ADC21

FROM THE GROUND UP:

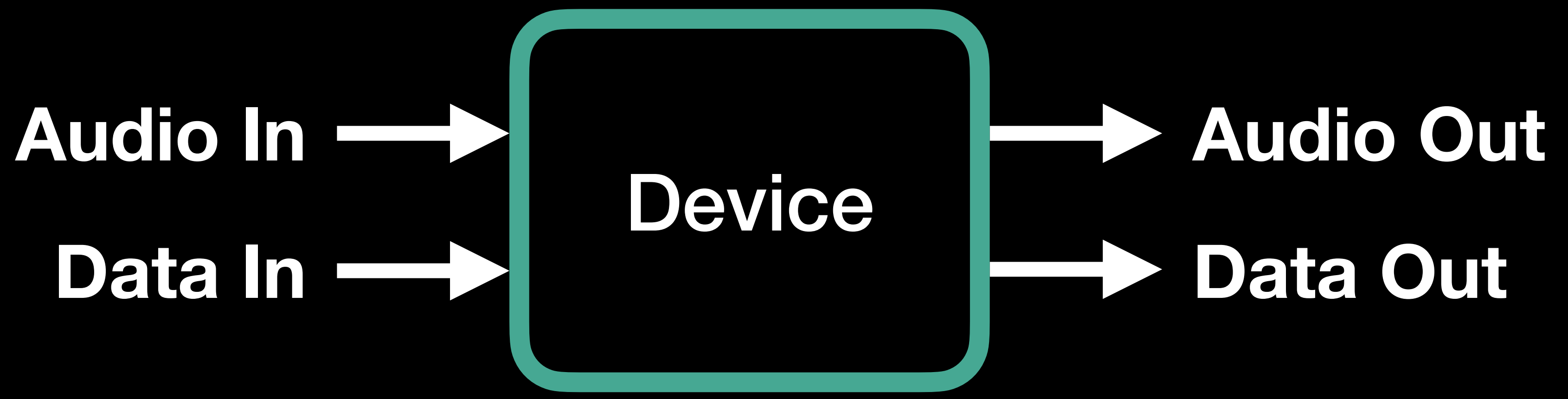
DEVELOPING AUDIO HARDWARE FROM SCRATCH

ALLEN LEE

Overview

What to Expect

- Introduction to building *digital* audio HW + FW



Examples:
Guitar Pedals
Audio Interfaces
Synthesizers

- Overview of:
  - Circuit building blocks
  - Schematic and PCB layout
  - Firmware architecture
  - Audio processing tips
  - Debugging and profiling bare-metal code

What NOT to Expect

- How to build analog hardware
- Comprehensive guide to HW + FW development

Allen Lee

Background Experience

Apple [3 Years]
Sensor Calibration Systems Development

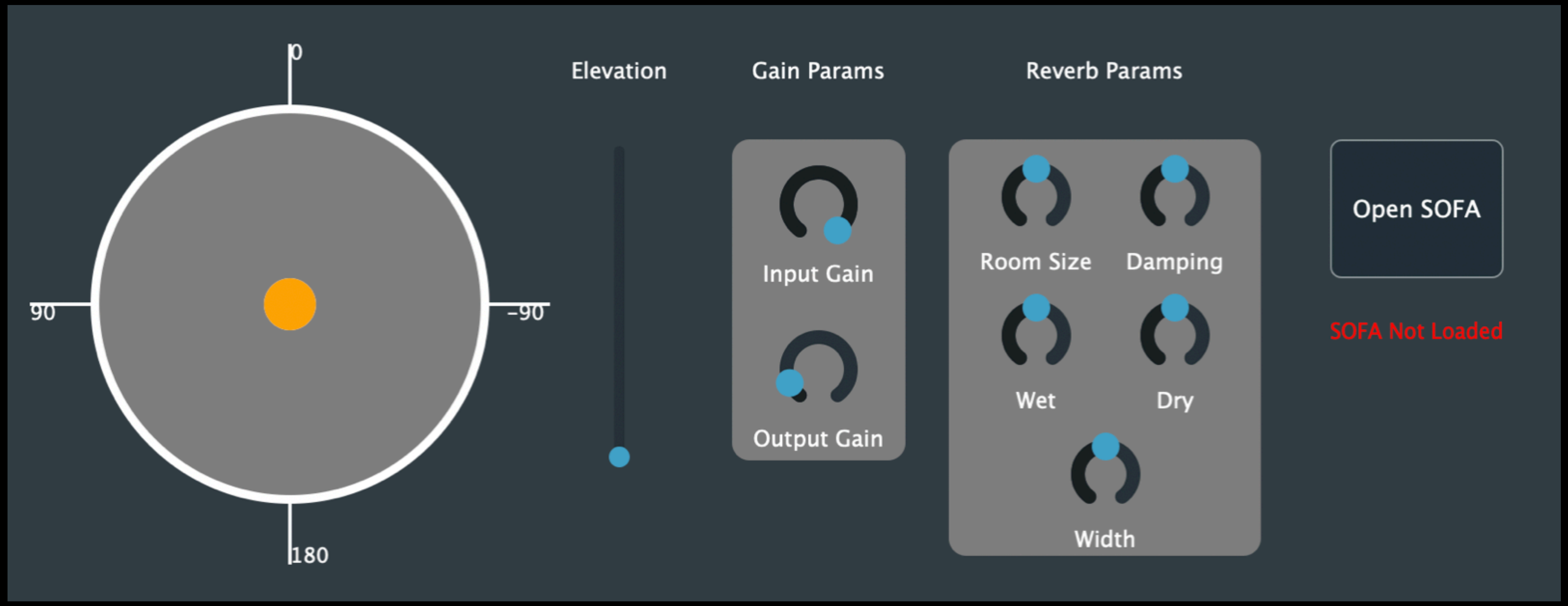
SMART Technologies [3 Years]
Automated Test Equipment (ATE) Development

Independent Audio Developer

Focusing on:

Spatial Audio
Bare-metal Audio Development

Orbiter
Spatial Audio Plugin

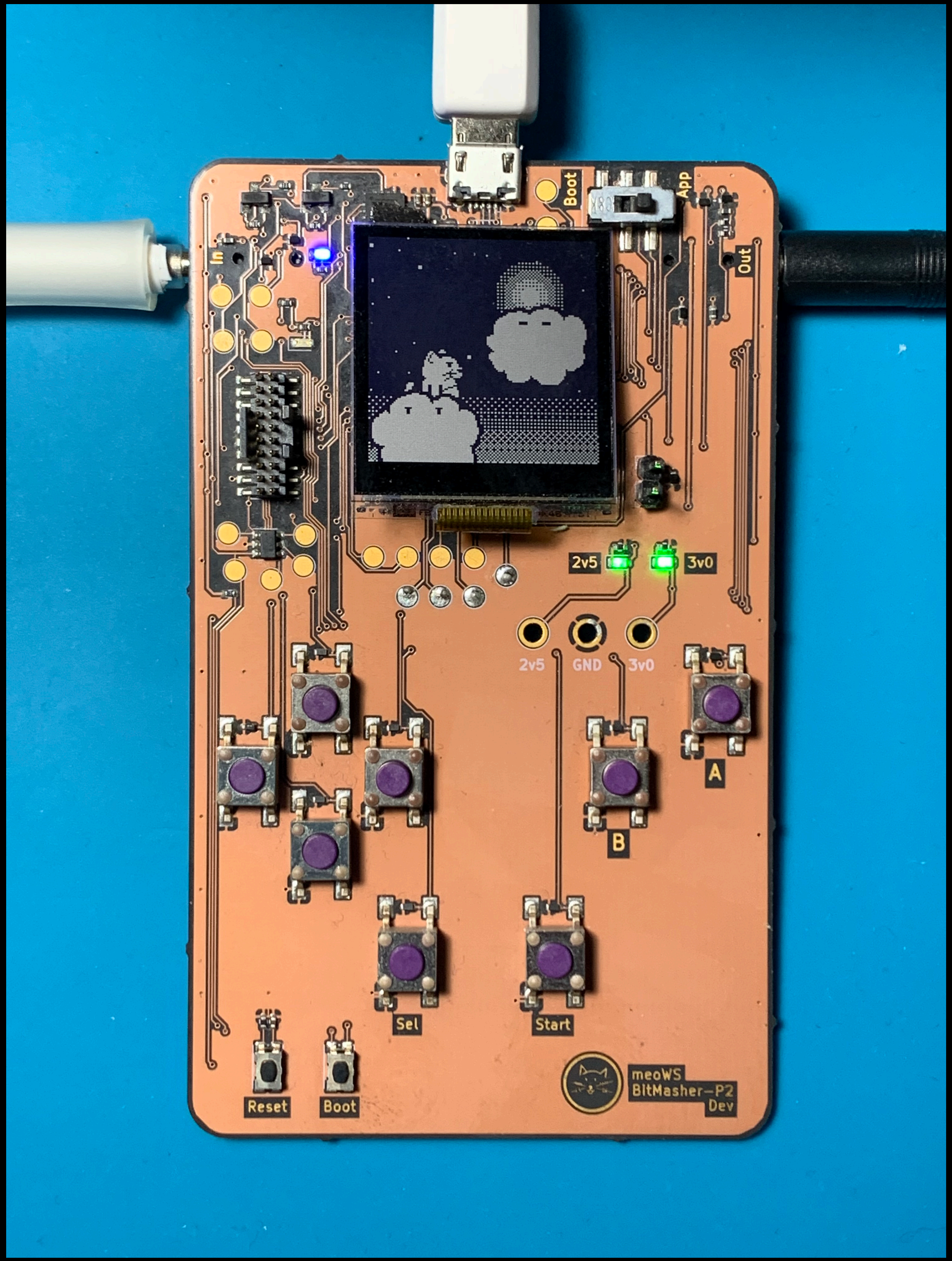


@superkittens
alee@meoworkshop.org

BitMasher

Introduction

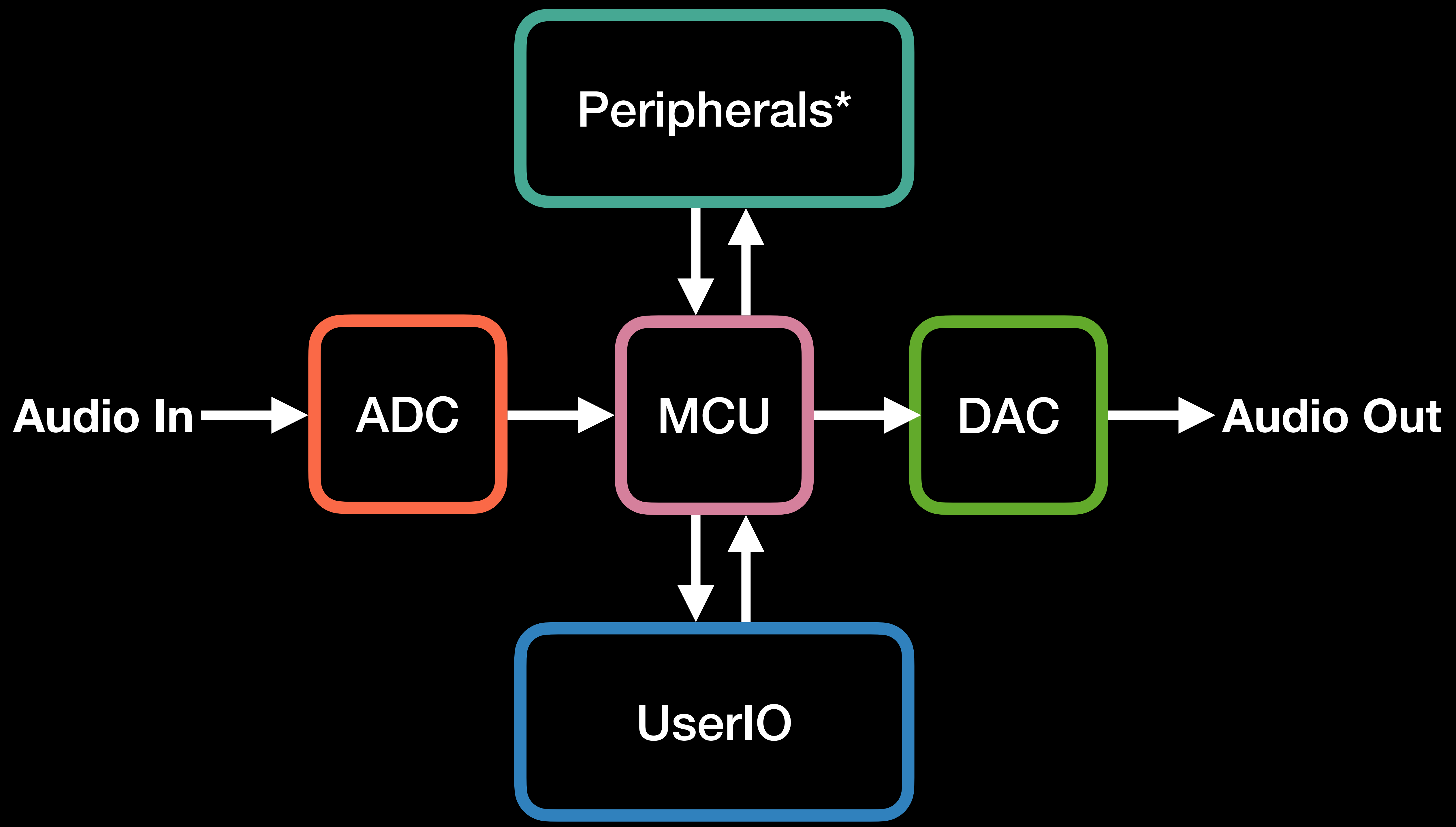
- Audio effects device with retro-game inspired UI
- 5 Main Effects:
  - Filters (LPF, HPF, BPF)
  - “Revolving Loudspeaker” Simulation
  - Bit Crusher
  - “Tape Playback” Simulation
  - Granular



Overview

Basic Architecture

- BitMasher's architecture is similar to those found in many audio products
- Specific architectural details differ between products but the basic form is generally the same
- Many of the design steps in BitMasher apply to other audio products!

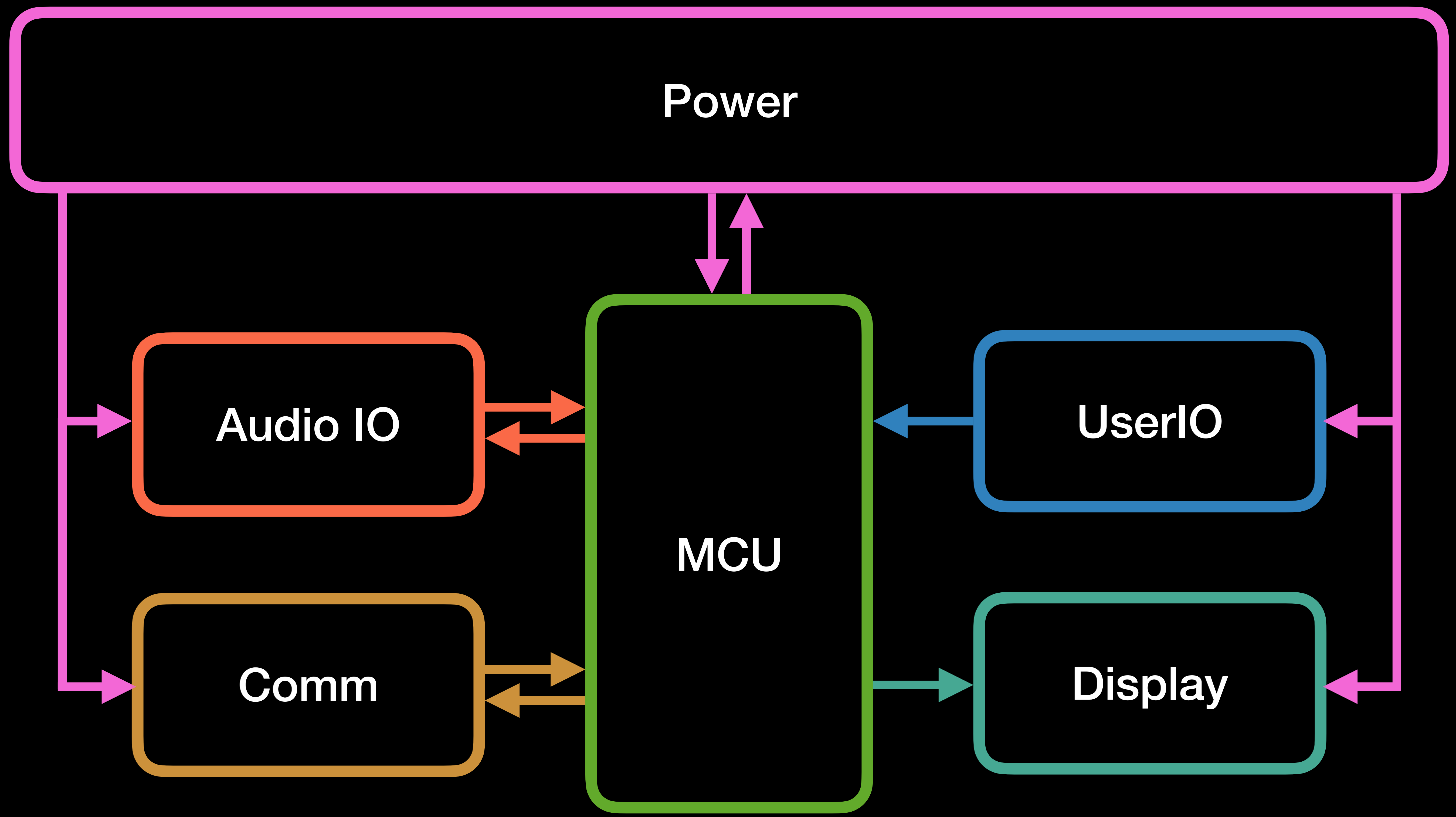


***Peripherals may include:**
Non-volatile storage
Communication to Host PC
MIDI Signalling
Display
etc

The Hardware

Hardware

BitMasher HW Block Diagram

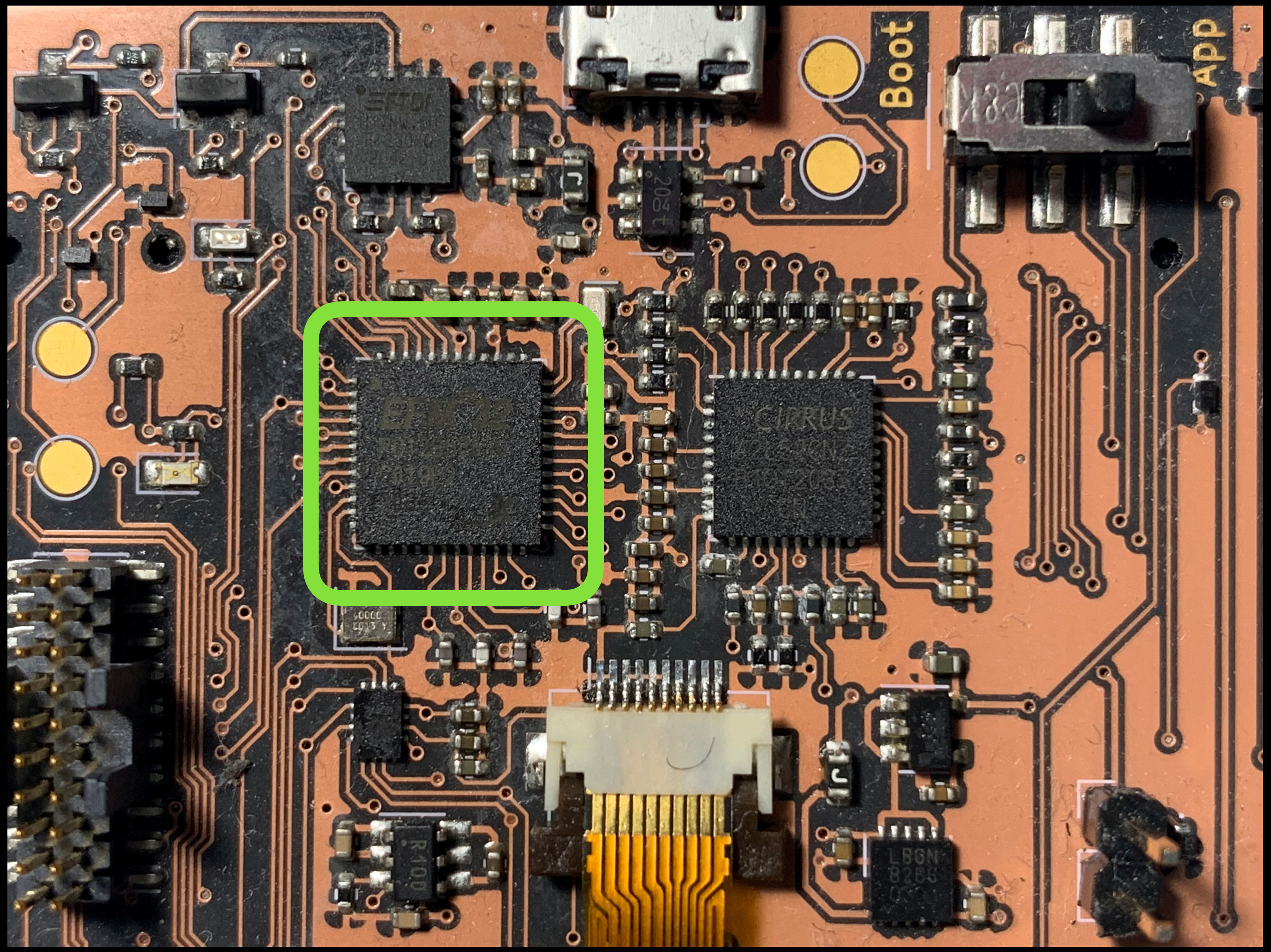


Microcontroller (MCU)

