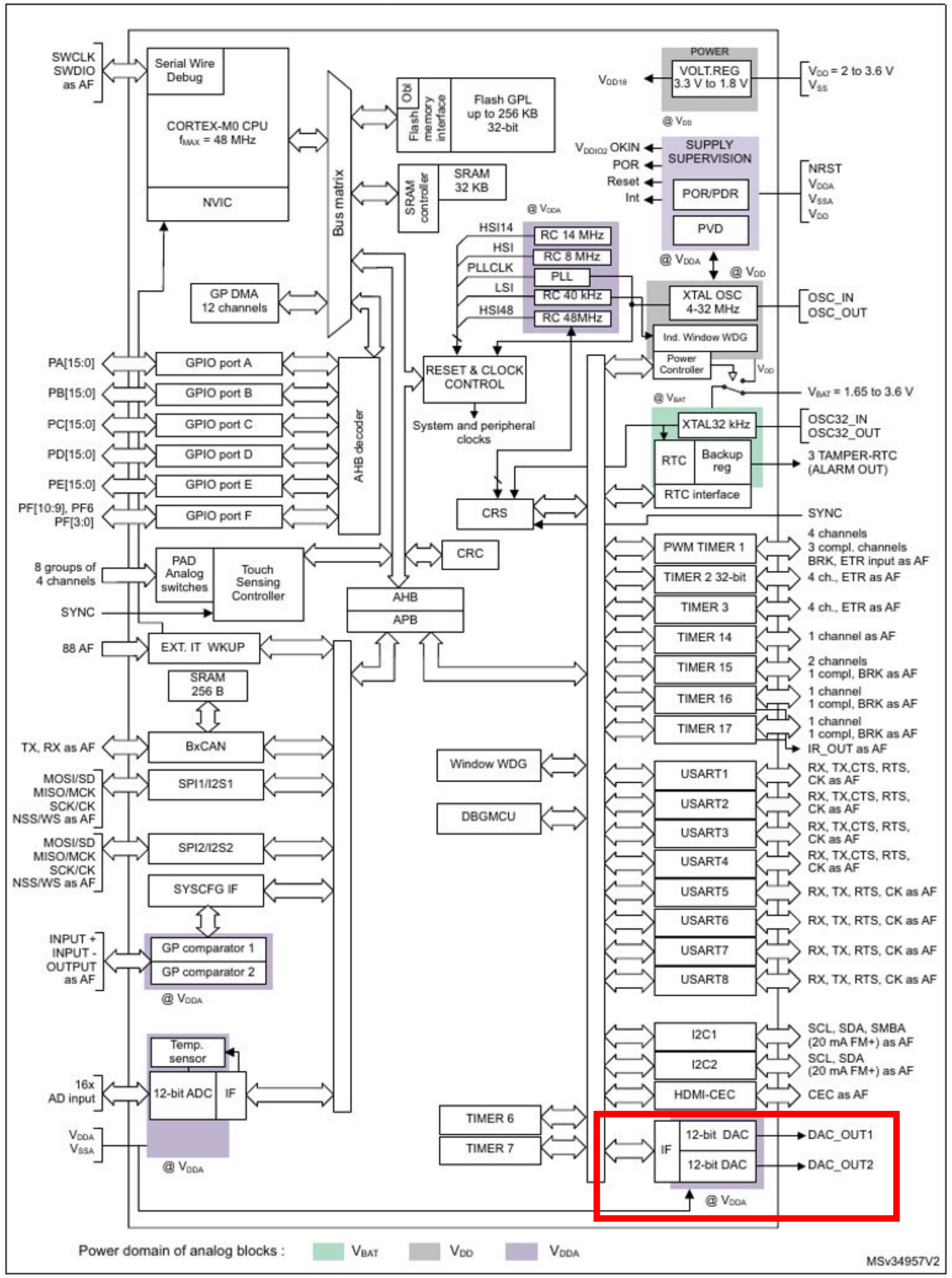
Module 3

Digital to Analog Conversion
(DAC)

Tim Rogers 2022

Reading

- Textbook, Chapter 21, Digital-to-Analog Conversion, pp. 507 – 526.
  - Read this first.
- FRM, Chapter 14, Digital-to-analog converter (DAC), pp. 269 – 281.
  - Scan. Learn basics like I/O registers, enabling, use.
- Textbook, Chapter 20, Analog-to-Digital Conversion, pp. 481 – 506.
  - Read this later.
- FRM, Chapter 13, Analog-to-Digital converter (ADC), pp. 229 – 268.
  - Scan this later. Learn basics like I/O registers, enabling, use.

