** hardware.
  - Supported: SAMD21, SAMD51, nRF52840, STM32F4
  - Coming soon: iMX RT
- Reach out on our Discord chat: https://adafru.it/discord
- Code: https://github.com/adafruit/circuitpython

Contact

@tannewt

scott@adafruit.com

circuitpython.org

Thank you!