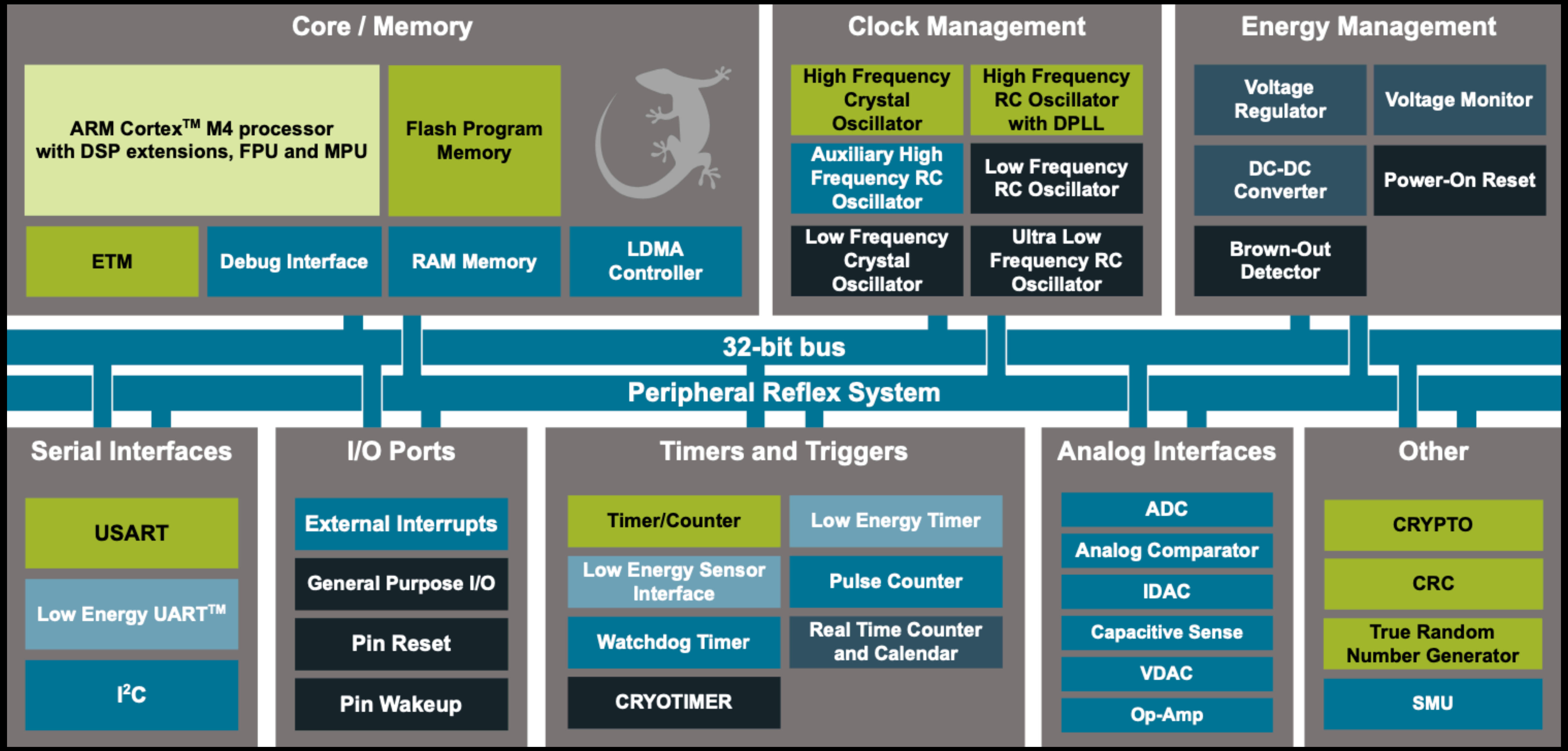
Overview

- The “brains” of your audio hardware
- Combination of one (or more) CPU cores and peripherals in one package
- Less powerful than desktop/laptop processors but can still do lots with them!
- BitMasher uses the Silicon Labs Pearl Gecko family of MCUs

BitMasher PCB



EFM32PG12 Layout



Feature chart from EFM32PG12 reference manual

Microcontroller

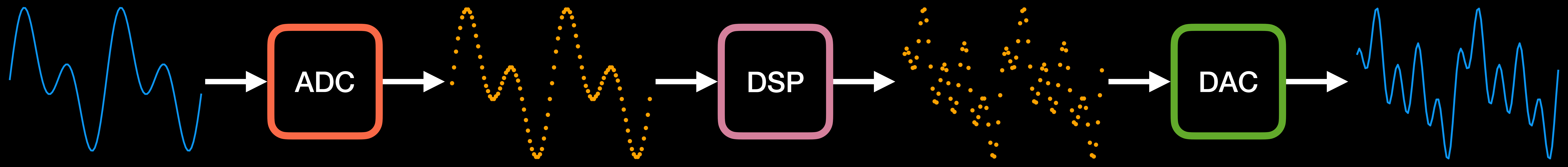
Microcontroller Selection

- There is a *huge* choice of microcontrollers which can make MCU selection difficult!
- The following (non-exhaustive) selection criteria can help narrow your choices

Criteria	Notes
Architecture	8-bit systems good for simple tasks, 32-bit systems good for more complex tasks (e.g. DSP)
Peripherals	Will you need ADCs? DACs? Hardware Timers? Etc
Memory	Audio DSP routines typically need lots of memory. MCUs with large RAM will be helpful
Power	If your project is battery powered, consider low energy MCUs
Availability + Vendor Support	Is the part easily available? Does the vendor offer good support? Do forums exist?
Cost	$$$

Audio IO

- Convert analog audio signals to digital ones
- Send digitized signals to MCU
- Receive processed data from MCU
- Convert digital audio signal to analog
- Few ways to implement this chain

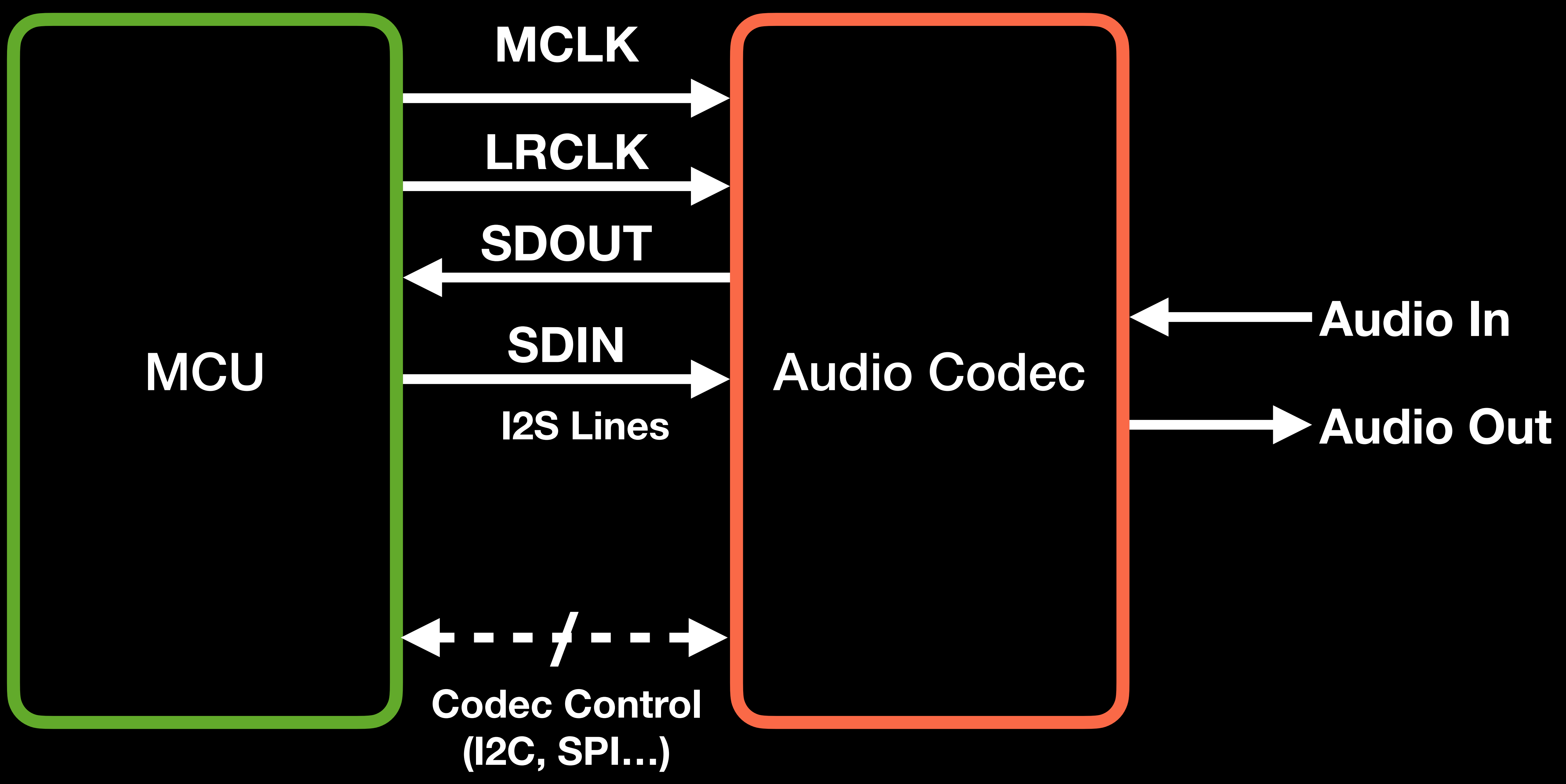


Dedicated ADCs/DACs

Overview

- A number of dedicated ADC and DAC ICs are available
- Some combine both functions (typically called *audio codecs*)
- Audio data transfer to the MCU is often through the I2S protocol

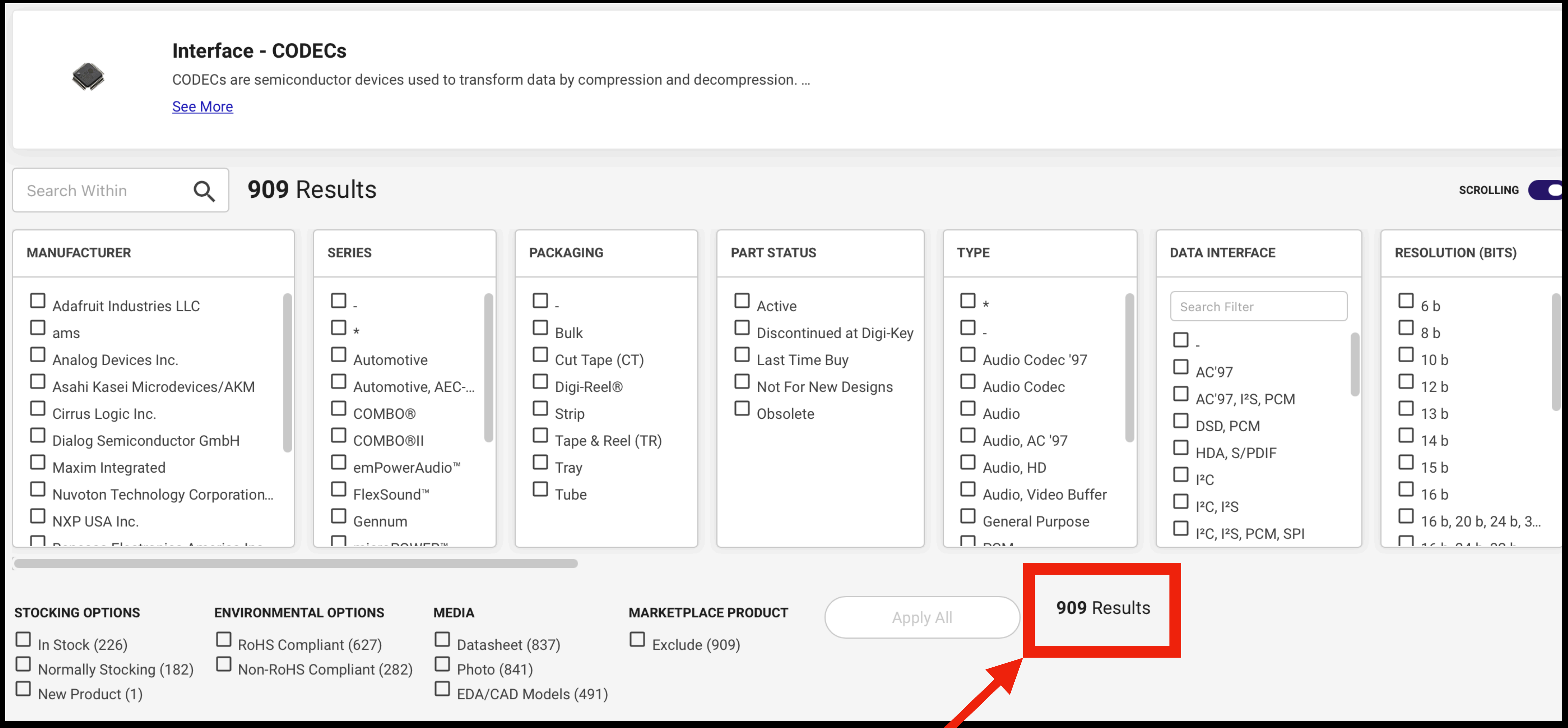
Audio Codec Typical Connection



Dedicated ADCs/DACs

Codec Selection Criteria

- There is also a large choice of audio codecs!
- Therefore, down-selecting potential codecs depends on different criteria such as:
  - Bit-depth
  - Sampling Frequency
  - Power Consumption
  - Footprint
  - Noise Performance
  - Cost and Availability
  - Vendor Support



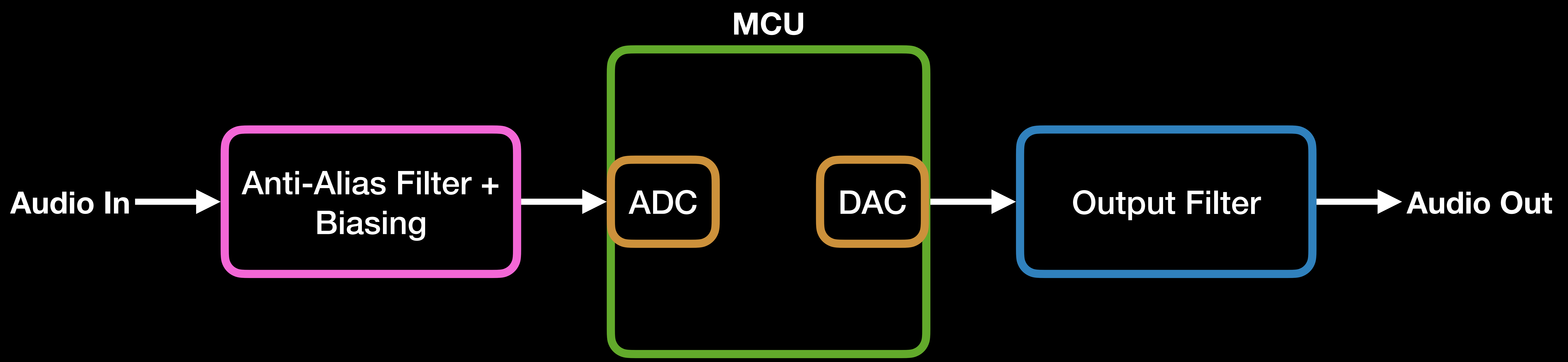
A search for *audio codec* on Digikey returned 909 results!!

Microcontroller ADCs/DACs

Overview

- A MCU's internal ADC *can* be used to convert audio signals
- Many MCUs do not feature a DAC but if yours does, they can also be used!
- There are however, some additional circuitry that *may* be needed
- Note that many internal ADC/DACs do not have the bit-depth of dedicated audio ADCs/DACs

Audio Signal Flow Using Internal ADCs and DACs

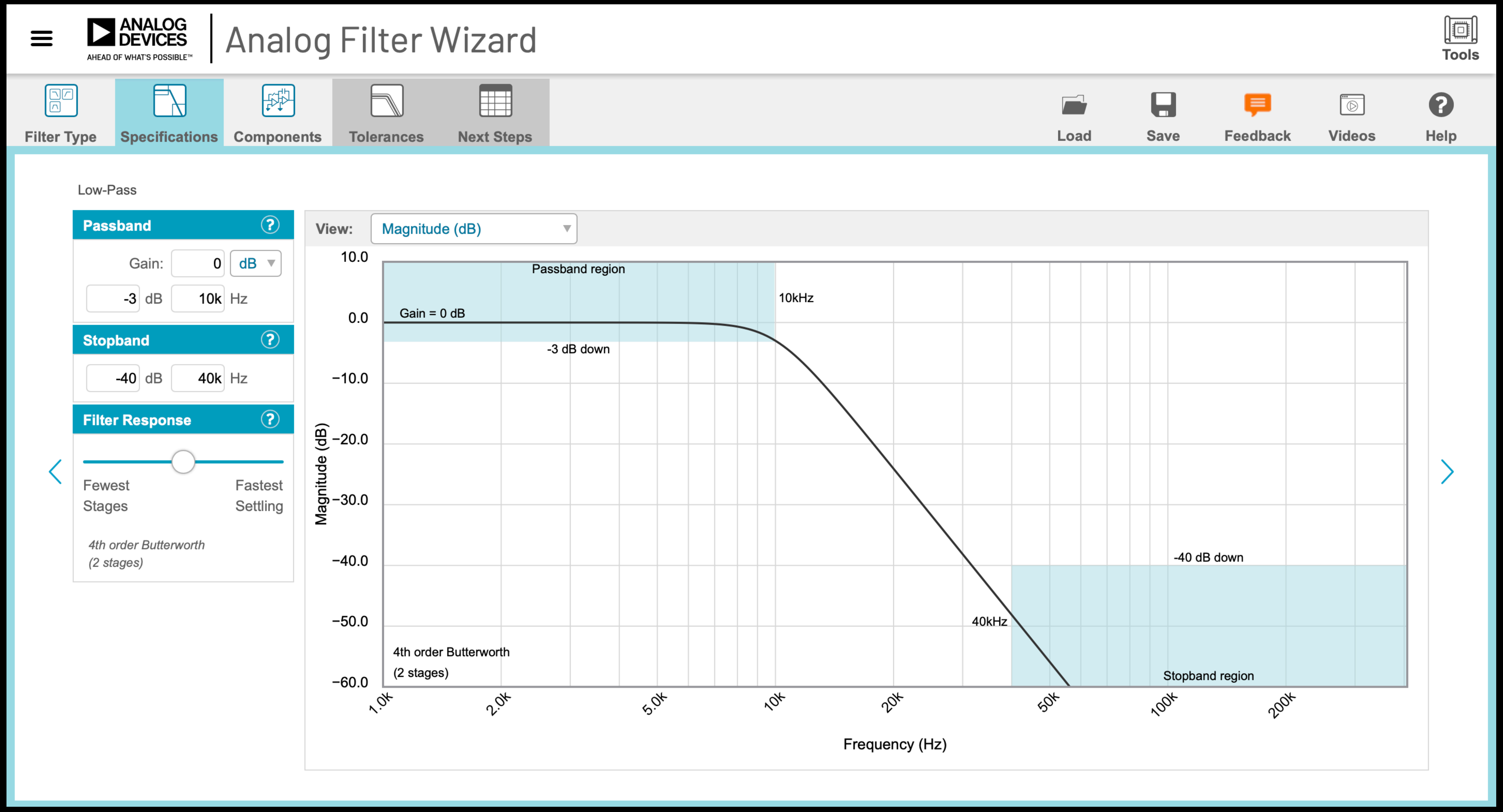


Microcontroller ADCs/DACs

ADC Input Anti-Alias Filter

- When converting an analog signal, usually some form of low-pass anti-aliasing filters are needed
- Active filters recommended over passive ones
- Many excellent online filter design tools are available!
- Oversampling is another option but may be prohibitive depending on processor capabilities

Analog Devices Filter Design Tool



https://tools.analog.com/en/filterwizard/

Minimum Stop Band Attenuation Formula

$$SB_{dB} = 20\log(2^{-N})$$

N = ADC Bit Depth

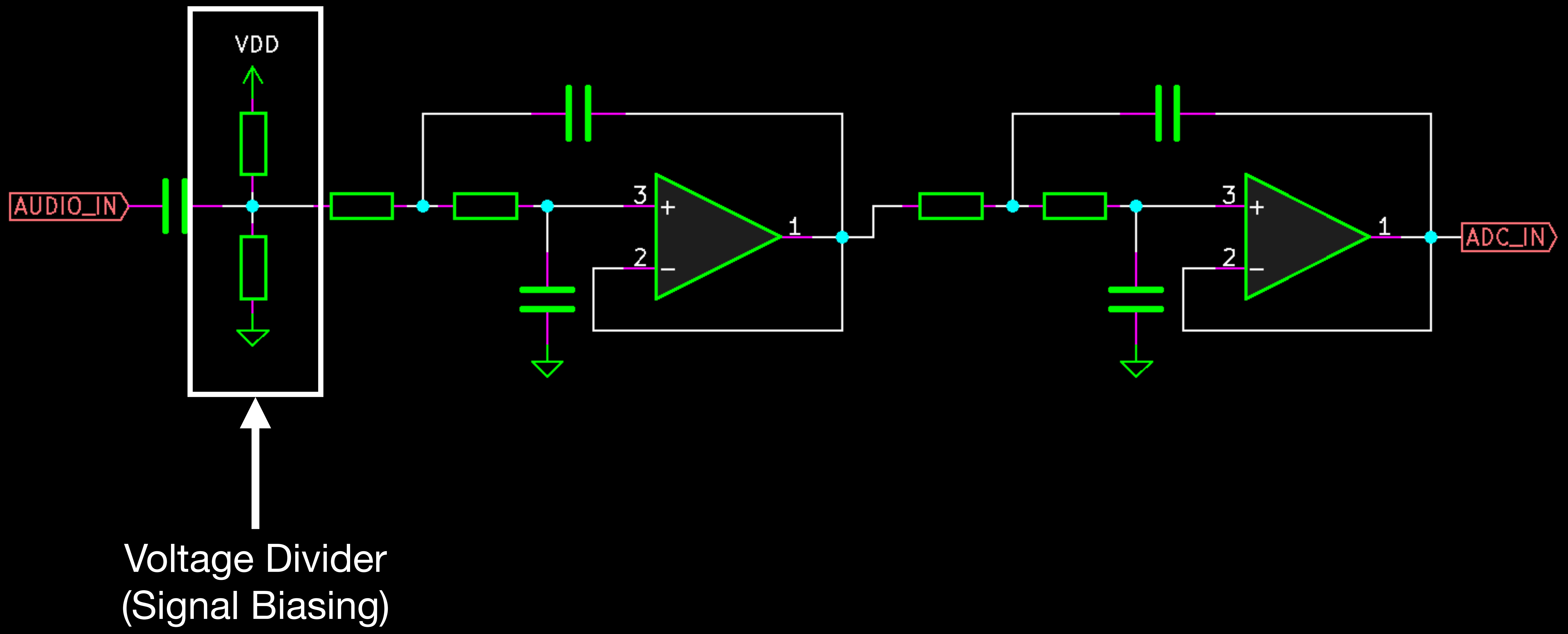
Example:
For a 12-bit ADC, the minimum stop band attenuation is 20log(2^12) = -72 dB

Microcontroller ADCs/DACs

Anti-alias Filter Design

- Many ADCs in MCUs are not dual-rail (+ve and -ve voltage swing) capable
- Therefore, a bias will need to be introduced to the signal to avoid signal clipping
- This may affect your DSP algorithms so remember to remove the DC component in-software!!