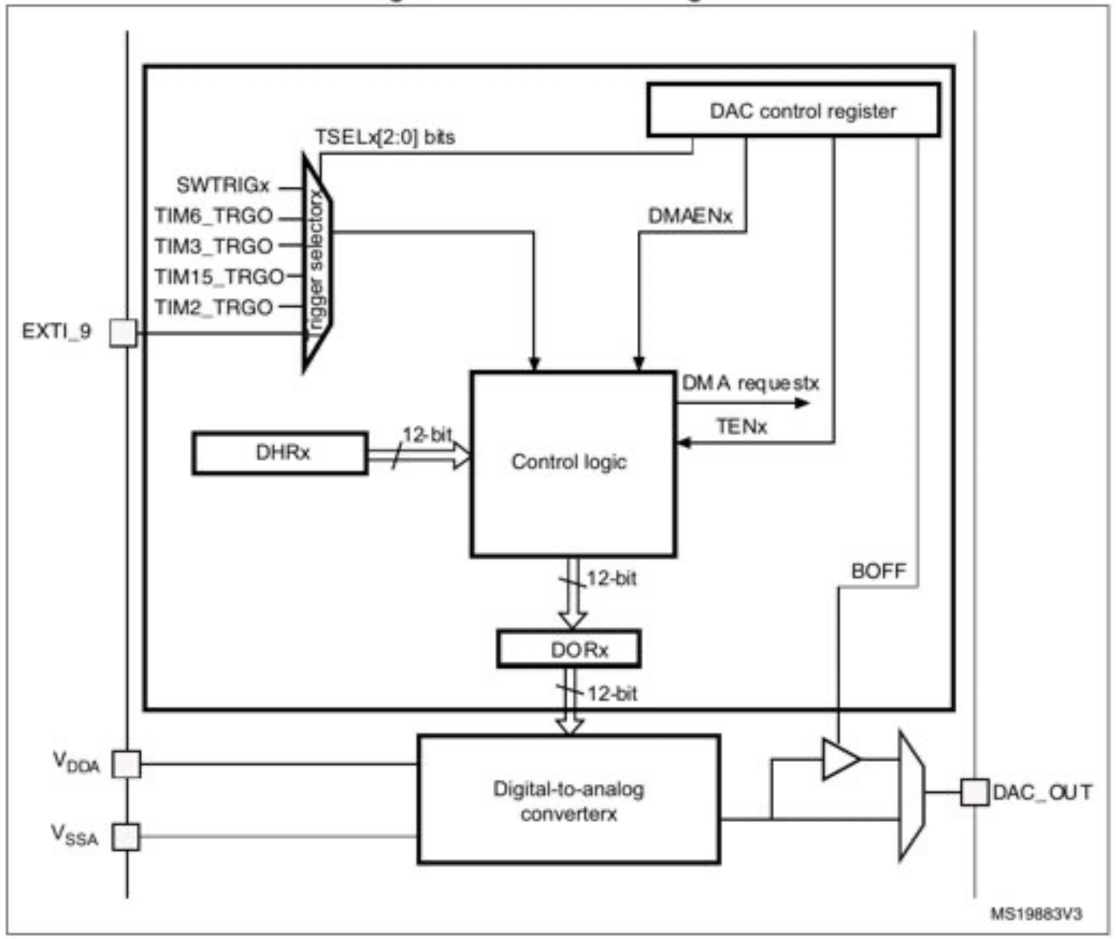


DAC on our micro

- Digital-to-Analog Converter
  - 2 external channels (DAC_OUT1, DAC_OUT2)
    - What pins are those?
      - You can look those up. You know how.
      - Or pay close attention to the lecture.
  - 0 – 3.6V conversion range
  - Our voltage reference is about 3.0V
  - 12-bit resolution, by default
  - 8-bit resolution too
  - right-aligned (default) or left-aligned