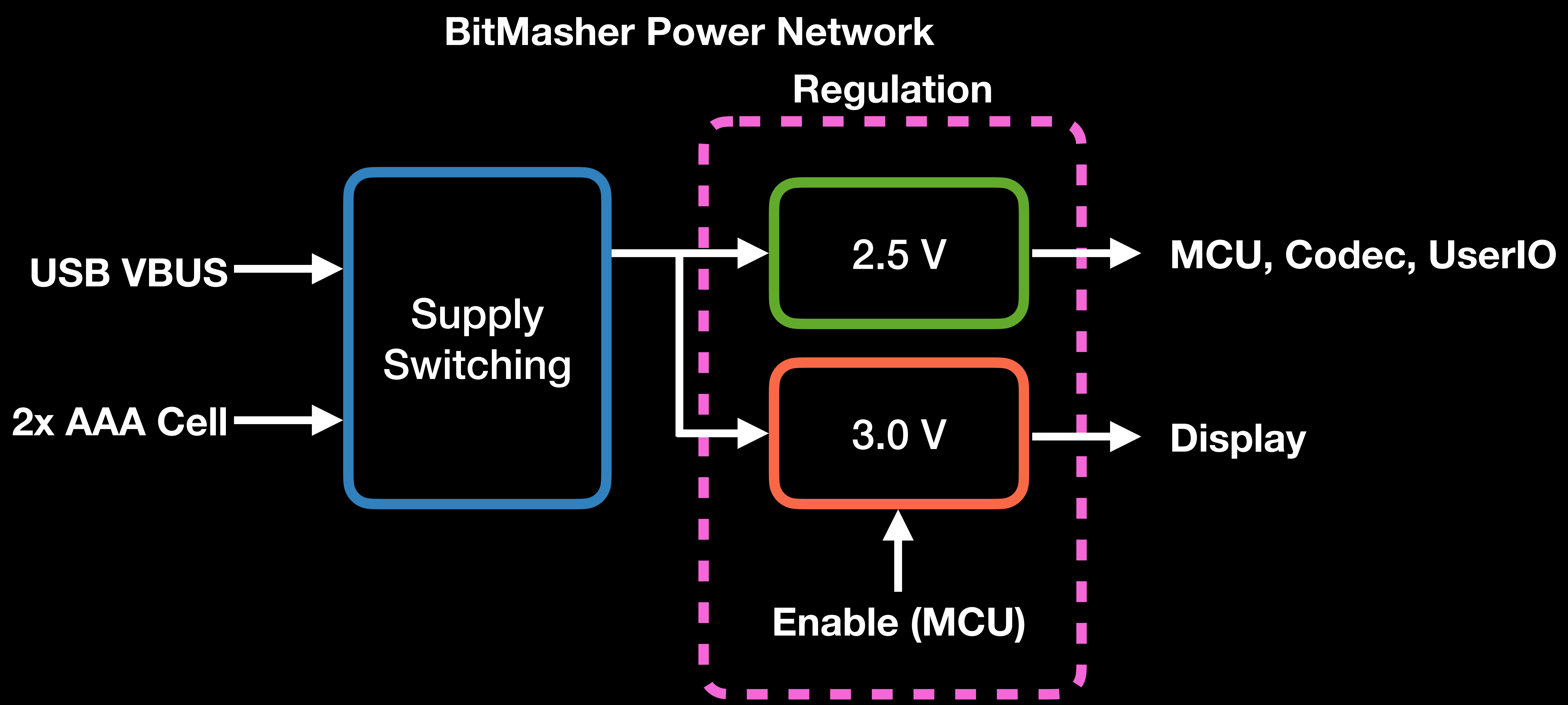
4th Order Sallen-Key Filter



System Power

Overview

- Power circuit designs vary depending on system complexity and needs
- Careful design considerations must be made with regards to supply, capacity, signal integrity, protection etc



System Power

Battery Chemistry

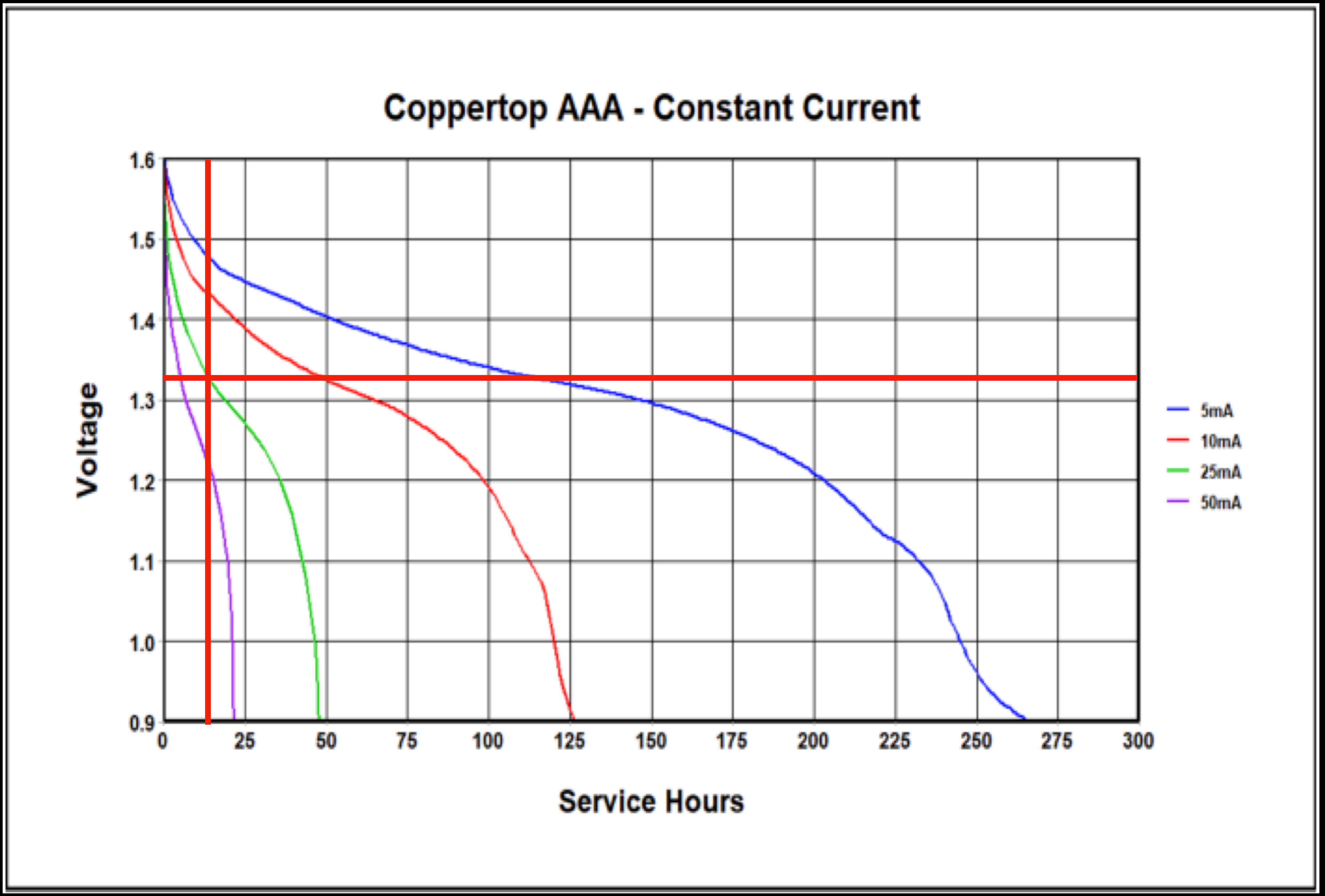
- Batteries are the most popular way to power portable electronics
- Different battery chemistries are suited for different applications

Chemistry	Pros	Cons
Alkaline	Widely available, reasonably good power density, good selection of capacities	Not rechargeable, may not be suitable for high discharge applications
LiPo	Rechargeable, high power density, small form factor, capable of high discharge rates	Dangerous if not handled properly, expensive

System Power

BitMasher Expected Lifetime

- Measured current draw = 20 mA
- Minimum VIN voltage for 2.5 V regulator = 2.675 V



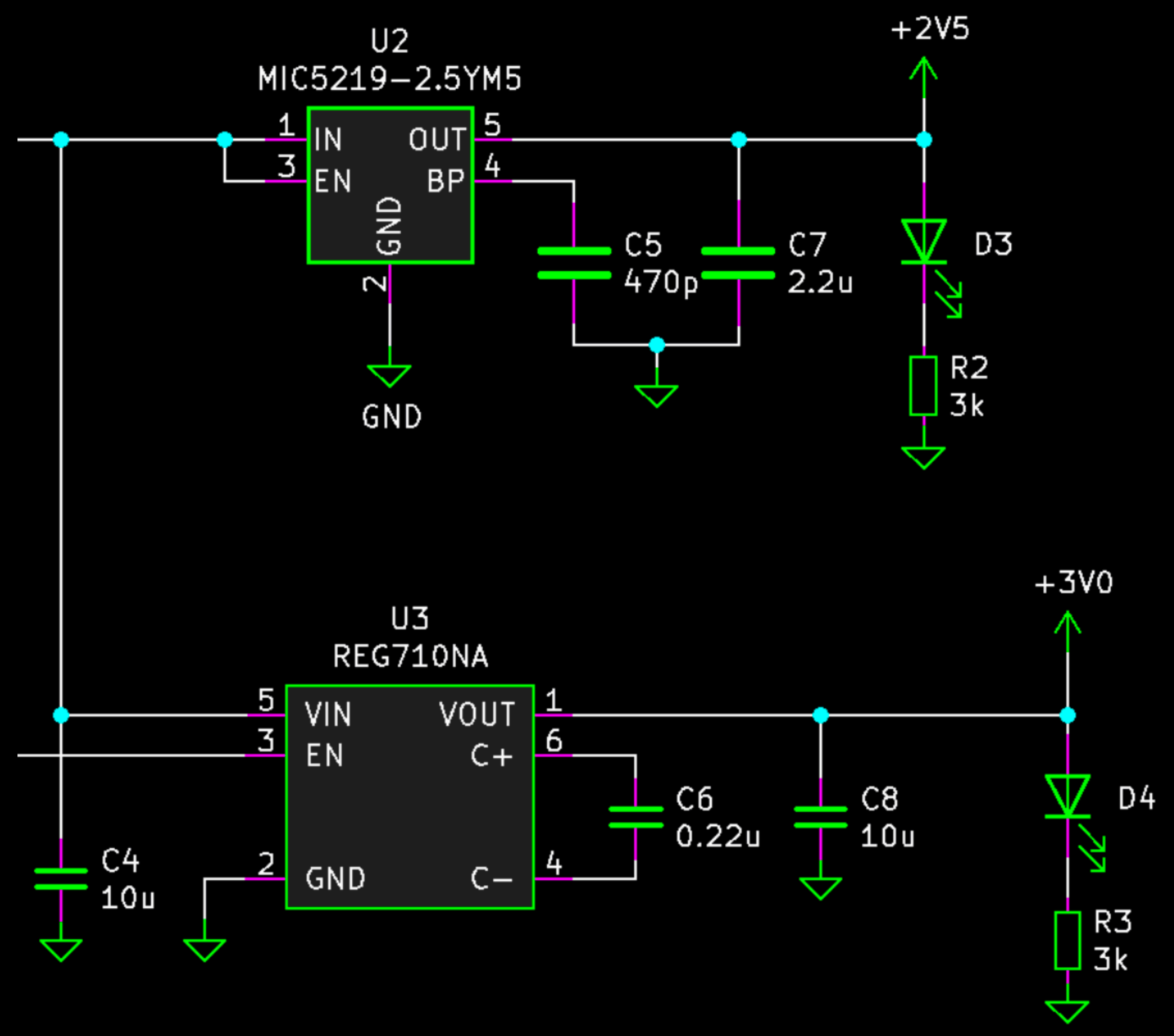
https://www.duracell.com/wp-content/uploads/2020/02/MN24US1119.pdf

If constantly drawing 20 mA,
Expect about 13 hours of operation
May benefit from using 1.8 V regulator instead
at the cost of audio input overhead!

System Power

Voltage Regulation

- Voltage regulators supply a stable voltage source to components
- Like many components, there is a wide selection!
- Some criteria include desired voltage, current output, footprint and type
- Two main types: Linear and Switching



2.5 V Linear Regulator

3.0 V Switching Regulator

System Power

Linear Regulators

- Very simple to understand
- Clean voltage output = Great for analog circuits!
- Vout < Vin only
- Be careful of dropout voltages as this may affect your battery choices!

MIC5219 Dropout Voltage Characteristics

$V_{IN} - V_{OUT}$	Dropout Voltage(6)	I_{OUT} = 100µA		10	60 **80**	mV
		I_{OUT} = 50mA		115	175 **250**	mV
		I_{OUT} = 150mA		175	300 **400**	mV
		I_{OUT} = 500mA		350	500 **600**	mV

BitMasher draws about 20 mA @ 2.5 V so we will use IOUT = 50 mA value.

Therefore, VIN >= 2.5 + 0.175 = 2.675 V

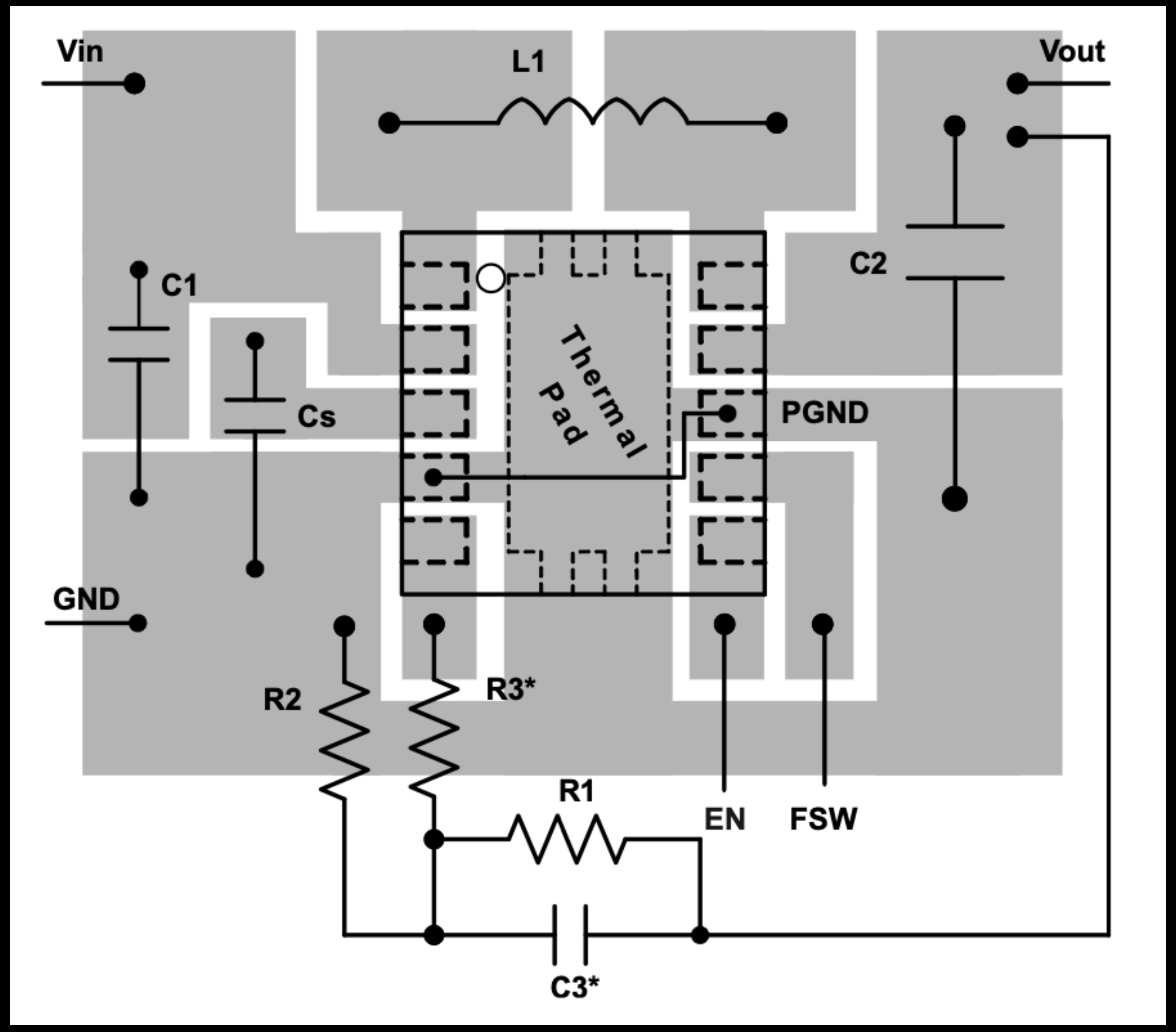
BitMasher 2.5 V Output

System Power

Switching Regulators

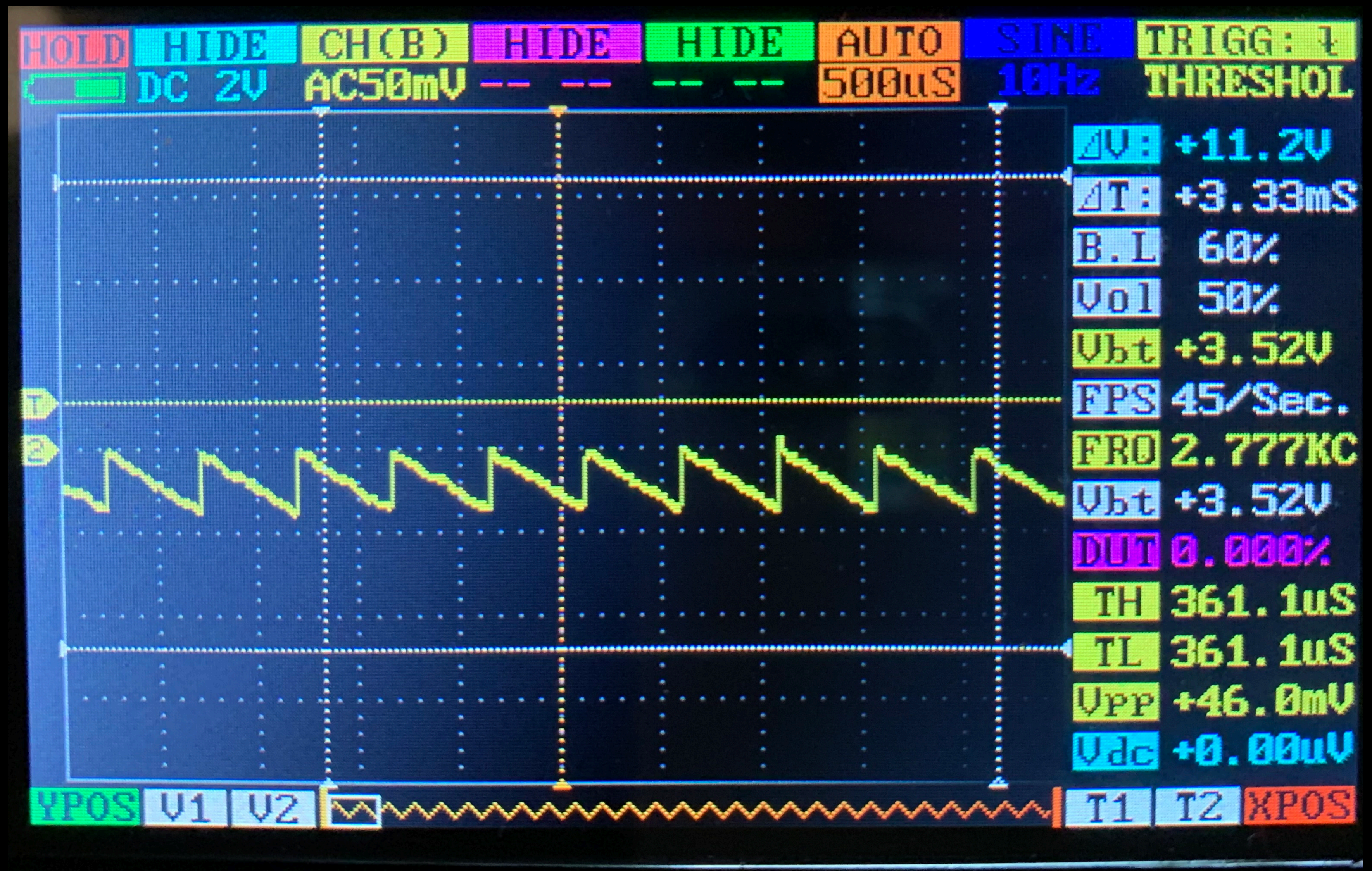
- Very efficient!
- Buck AND boost (Vout > Vin) options
- Can be noisy = Not great for analog circuits
- Improper PCB layout and component selection can cause stability and EMC issues

Example PCB Layout for Switching Regulator



From TI TPS6108 Datasheet

BitMasher 3.0 V Output Ripple



***Be careful when selecting capacitors and inductors! Poor selection can lead to worse noise performance and/or stability issues!**
Follow manufacturer recommendations

System Power

BitMasher Power Needs

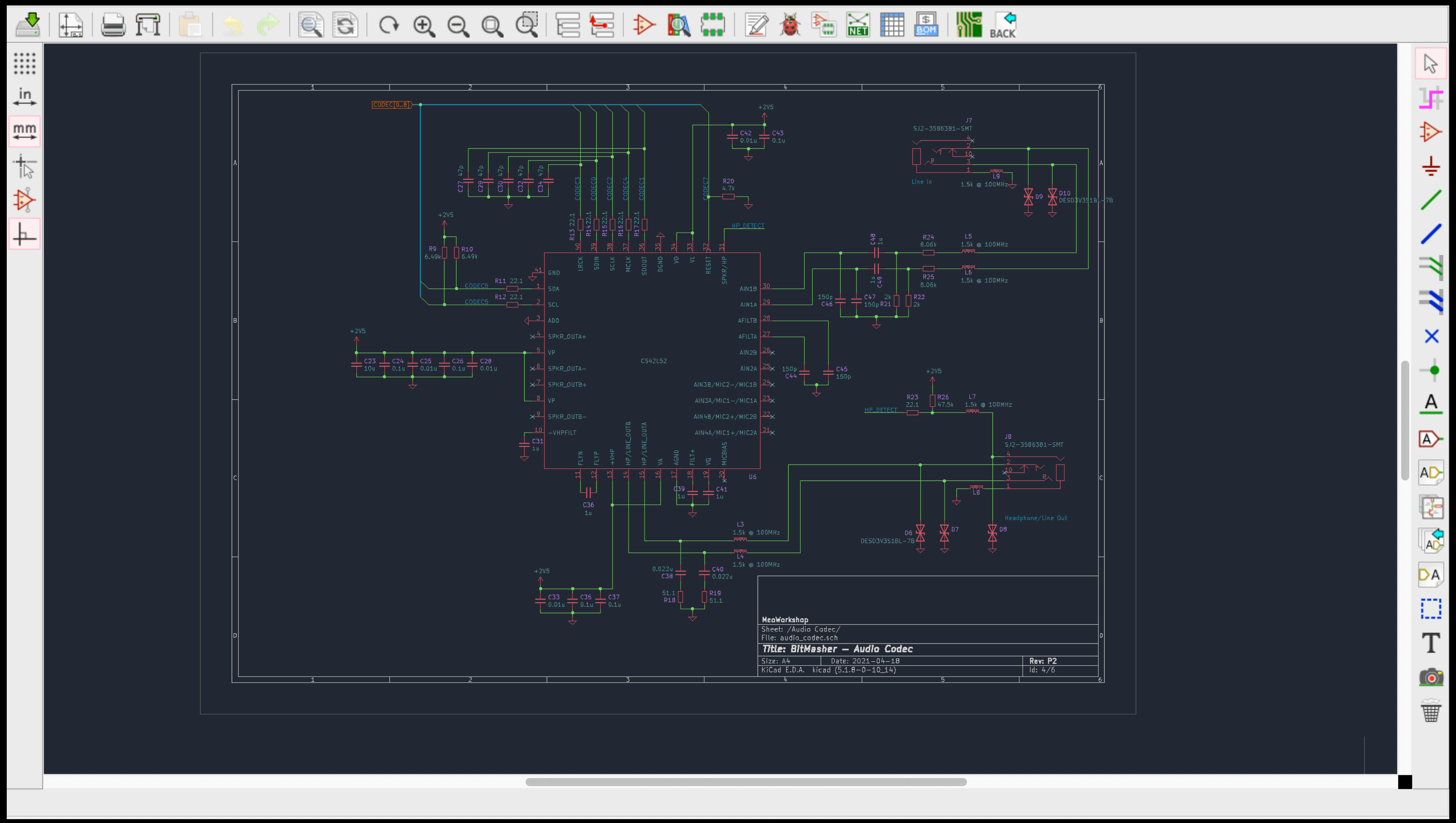
Component	Operating Voltage + Current Consumption	Requirements
MCU	1.8 - 3.8 V Operating voltage, 126μA / MHz current draw (@40 MHz, 5.04 mA, no peripherals)	1x 1.8 V regulator with at least 16 mA current output capability (Increase to 2.5 V for additional audio input overhead)
Codec	1.8 - 2.63 V Operating voltage, ~11 mA current draw (stereo + headphone output)	
Display	2.7 - 3.3 V Operating voltage, ~50 uA current draw	1x 3.0 V boost and buck regulator with at least 50 uA current output capability

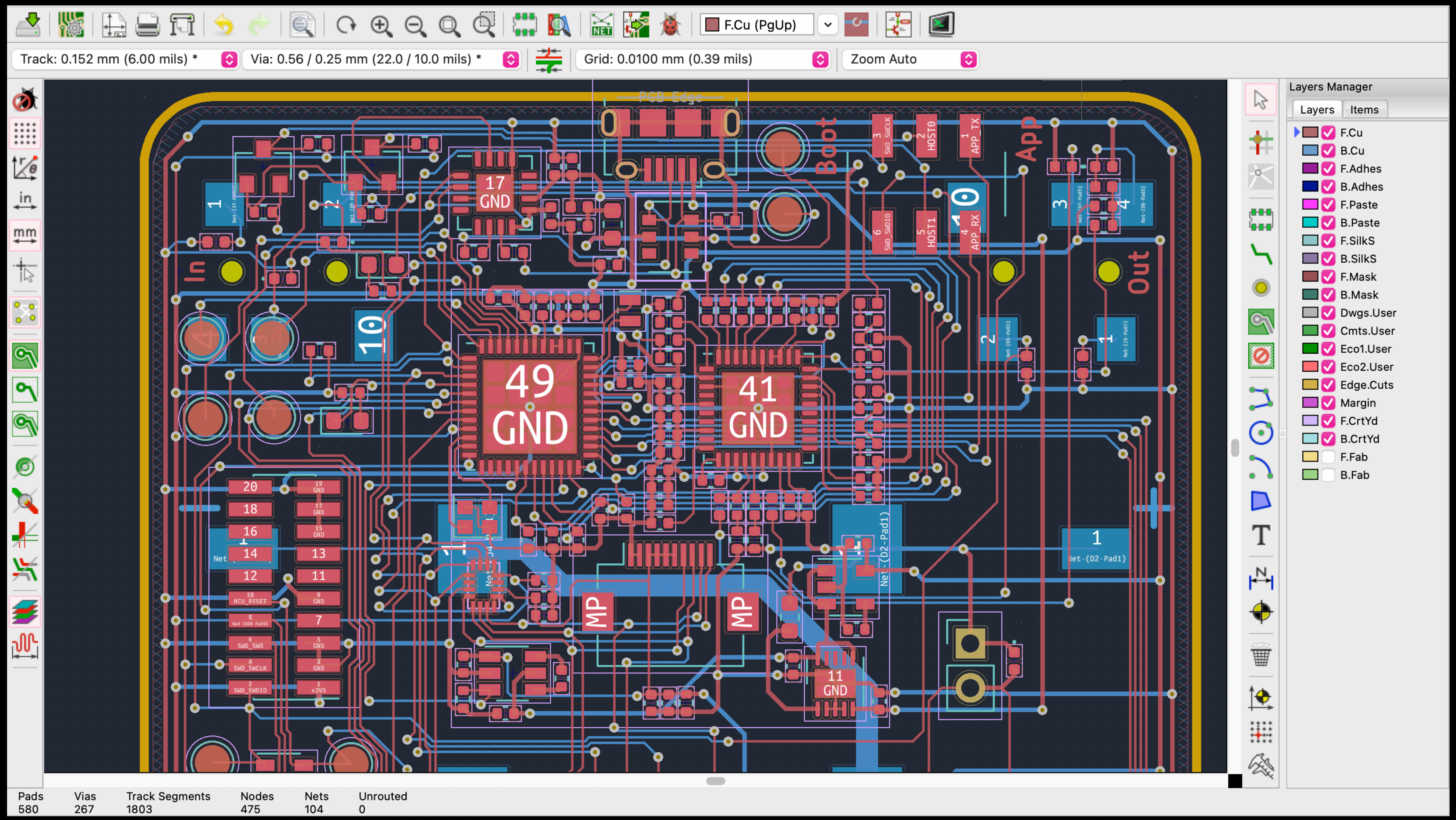
Schematic Capture + Layout

EDA Tools

- Electronic Design Automation Tools
- Encompasses schematic capture, PCB layout, RF simulations, etc
- Can range from free to $$$$$$$$$$$
- Popular hobbyist EDA tools include KiCad, EAGLE, Altium CircuitMaker/CircuitStudio

KiCad (BitMasher Schematic and PCB)



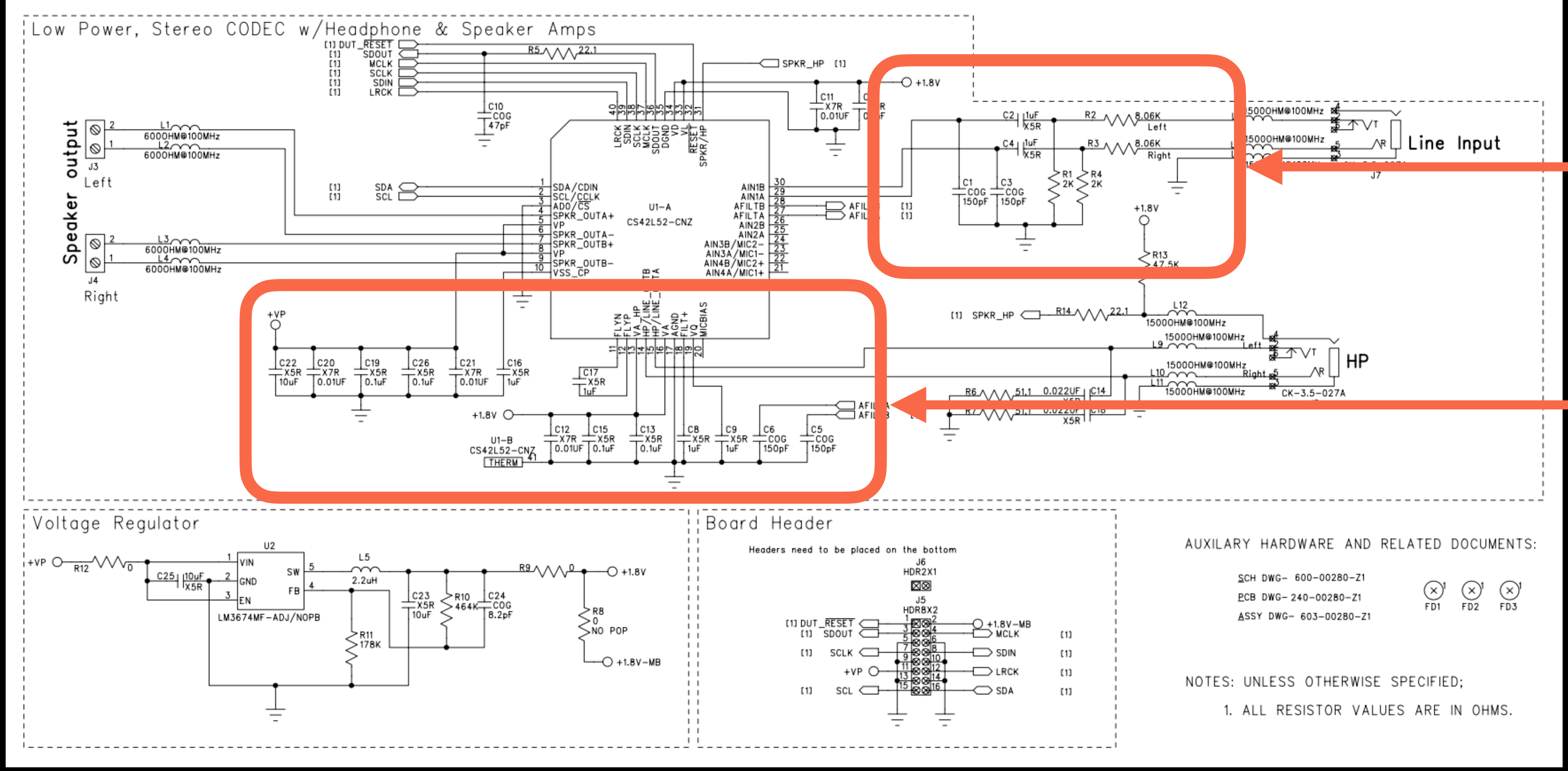


Schematic Capture + Layout

Reference Designs

- IC manufacturers often provide reference designs found in the data sheet or application note
- Example circuits usually specify recommended passives and wiring

CS42L52 Reference Design



Input filtering network

Recommended bypass capacitor network

Cirrus Logic CS42L52 Reference Design
CRD42L52

Schematic Capture + Layout

RTFM

- READ YOUR DATA SHEETS!
- Make sure that a given peripheral DOES offer the feature you want!

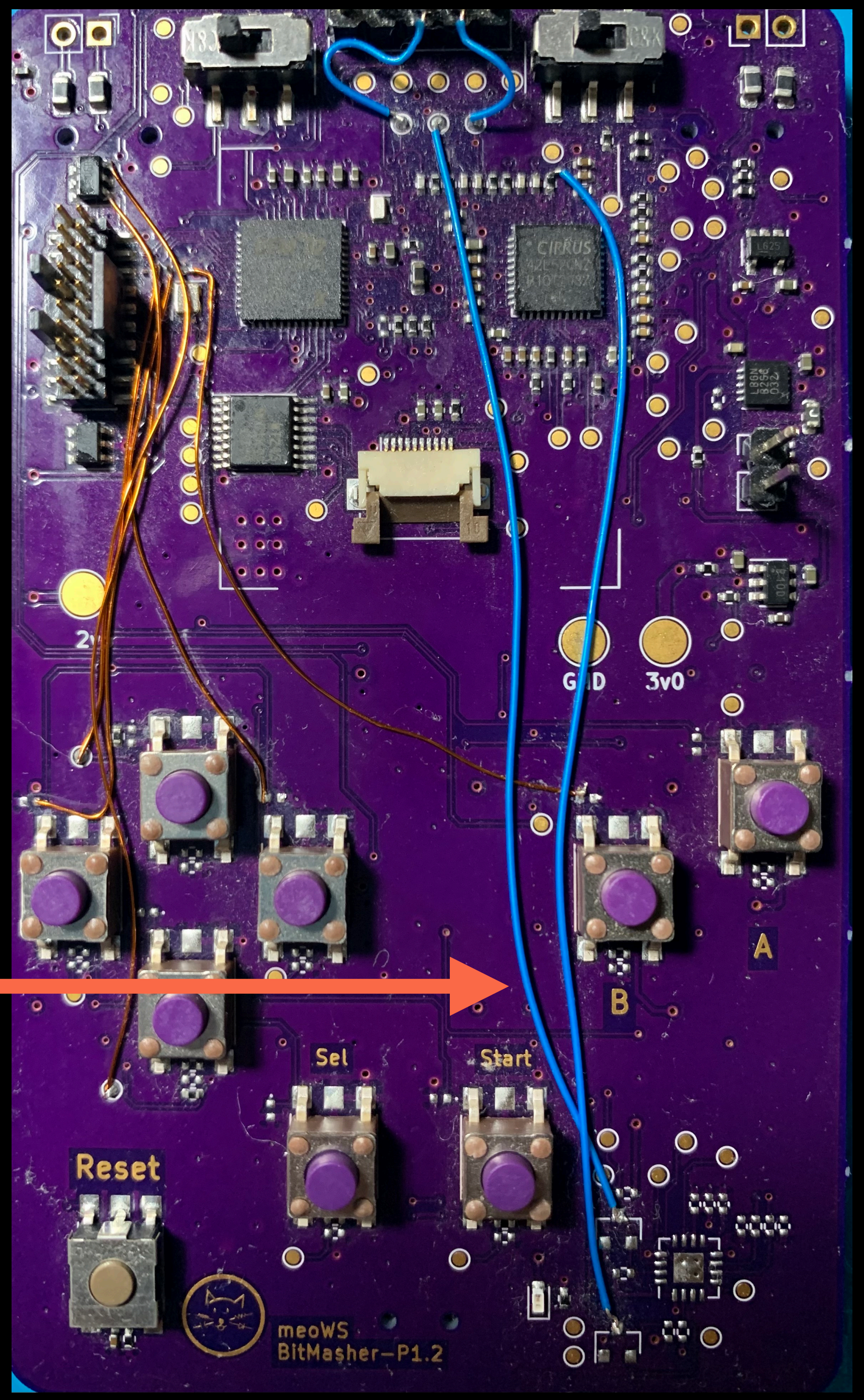
Example

- EFM32PG12 MCU offers I2S via USART (serial) modules
- Multiple USART modules available (USART0, 1, 2…)
- Not all USART modules are I2S capable however…

Module	Configuration	Pin Connections
USART0	IrDA SmartCard	US0_TX, US0_RX, US0_CLK, US0_CS
USART1	I^2S SmartCard	US1_TX, US1_RX, US1_CLK, US1_CS
USART2	IrDA SmartCard	US2_TX, US2_RX, US2_CLK, US2_CS
USART3	I^2S SmartCard	US3_TX, US3_RX, US3_CLK, US3_CS

Guess who put I2S on UART0

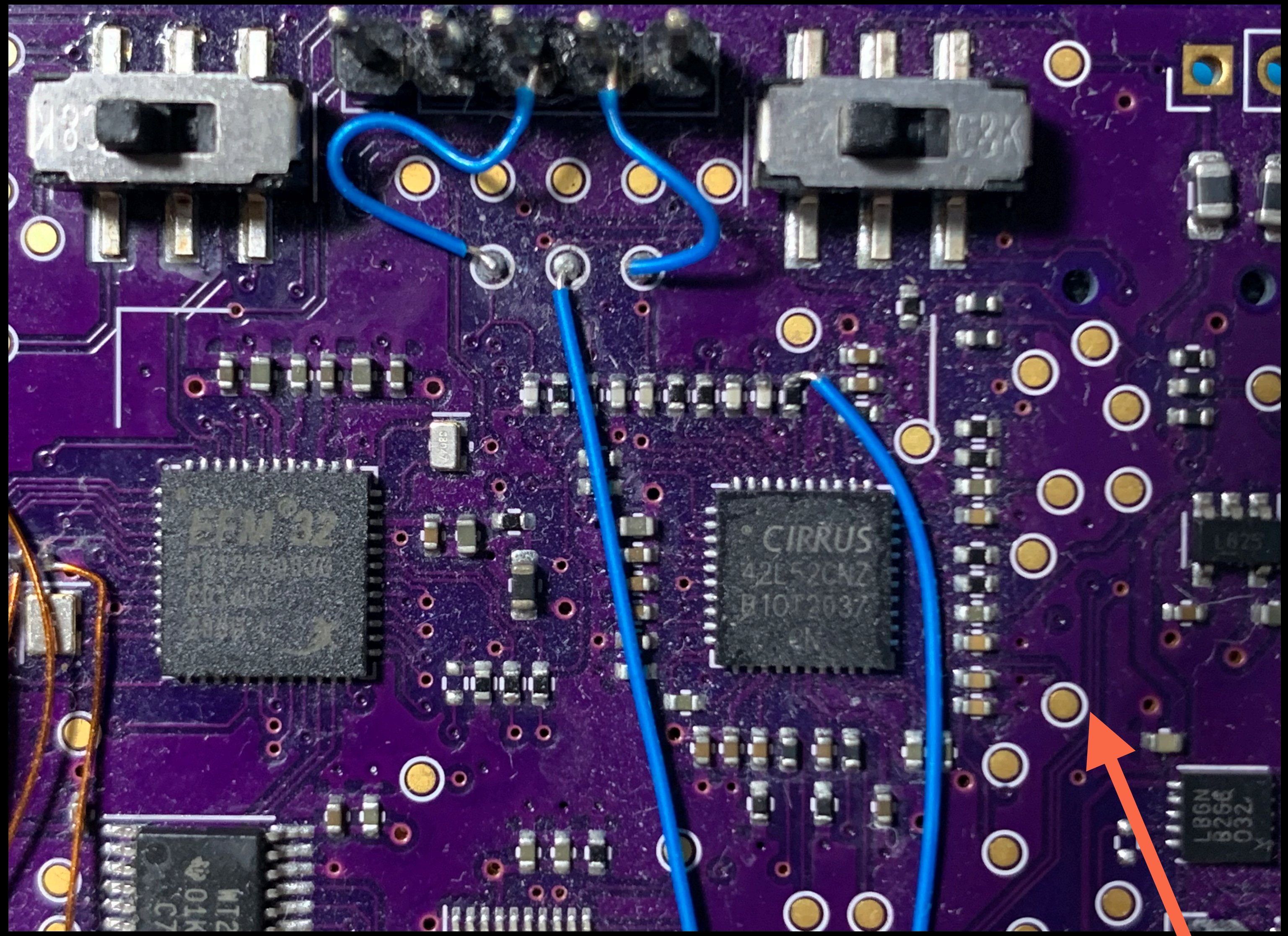
Oops!
(Rework wires to move
Codec connetions to USART1)



Schematic Capture + Layout

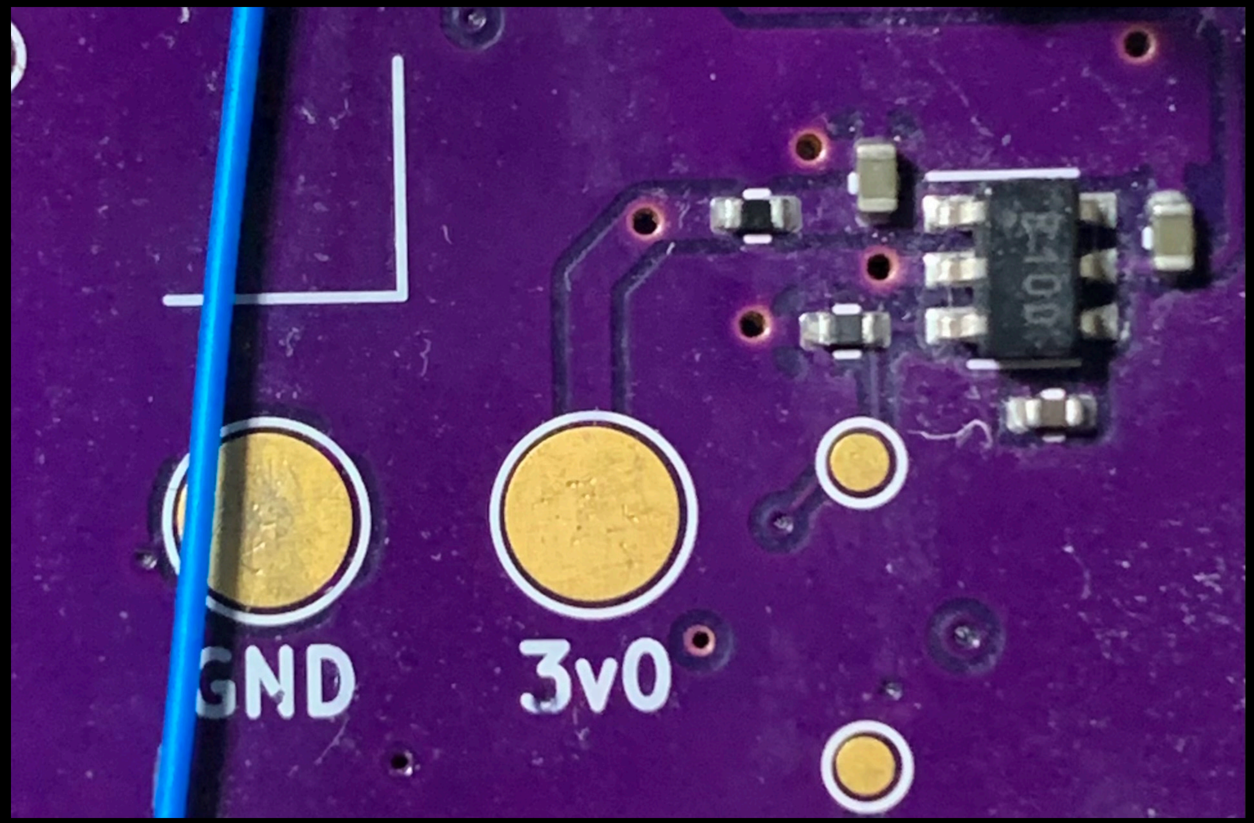
Test Points

- Makes debugging, verification and reworking much easier!
- Allows for circuit and functional testing if manufacturing your device
- Break out unused MCU pins to test points or connector
- You might end up needing them!

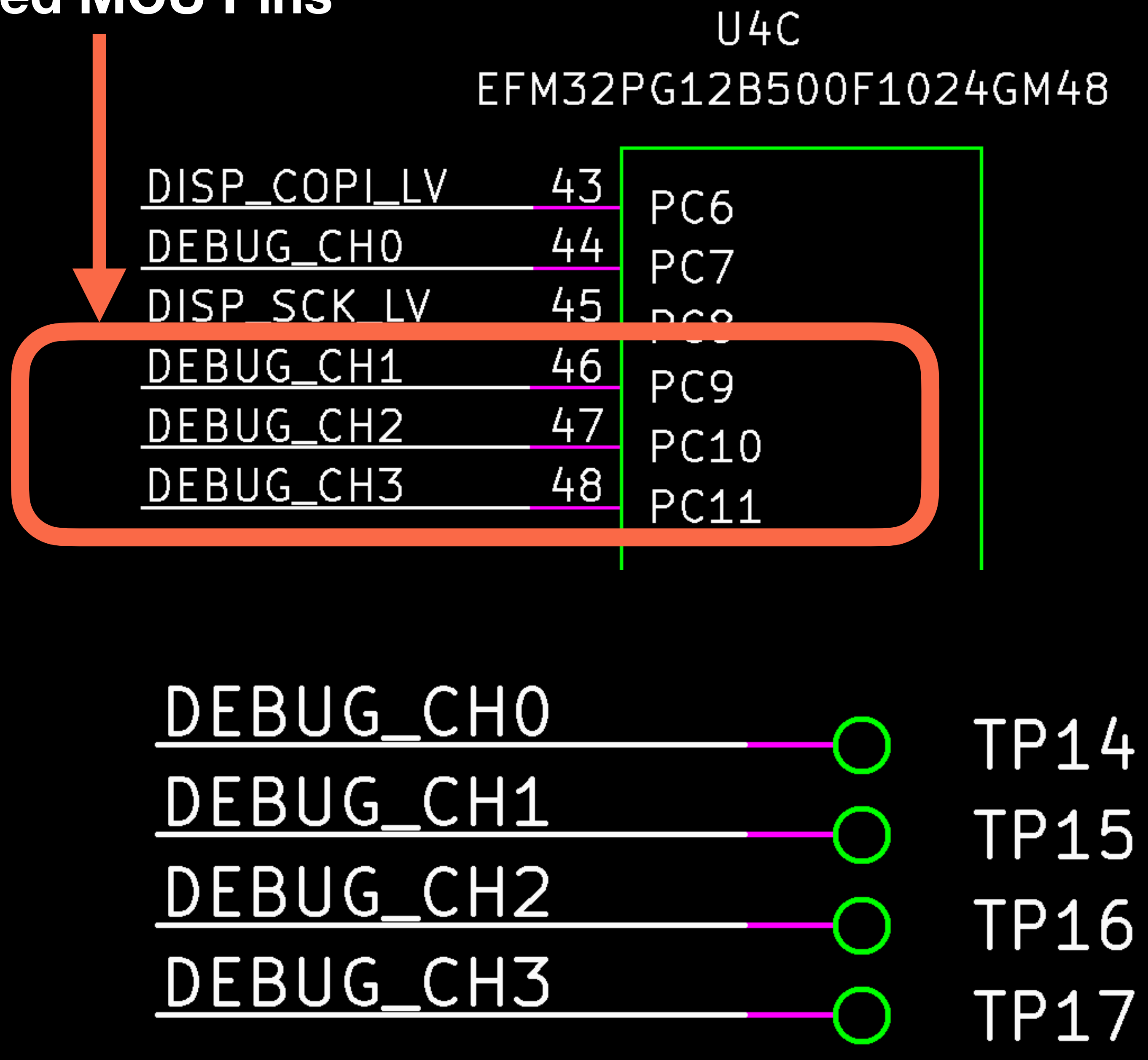


Test Points

Voltage Test Points



Unused MCU Pins



Schematic Capture + Layout

PCB Fabrication

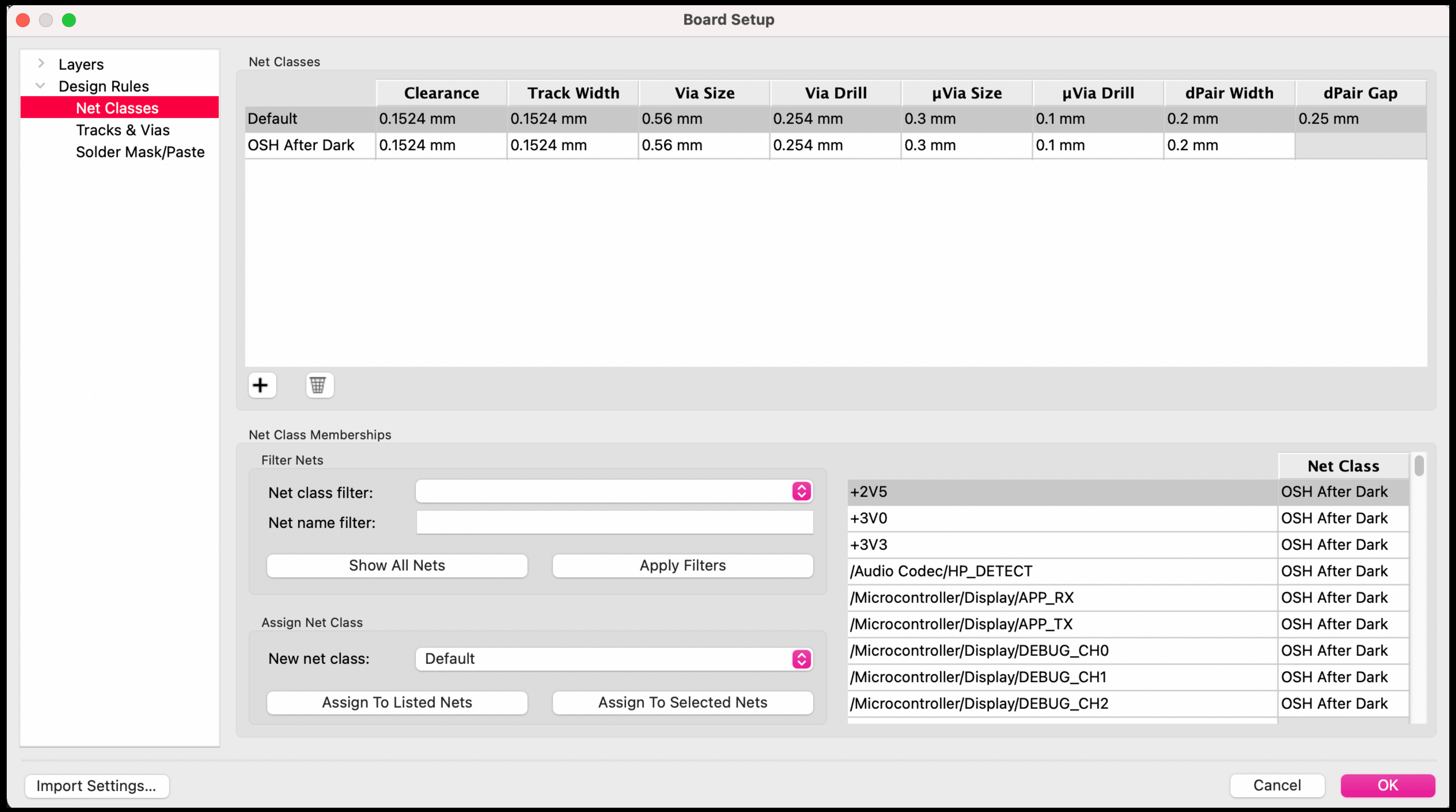
- PCB manufacturers will provide a list of manufacturing tolerances
- These tolerances should be added to your software's Design Rules Checker (DRC)
- Add extra margin to manufacturer's minimum trace widths, spacing and drill sizes!

Sample Tolerances

Spec	Value
Copper Layers	2
Copper Weight	1oz
Trace Spacing	6mil (0.1524mm)
Trace Width	6mil (0.1524mm)
Annular Ring	5mil (0.127mm)
Board Edge Keepout	15mil (0.381) from nominal board edge
Via Plating Thickness	1mil (0.0254mm)

https://docs.oshpark.com/services/two-layer/

KiCad Design Rules Check Settings

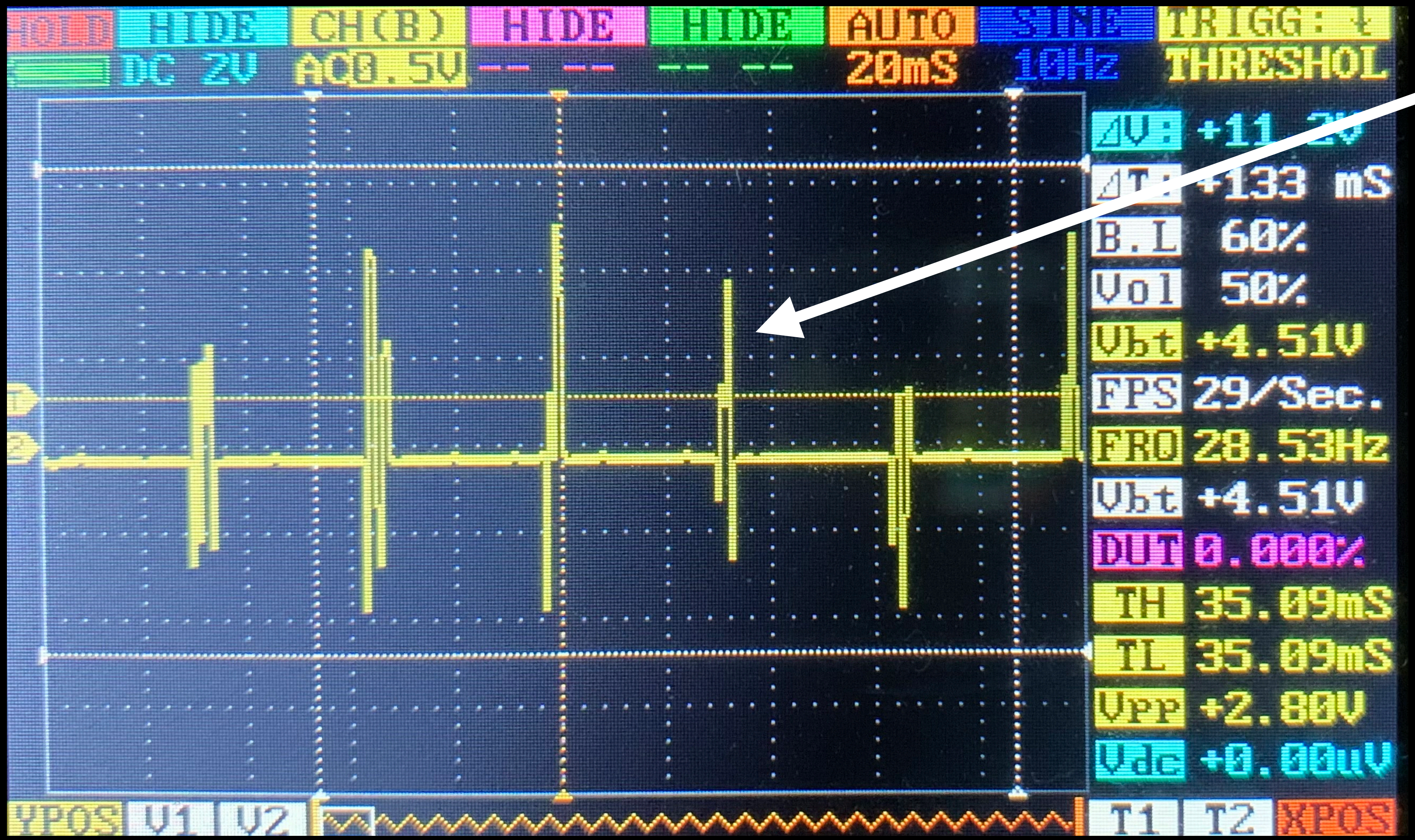


Schematic Capture + Layout

Mixed Signal Layout Considerations

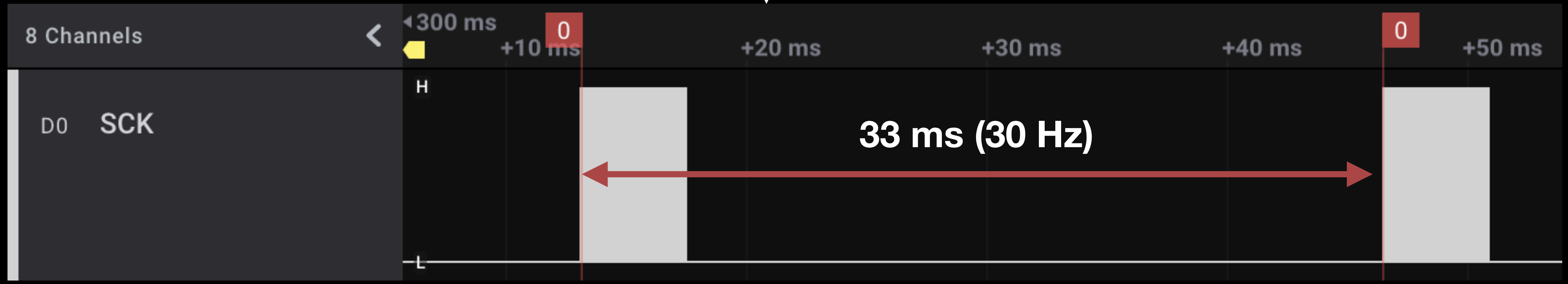
- PCBs with a mix of analog and digital circuits require some extra attention
- Component and trace placement becomes very important
- Grounding becomes *especially* important
- Improper layout may cause switching noise to couple into analog lines!

BitMasher Audio Output Capture



Connecting an oscilloscope probe to the audio output shows spikes in the audio!

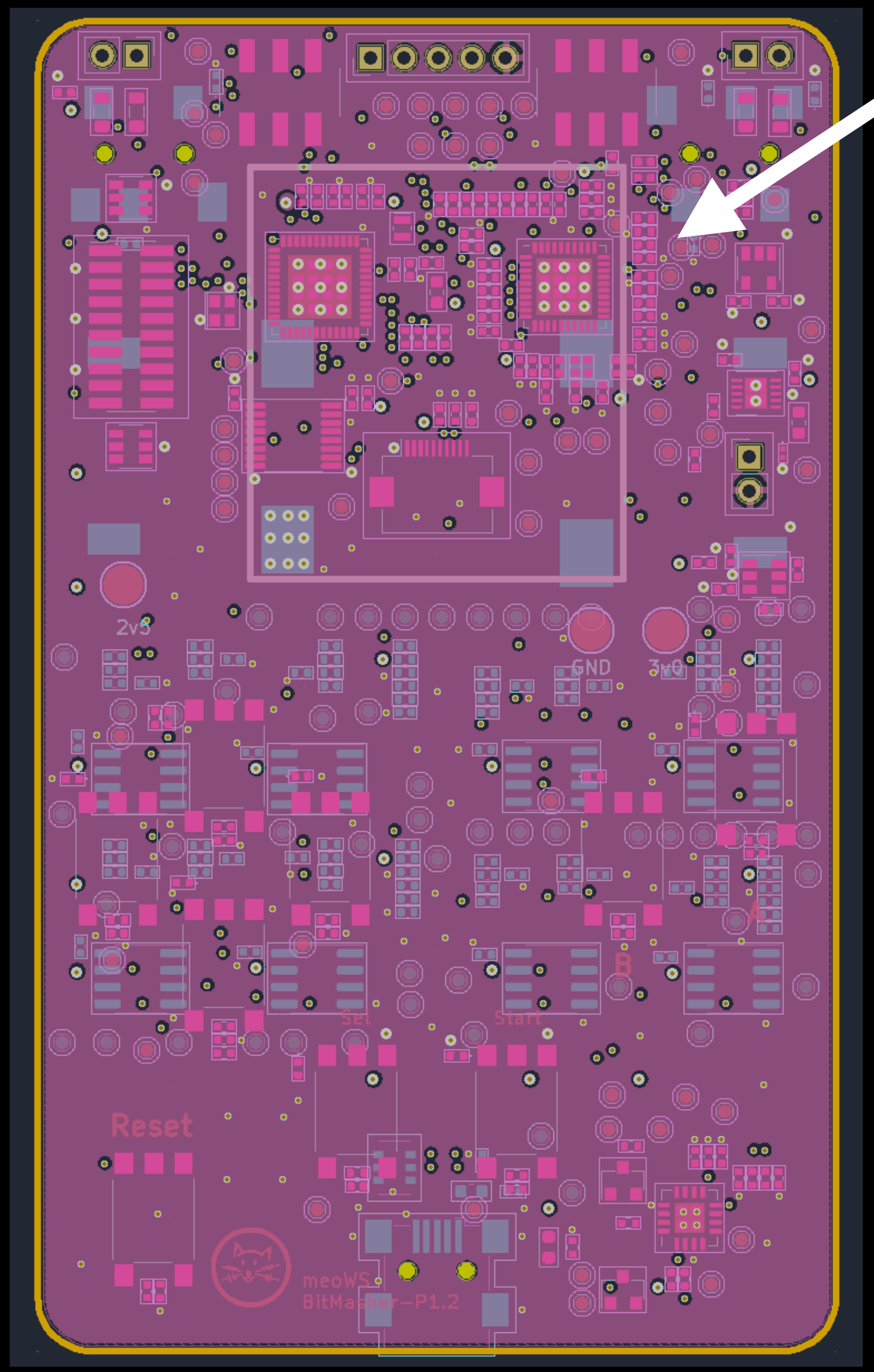
Root cause is from the display SPI lines
Note that the time between audio spikes is the same as the time between SCK groups



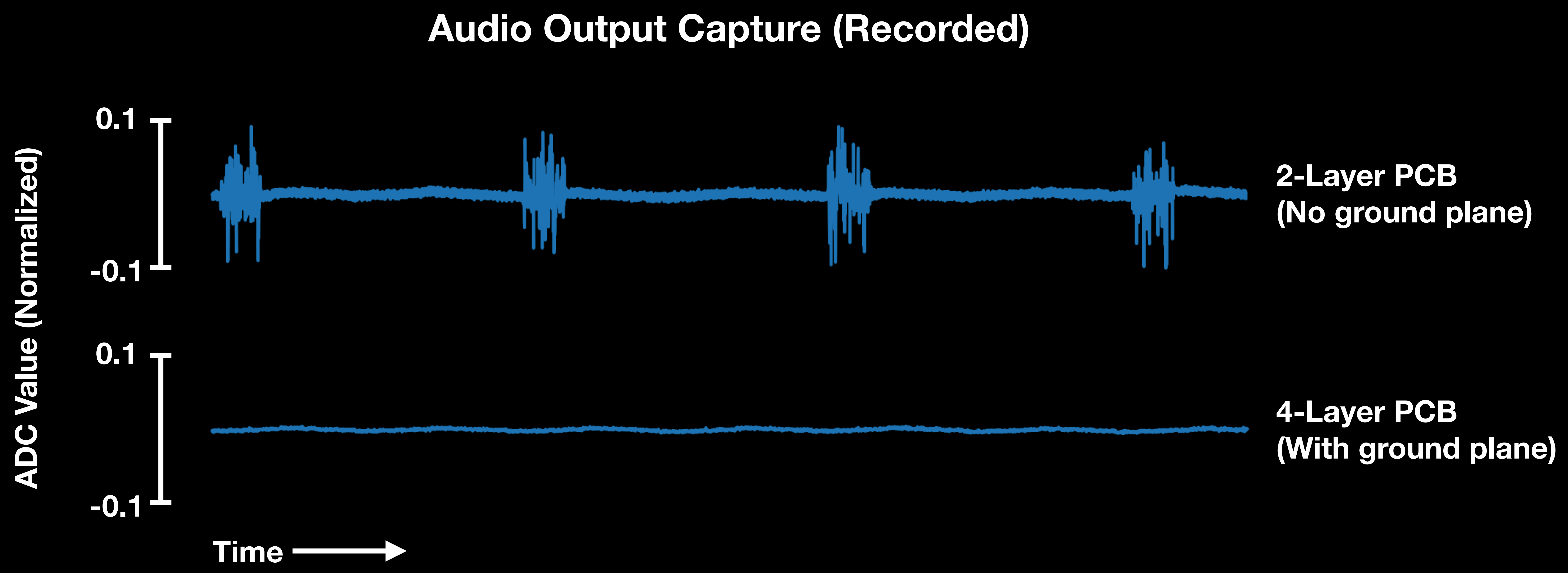
Schematic Capture + Layout

Ground Planes

- PCBs can have multiple copper layers to place traces
- Dedicated ground planes provide shorter current return paths for high speed signals
- Benefits include reduced noise and EMC emissions



Dedicated internal ground plane (Purple area)

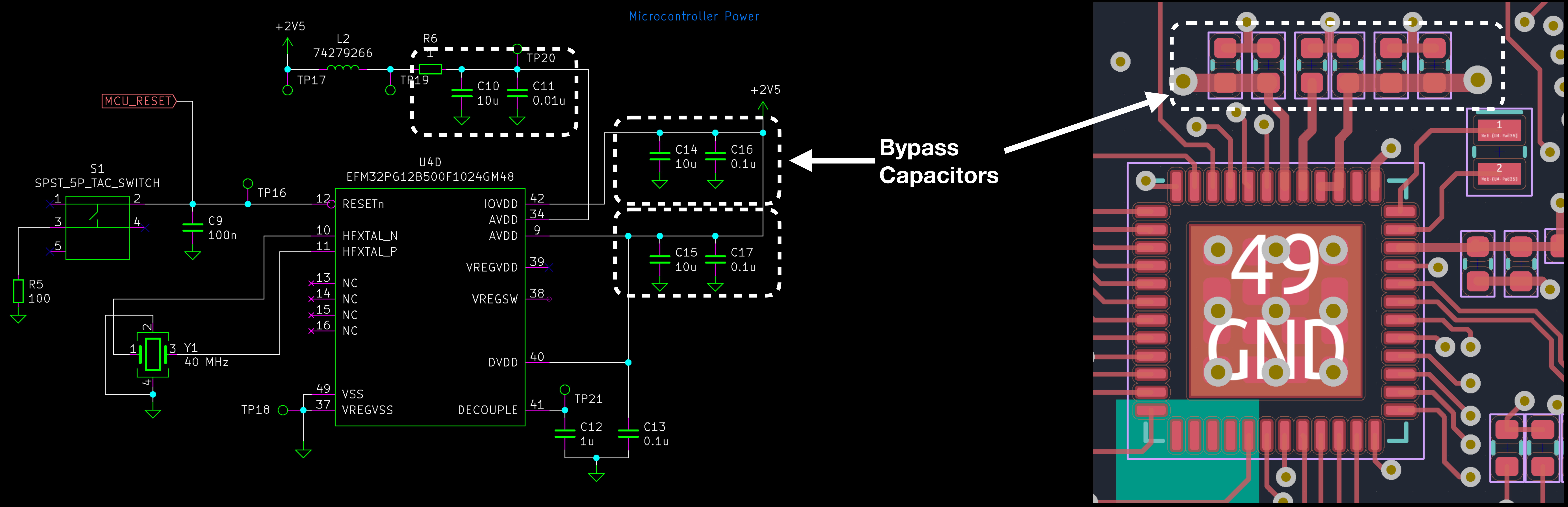


*Grounding is a complicated topic and techniques vary on a case-by-case basis!

Schematic Capture + Layout

Bypass Capacitor Placement

- Bypass capacitors should be placed *as close as possible* to the IC
- Longer distances → longer traces → greater inductance → greater EMC emission risk
- Longer distances also impede the capacitor from supplying power during transients



PCB Assembly

Soldering

*Soldering presents health and safety hazards!
Take appropriate precautions before soldering!!

- Two main soldering methods for prototyping:
  - Hand soldering
  - Reflow soldering
- Reflow soldering is predominantly used in manufacturing
- Therefore, if planning to manufacture your product, it is recommended to use SMT components!

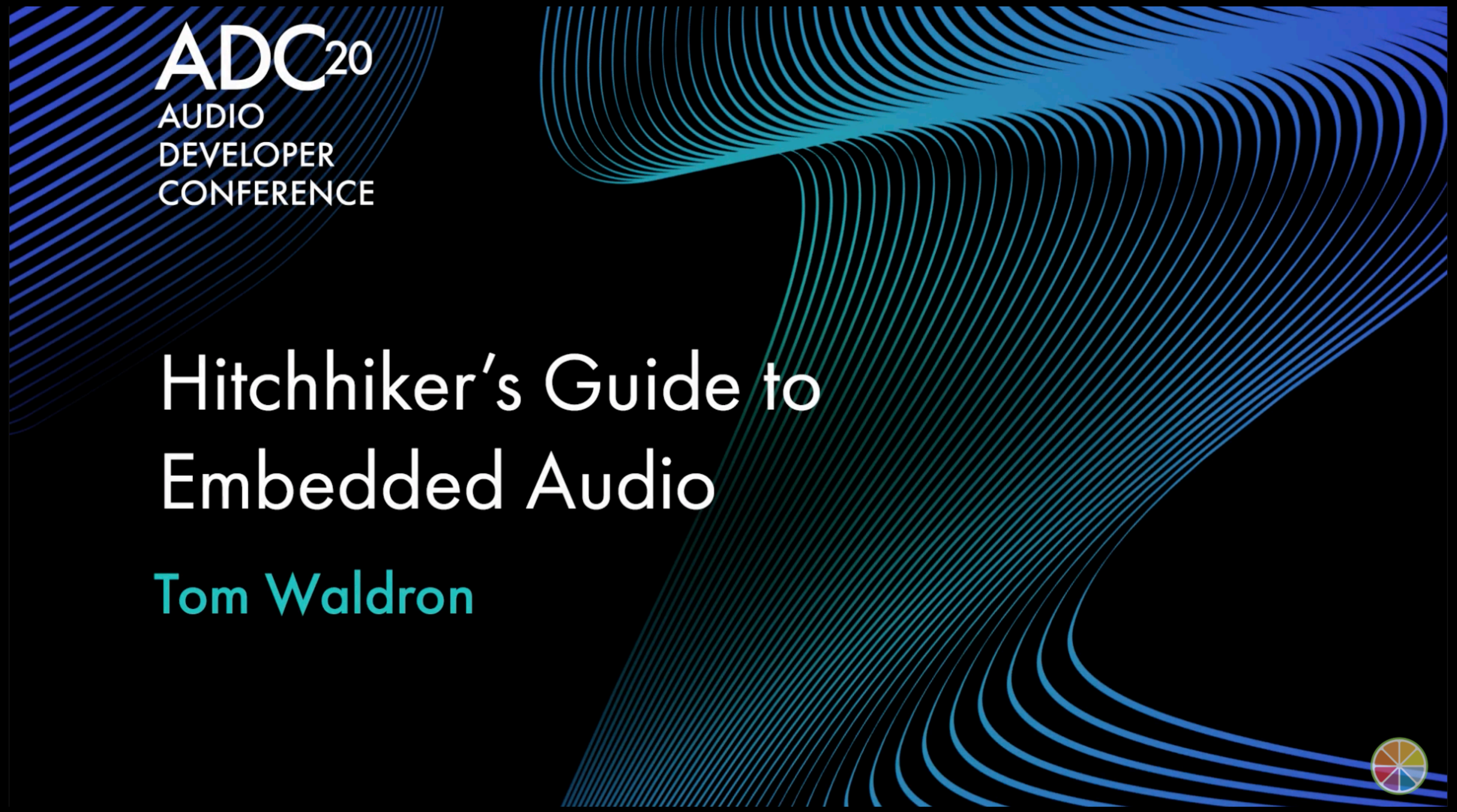
Preparing PCB for Reflow

The Firmware

The Firmware

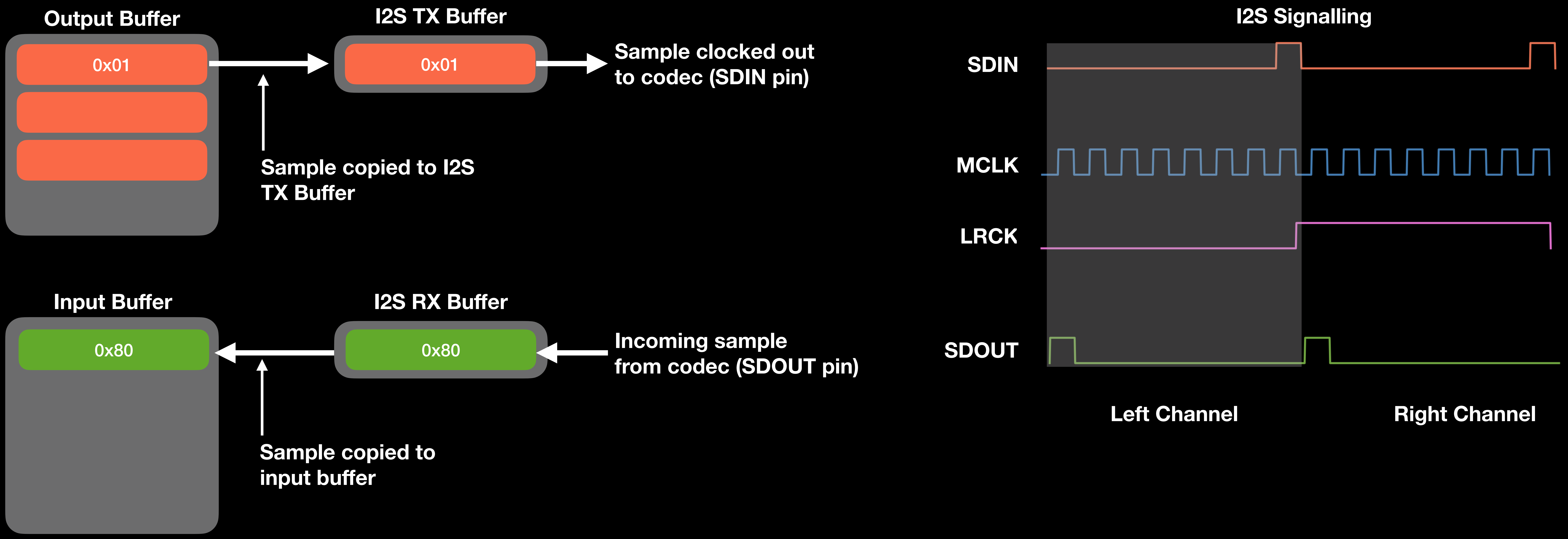
Past ADC Talks

- For a more in-depth introduction to embedded programming, the below ADC20 talks are highly recommended



Audio IO

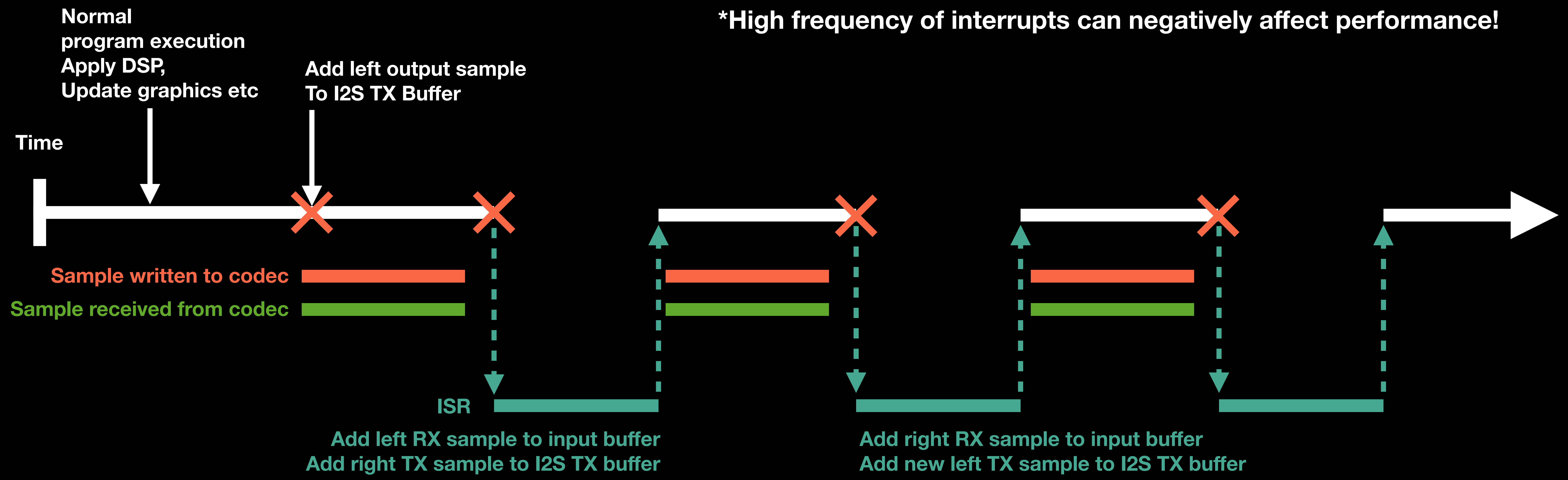
Audio Data Flow (I2S)



Audio IO

Sequence of Events

*High frequency of interrupts can negatively affect performance!

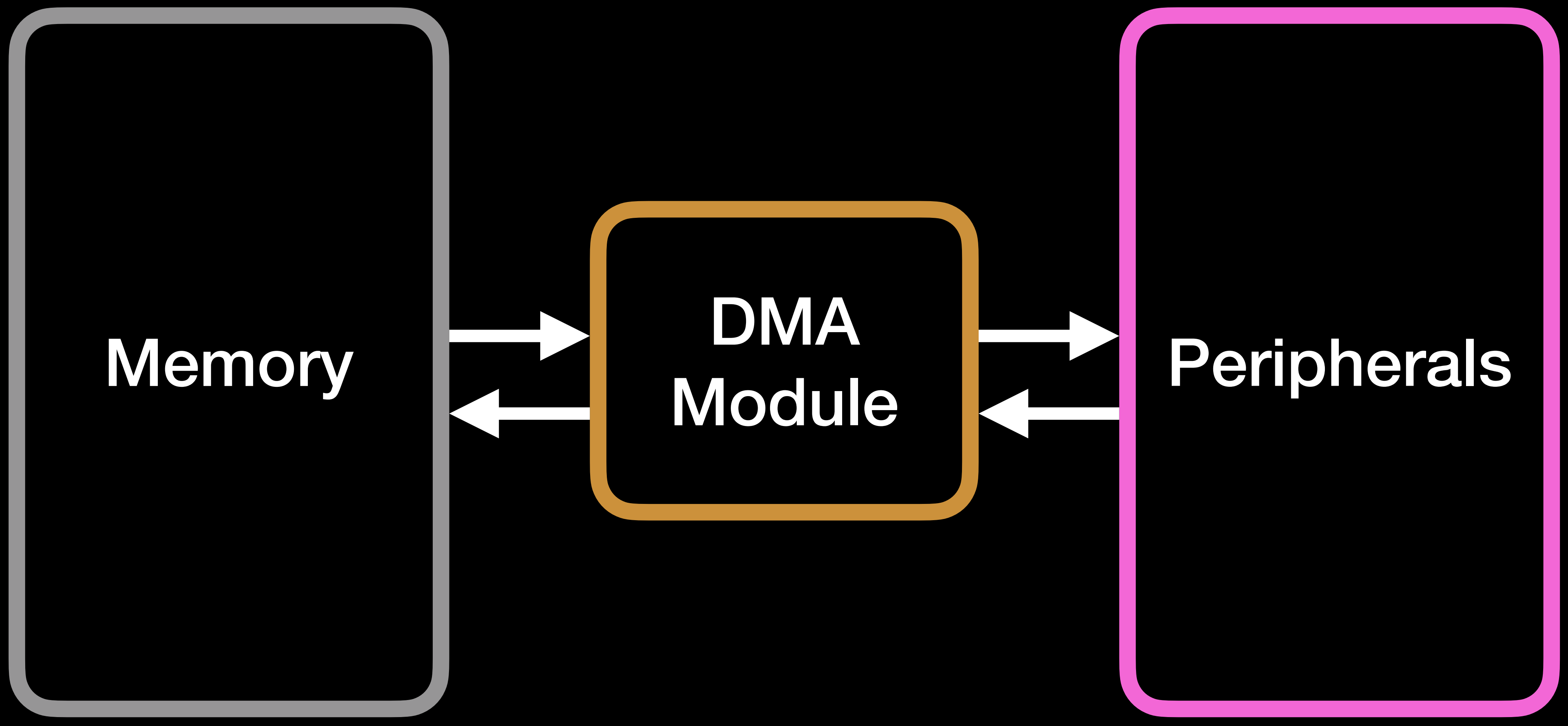


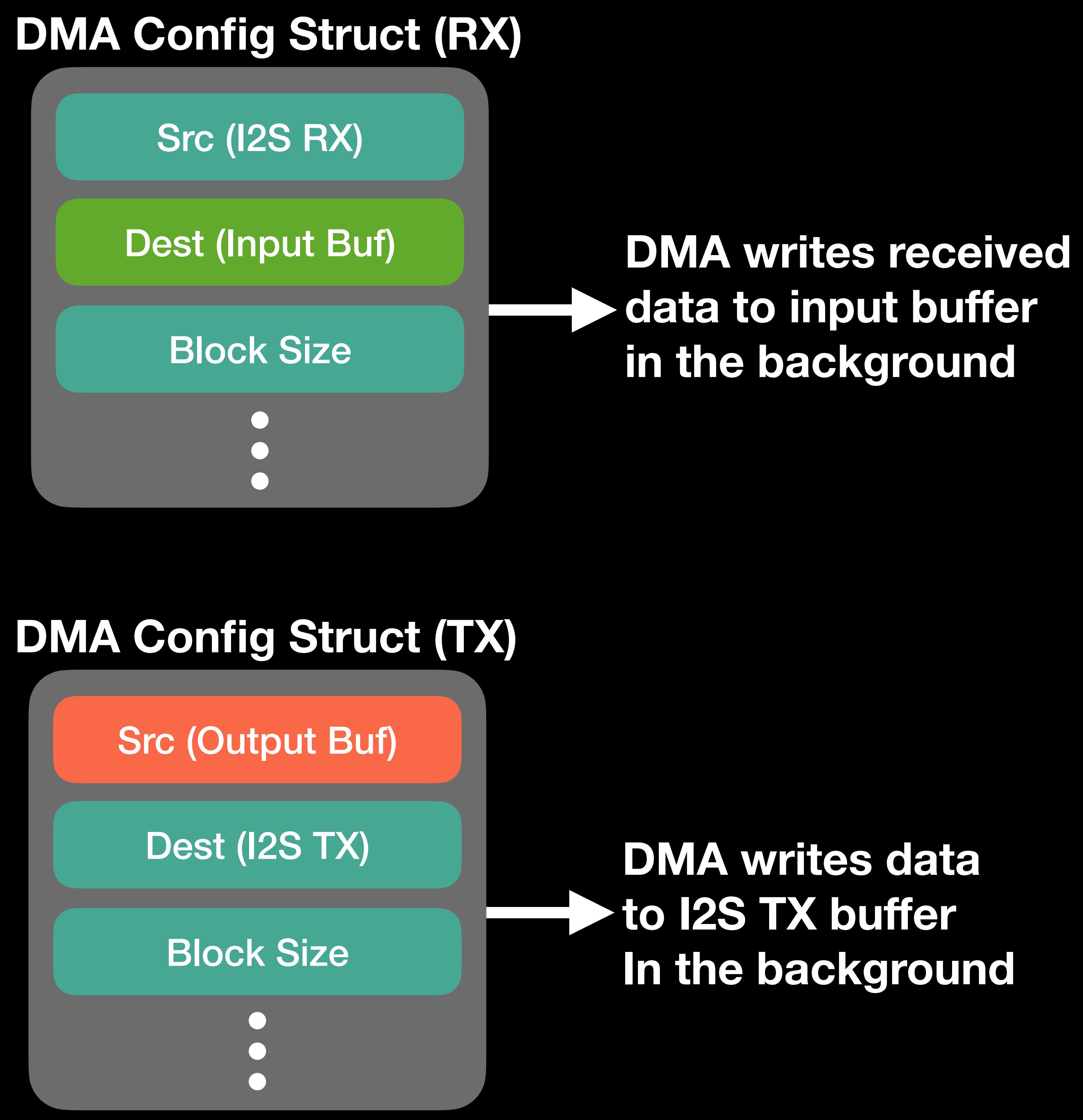
Audio IO

Direct Memory Access

- High frequency of interrupts can affect performance!
- Direct Memory Access (DMA) module offloads movement of data from CPU
- Asynchronous transfer and reception of audio data leaves more time for processing!

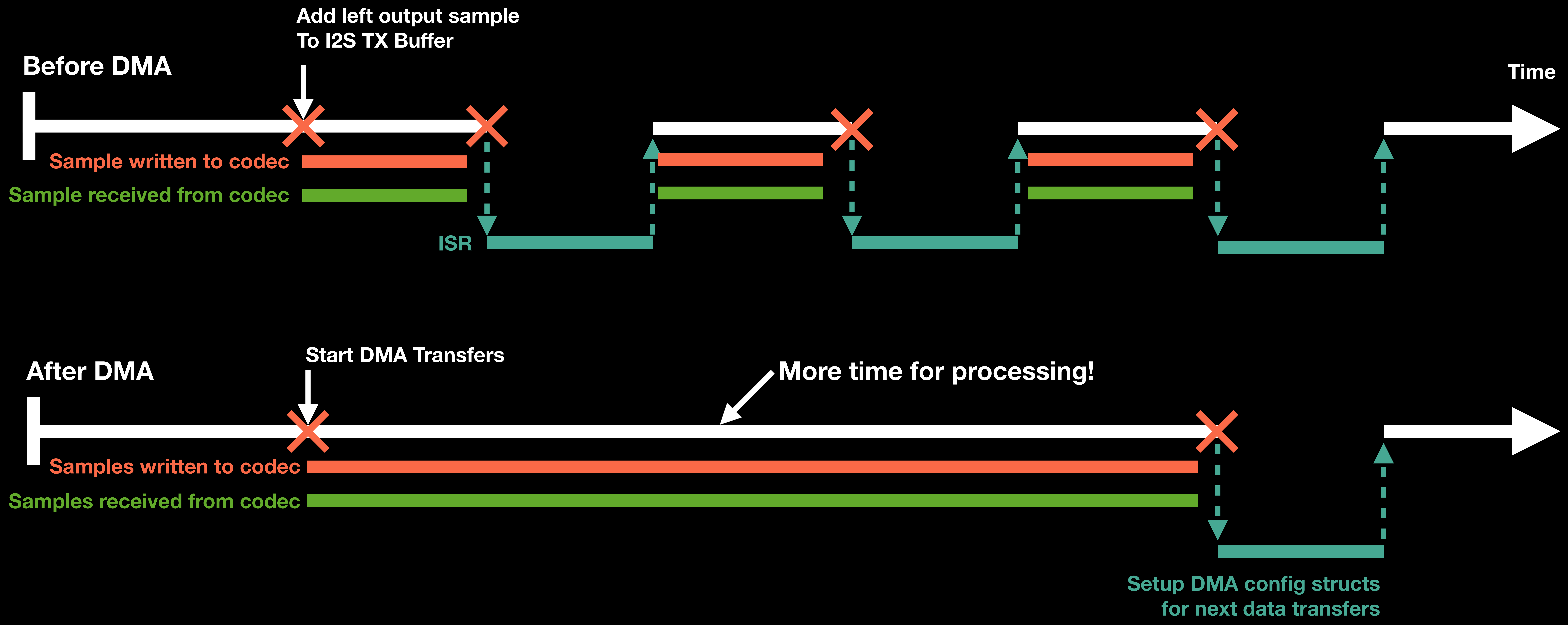
Data Flow from Memory to Peripherals





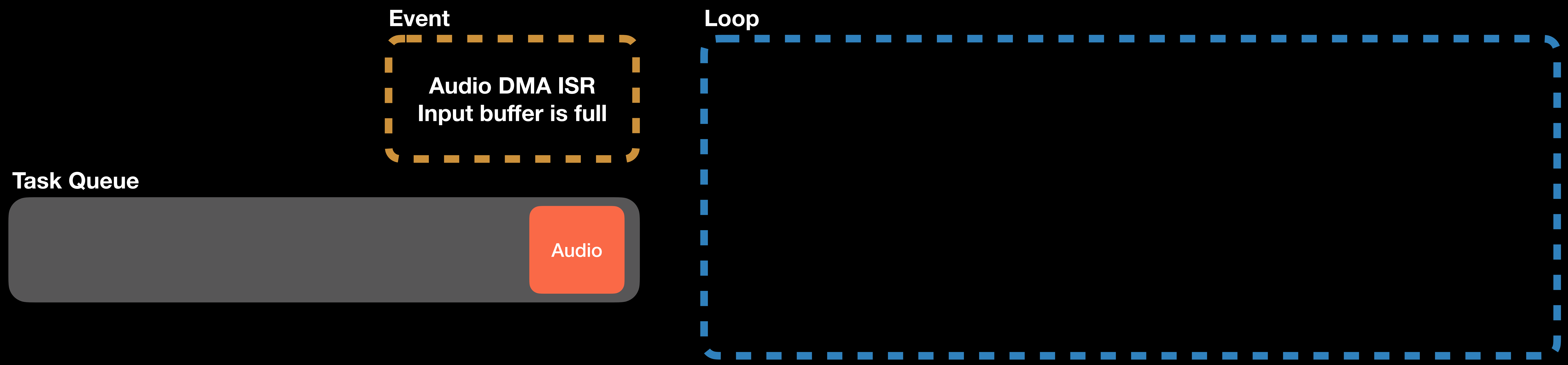
Audio IO

Direct Memory Access



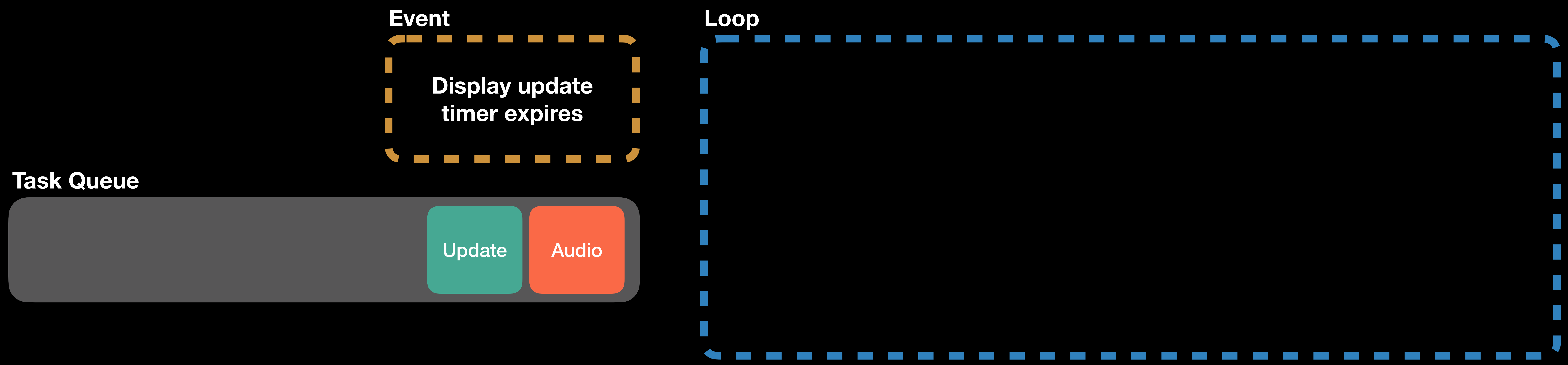
Firmware Architecture

- Managing many different tasks (audio processing, transport, IO, display etc) can be difficult
- Many approaches to task management
- BitMasher uses a *task queue*



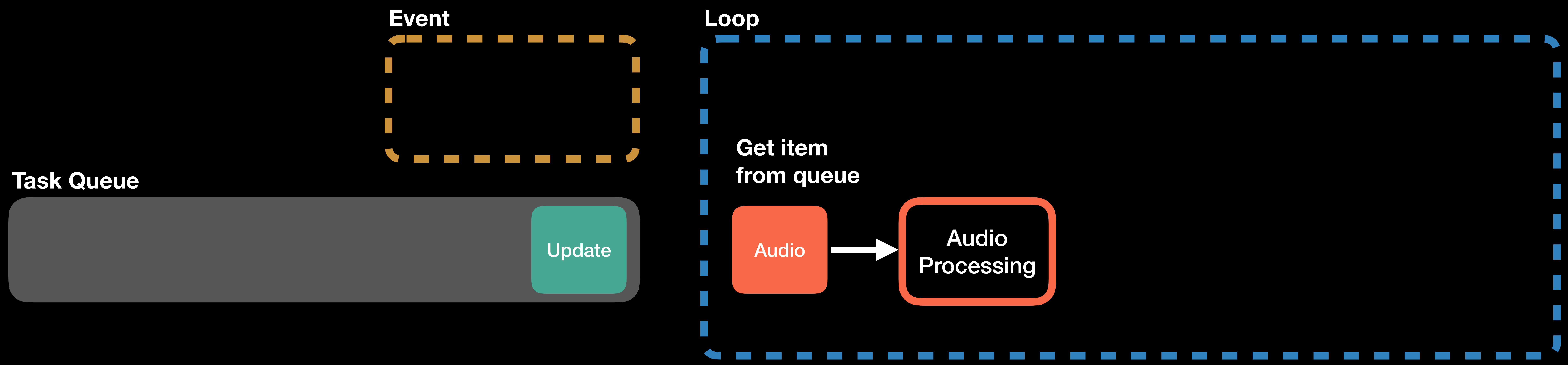
Firmware Architecture

- Managing many different tasks (audio processing, transport, IO, display etc) can be difficult
- Many approaches to task management
- BitMasher uses a *task queue*



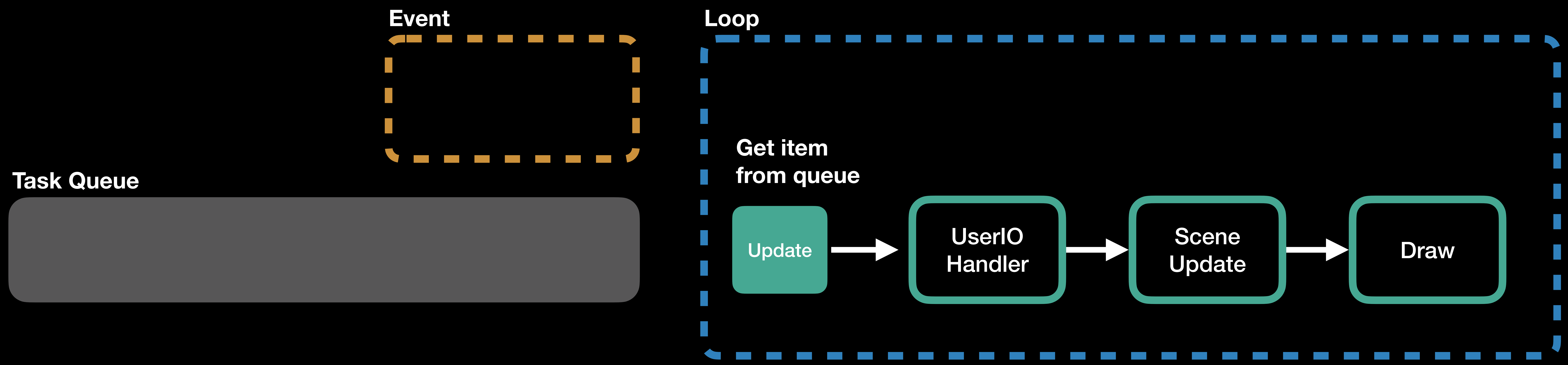
Firmware Architecture

- Managing many different tasks (audio processing, transport, IO, display etc) can be difficult
- Many approaches to task management
- BitMasher uses a *task queue*



Firmware Architecture

- Managing many different tasks (audio processing, transport, IO, display etc) can be difficult
- Many approaches to task management
- BitMasher uses a *task queue*



Firmware Architecture

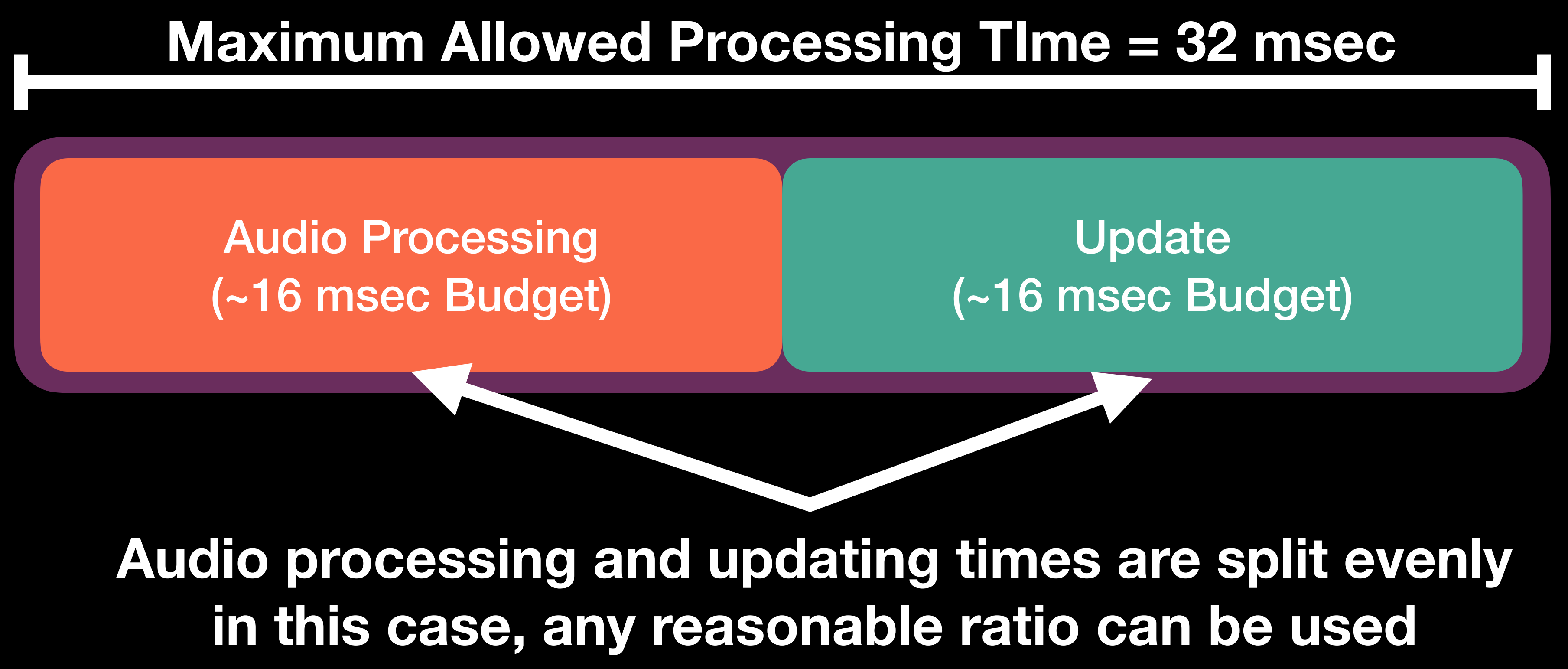
Timing Budgets

- Working with real-time audio means that there are timing constraints!
- Timing can be especially tricky when there are *other tasks*
- To determine the maximum time allowed for *all tasks* to complete, use the task with the highest bandwidth as a basic starting point

Example

Audio — New buffer of output samples expected every 32 msec
fs = 32 kHz
Buffer Size = 1024 samples

Update — Display Refresh Rate = 30 FPS
(33 msec per frame change)



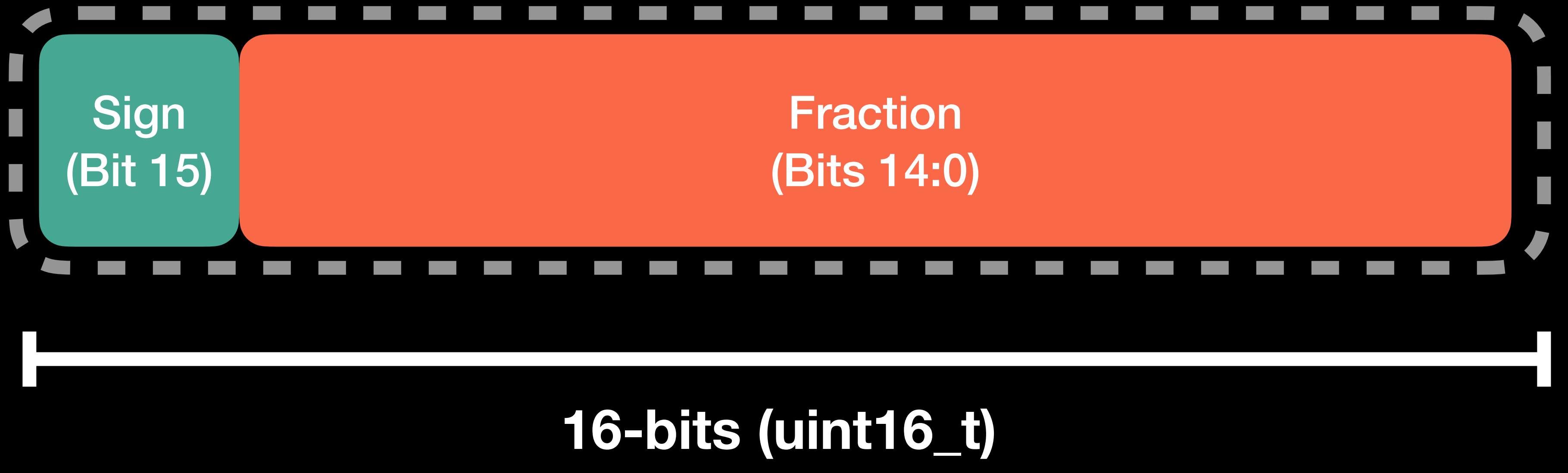
***This illustrates one such timing strategy but many others exist!**
Another technique is to assign priorities to certain tasks and allow higher priority tasks to *preempt* lower priority ones

Audio Processing

Fixed Point Notation

- Many MCUs do not have floating point units (FPU)
- For MCUs with no FPU, floating point operations need to be done *in-software* = *performance hit*
- Because of this, fixed-point notation is often favoured

Q1.15 Fixed Point Number



Lowest Number Representable: $\dfrac{-32768}{2^{15}} = -1$

Highest Number Representable: $\dfrac{32767}{2^{15}} = 0.99997$

Arithmetic Operations

Addition

$$A + B = A + B$$

Subtraction

$$A - B = A - B$$

Multiplication

$$AB = (A * B) >> 15$$

Division

$$\frac{A}{B} = (A/B) << 15$$

*** Note that addition and subtraction operations Run the risk of over/underflow!**

*** When multiplying two Q1.15 numbers, the result should be stored in a 32-bit wide variable before shifting**

Audio Processing

ARM CMSIS DSP Library

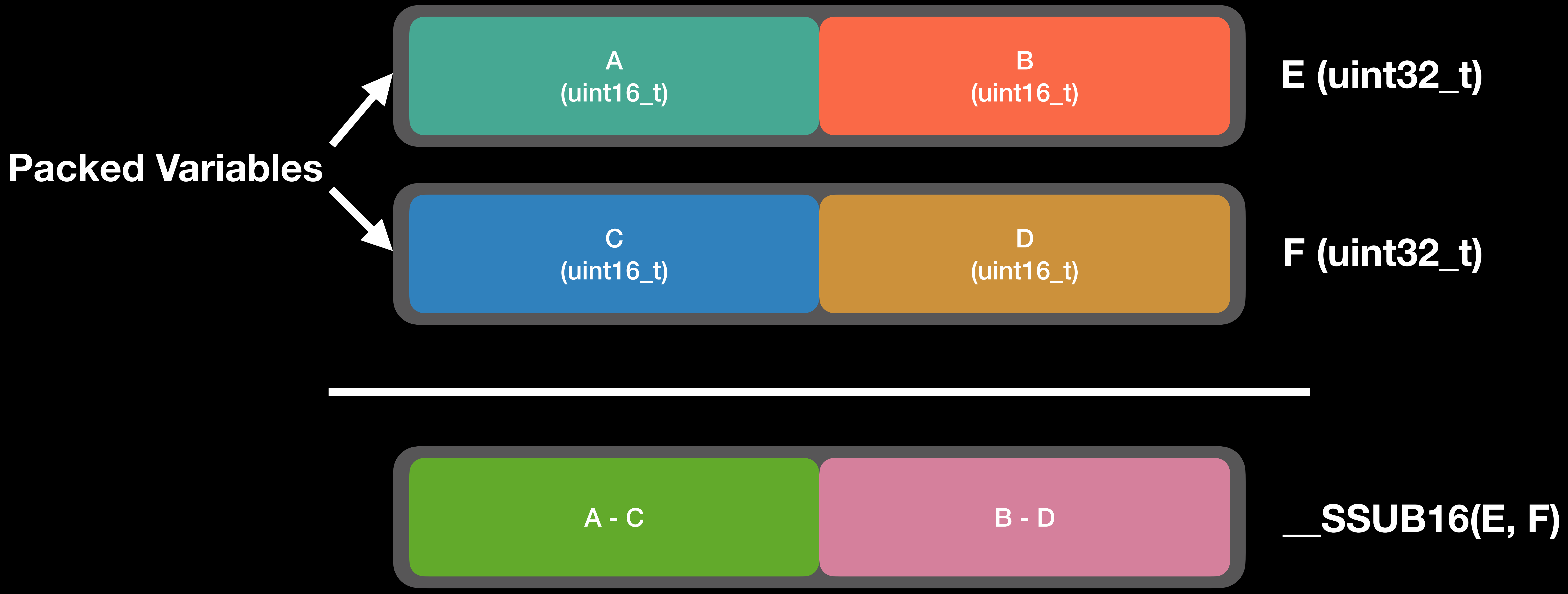
- DSP libraries written by ARM for ARM-based MCUs
- Helpful when optimizing audio DSP algorithms
- Offers:
  - Accelerated math functions
  - Fast trigonometric functions
  - FFT
  - FIR and IIR (Biquad) Filtering
  - And more!

```
case MW_BIQUAD_PARAM_EQ_NCQ:
{
    #ifdef NO_OPTIMIZE
    if (biquad->bufferSize != numSamples) while(1);
    #endif
    arm_copy_f32(buffer, biquad->copyBuffer, numSamples);
    arm_biquad_cascade_df2T_f32(&biquad->biquadInstance, buffer, buffer, numSamples);
    arm_scale_f32(buffer, biquad->coefficients[5], buffer, numSamples);
    arm_add_f32(buffer, biquad->copyBuffer, buffer, numSamples);

    break;
}

case MW_BIQUAD_PARAM_EQ_CQ:
{
    arm_biquad_cascade_df2T_f32(&biquad->biquadInstance, buffer, buffer, numSamples);
    break;
}
```

Audio Processing

SIMD

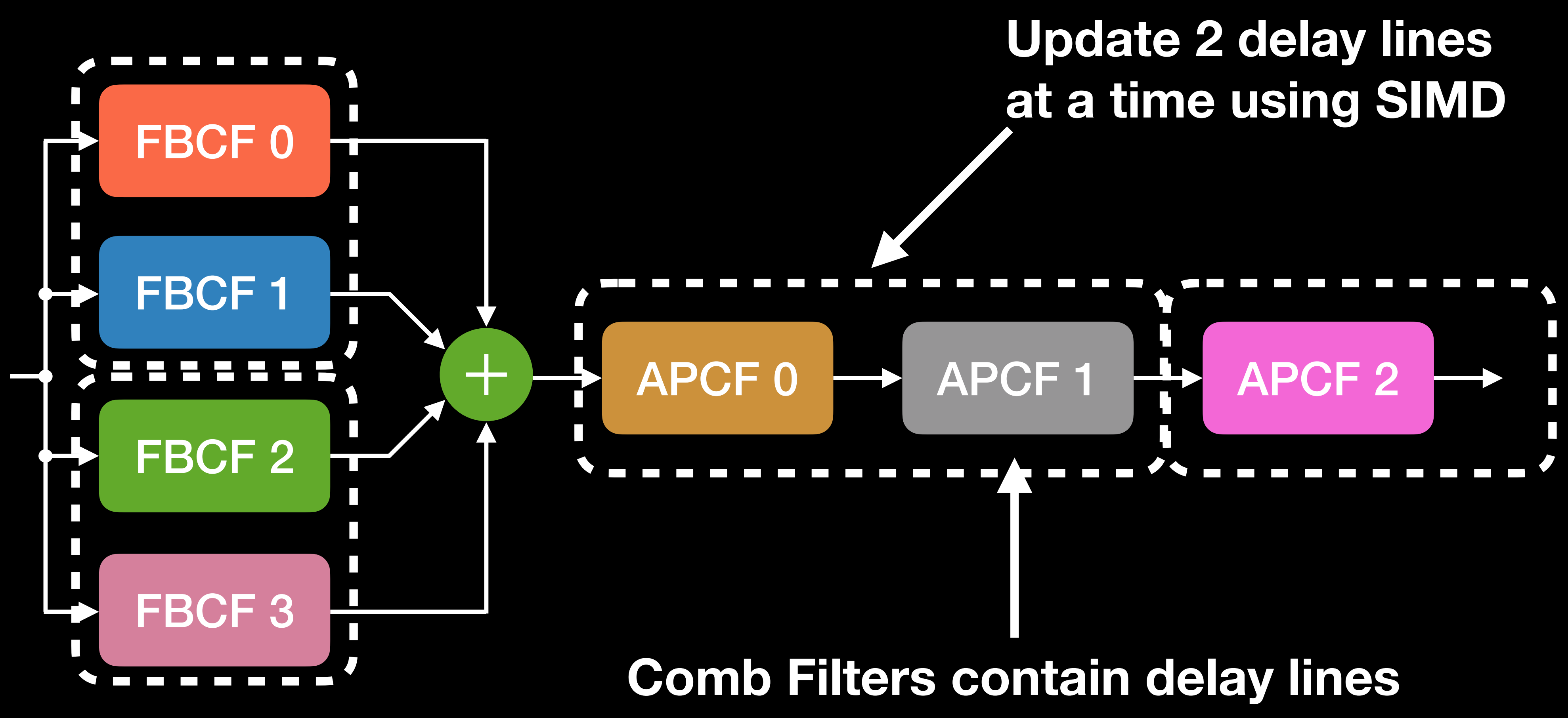
- ARM Cortex M3 and M4 cores offer limited SIMD functionality!
- 32-bit register length
- Not usable for floating point numbers but useful when using fixed point numbers



Audio Processing

SIMD Example

Schroeder Reverberator




```
float32_t MW_DSP_FBCF_tick(MW_DSP_FBCF *filter, float32_t x)
{
    #ifdef NO_OPTIMIZE
    if (filter == NULL)
        while(1);
    #endif

    float32_t v = (x * filter->b0) + (filter->delayLine[filter->currentPtr] * filter->am);
    filter->delayLine[filter->currentPtr] = v;

    filter->currentPtr -= 1;
    if (filter->currentPtr < 0)
        filter->currentPtr = filter->N - 1;

    return v;
}
```

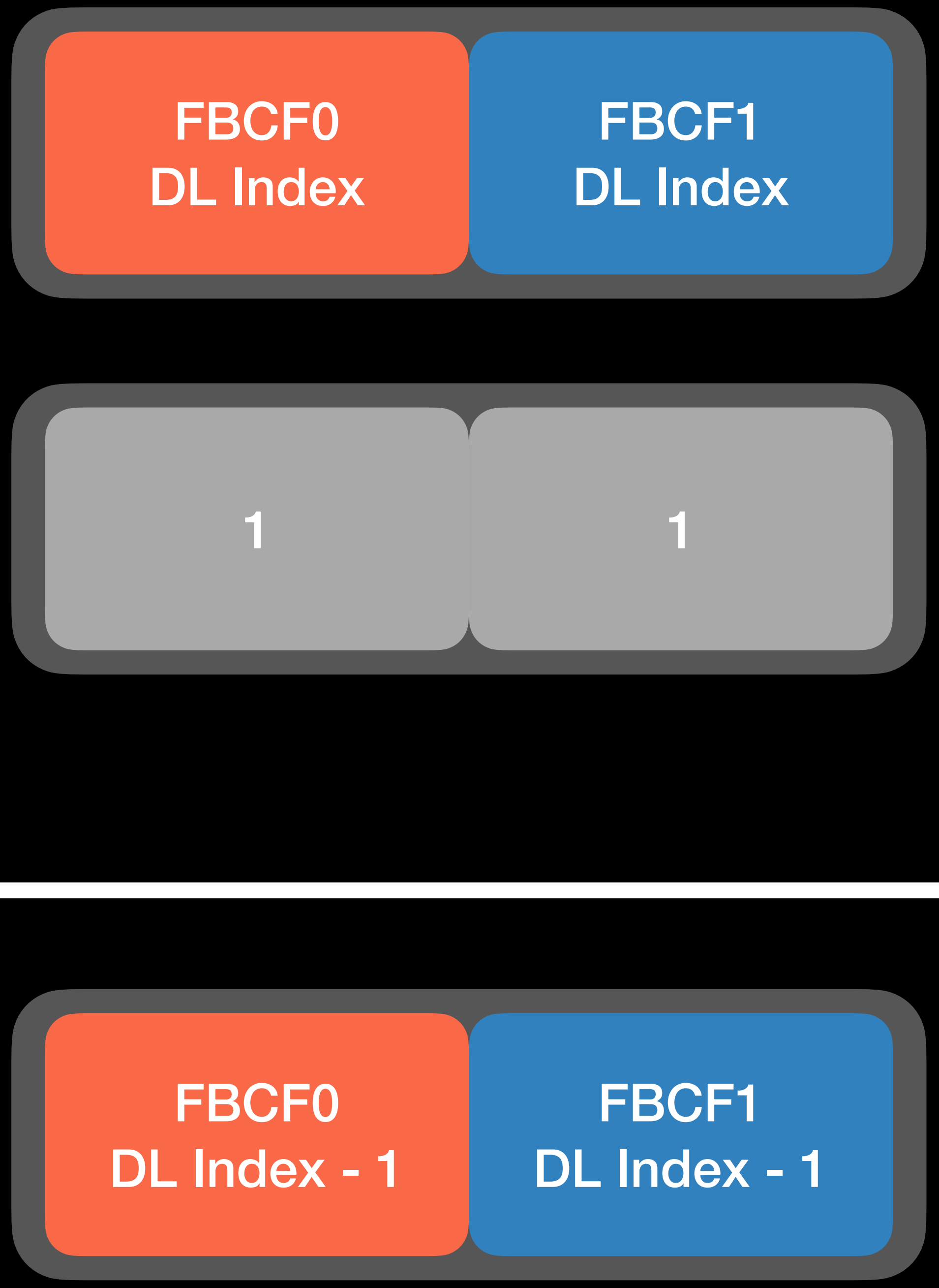
Delay line index decrement
(Point to next element to store and read samples)

Operation occurs for every comb filter

Audio Processing

SIMD Example

Comb Filter Delay Line Index Update



A — Comb Filter Delay Line Lengths
Converted to uint16_t
Packed into single uint32_t variable

B — Constant
2 uint16_t 1's packed into a
single uint32_t variable

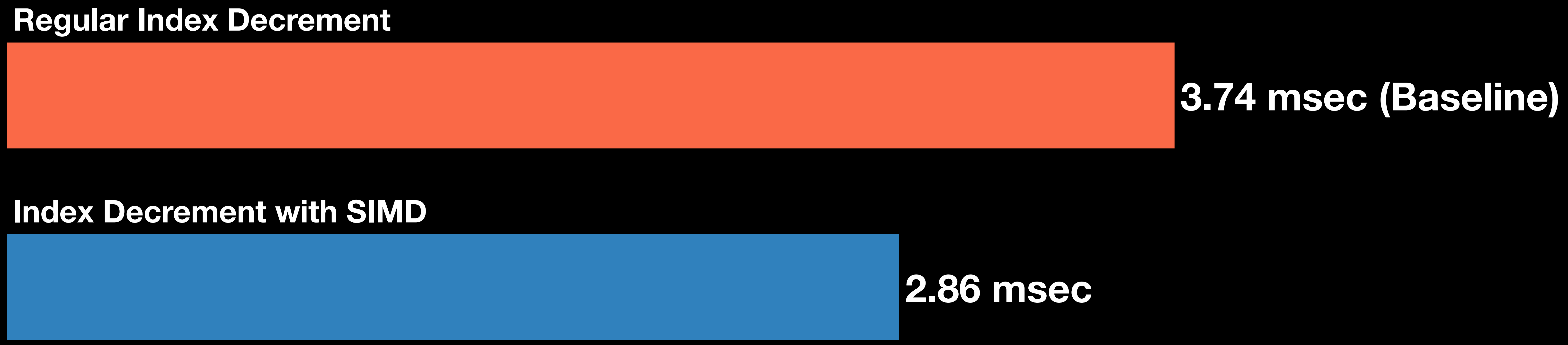
__SSUB16(A, B)

Result

Audio Processing

SIMD Example: Performance Gain

Schroeder Reverberator Processing Time



*Optimization level 3 (GCC) enabled
Processing Buffer Size = 1024 Samples
EFM32PG12 MCU @ 40 MHz

Debugging Tools

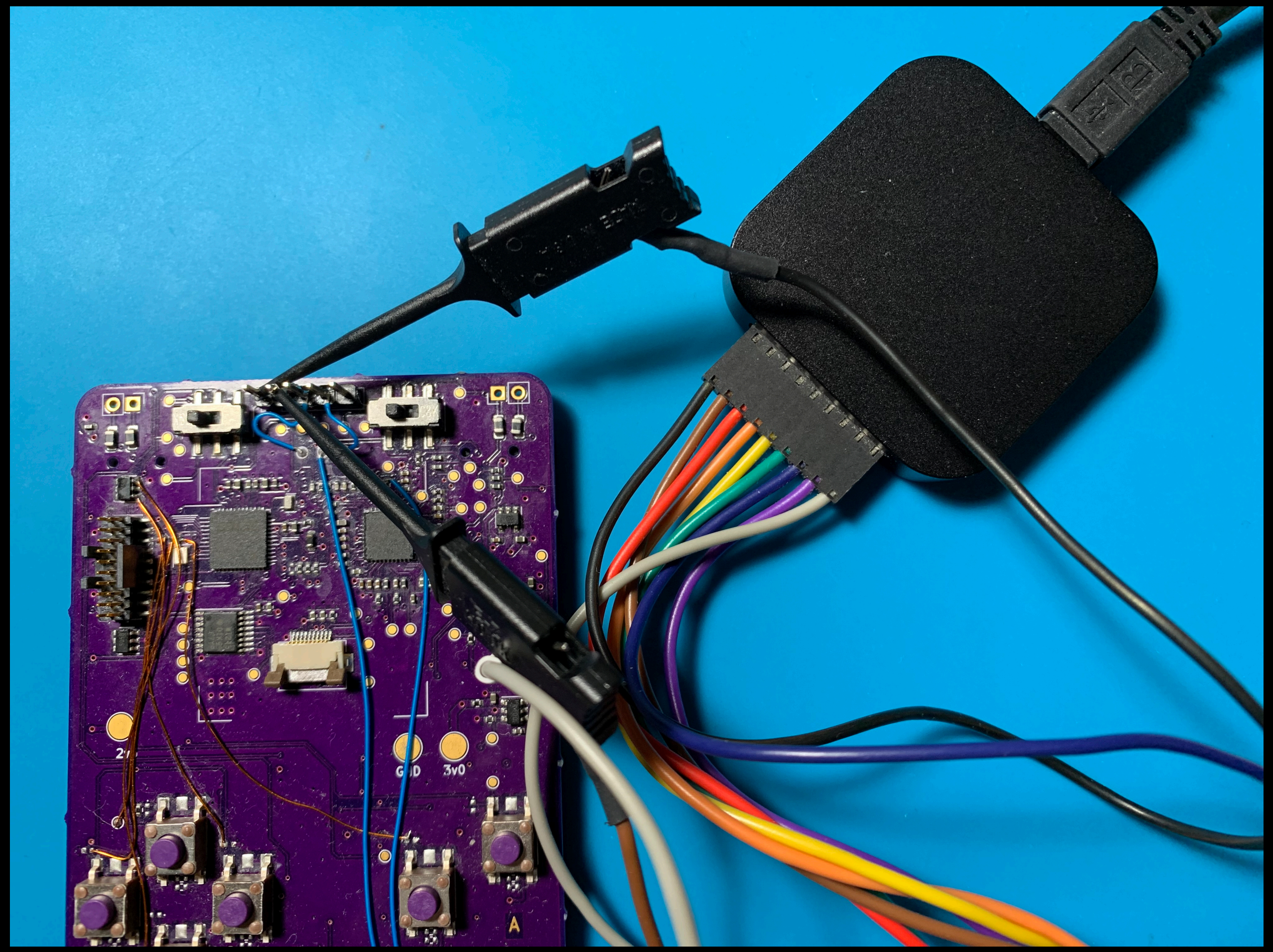
Hardware-Based Tools

- A multimeter is a <u>must</u> when developing circuits
- An oscilloscope is *highly recommended* especially when working with analog circuits
- Pocket oscilloscopes and second-hand markets offer budget-friendly options
- A logic analyzer is indispensable for profiling digital signals and code execution times

Multimeter

Oscilloscope

Logic Analyzer



Performance Profiling

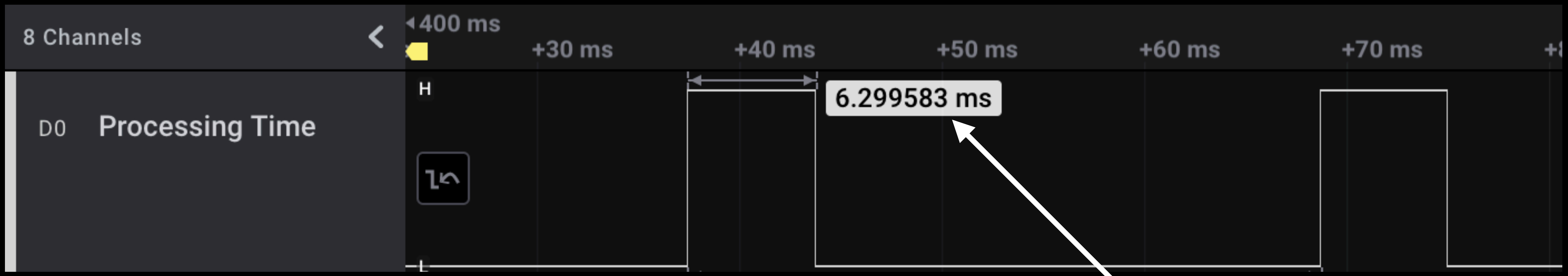
Execution Time Profiling: GPIO Toggle

Set GPIO HIGH before entering code in question then set to LOW when exiting

```
//  Set debug pin (DBP0) HIGH
BM_DebugServices_set(DBP0);

_scenes[_sceneIndex].processAudio(_audioFloatBuffer, AUDIO_BUFFER_SIZE);

//  Set debug pin (DBP0) LOW
BM_DebugServices_clear(DBP0);
```

***If using an API call to toggle pins, be aware that the function may be calling extra code which introduces measurement latency!**

Probe pin with oscilloscope or logic analyzer



processAudio( ) takes 6.3 msec

Performance Profiling

Execution Time Profiling: Hardware Timer

```
CMU_ClockEnable(cmuClock_TIMER0, true);
TIMER_Init_TypeDef init = TIMER_INIT_DEFAULT;
init.enable = false;
init.prescale = timerPrescale1024;

//  Initialize the timer but don't enable it yet!
TIMER_Init(TIMER0, &init);
```

Setup TIMER0 (EFM32PG12 MCU)
Note that increasing pre-scale values will increase time before timer overflow
But granularity will be decreased!

***It is a good idea to have an interrupt enabled in case of timer overflow!**

```
//  Clear the timer and start
TIMER0->CNT = 0;
TIMER_Enable(TIMER0, true);

_scenes[_sceneIndex].processAudio(_audioFloatBuffer, AUDIO_BUFFER_SIZE);

//  Stop the timer and read the counter value
TIMER_Enable(TIMER0, false);
uint32_t timeTaken = TIMER0->CNT;
```

Enable the timer before entry into code
Disable the timer after exit and read timer value

Performance Profiling

Execution Time Profiling: Data Watchpoint Trace

- Some ARM Cortex MCUs include a Data Watchpoint Trace module which includes a counter!
- Some MCU manufacturers may or *may not* choose to implement the DWT Module

```
DWT->CTRL |= 0x01;    //  Enable the CYCCNT register
DWT->CYCCNT = 0;      //  Reset counter value

_scenes[_sceneIndex].processAudio(_audioFloatBuffer, AUDIO_BUFFER_SIZE);

uint32_t numCycles = DWT->CYCCNT;  //  Get number of cycles counted
```

The DWT counter is incremented at each CPU Clock cycle
Like the hardware timer, there is a chance for counter overflow
Therefore, it is a good idea to add an overflow interrupt!

Final Remarks

- Building audio hardware consists of *many* different components!
- First time? Start small!
  - Start with basic circuits and PCBs
  - Arduinos offer a good introduction to bare-metal programming
  - Start with simple FW projects using one peripheral at a time
  - Then slowly work up to more complex systems
- No shortage of amazing online resources!

Resources

- The Art of Electronics (Horowitz and Hill)
- Small Signal Audio Design (Douglas Self)
- EEVBlog (Dave Jones)
- Contextual Electronics (Chris Gammell)
- Op-amp Applications Handbook (Analog Devices)
- The Hitchhikers Guide to Embedded Audio (Tom Waldron, ADC20)
- Bare Metal Audio Programming with Rust (Antoine van Gelder, ADC20)
- Altium Blog (For various PCB design tips)
- Sparkfun (Tutorials and parts)
- Adafruit (Tutorials and parts)

Thank you!