DAC Block Diagram

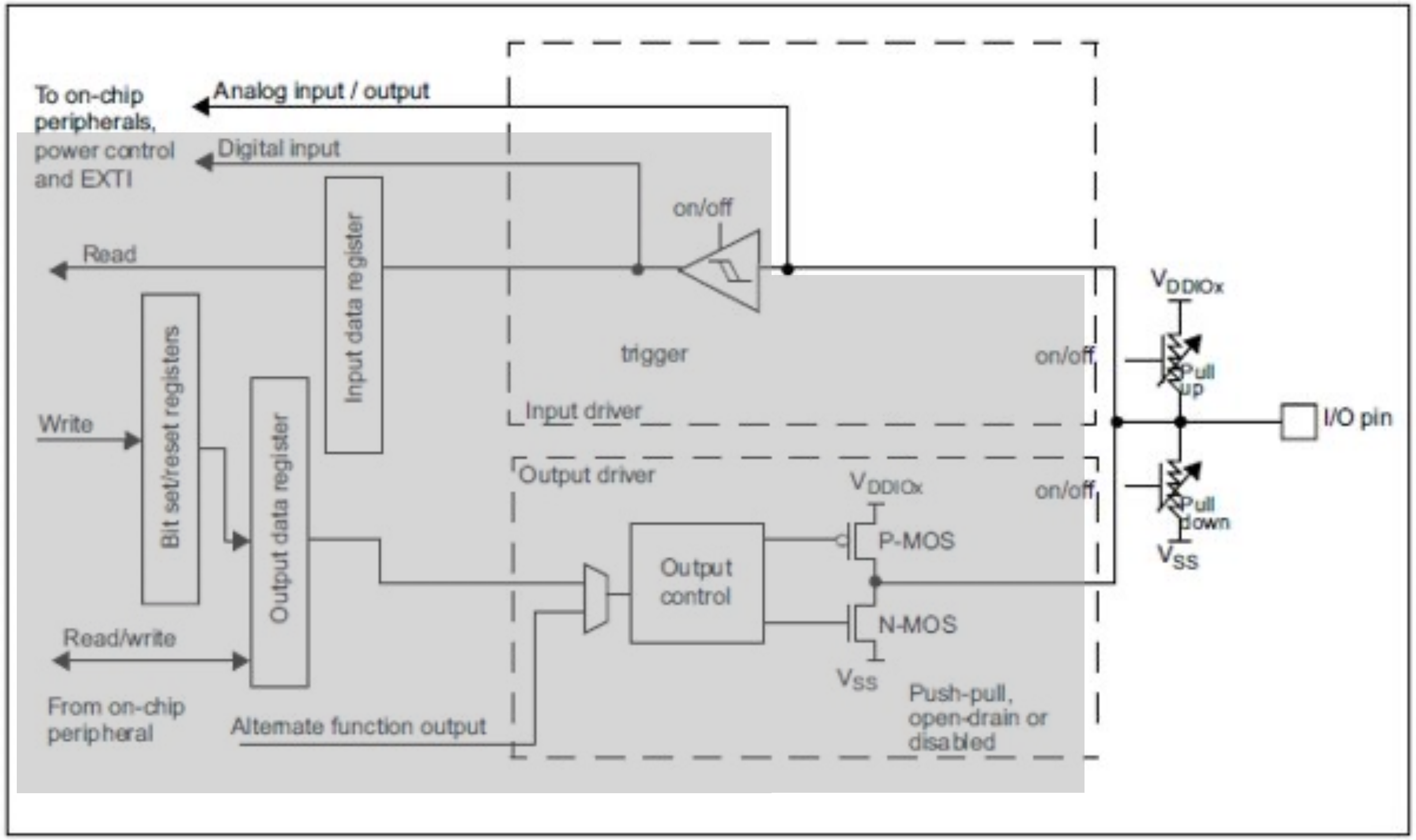
Figure 44. DAC block diagram



- A Data Output Register (DORx) holds a value that is converted to an analog voltage between 0 and 3V.
- CPU can't write directly to DORx.
- Must write to DHRx.
  - "Data Holding Register"
  - A trigger will move DHRx to DORx.

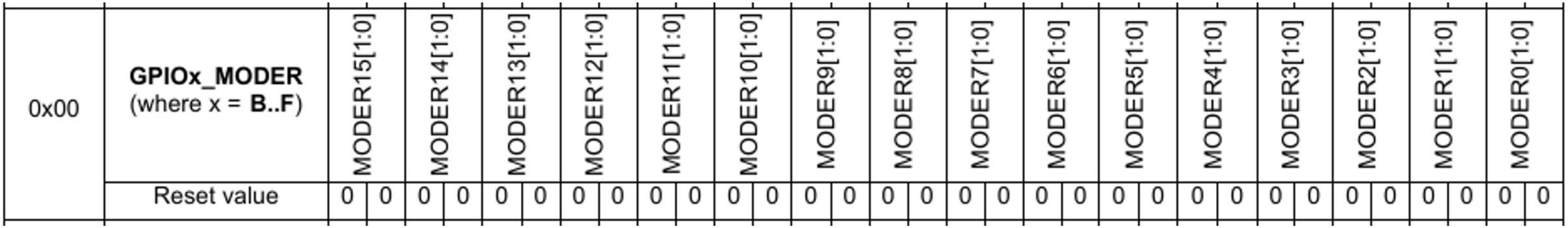
DAC Output Connection

- Some analog pin(s) can be output.



Port MODER

- 16 2-bit values determine, input, output, or special operation.
  - 00: Port pin is used for input
  - 01: Port pin is used for output
  - 10: Port pin has an alternate function
  - **11: Port pin is an analog pin**



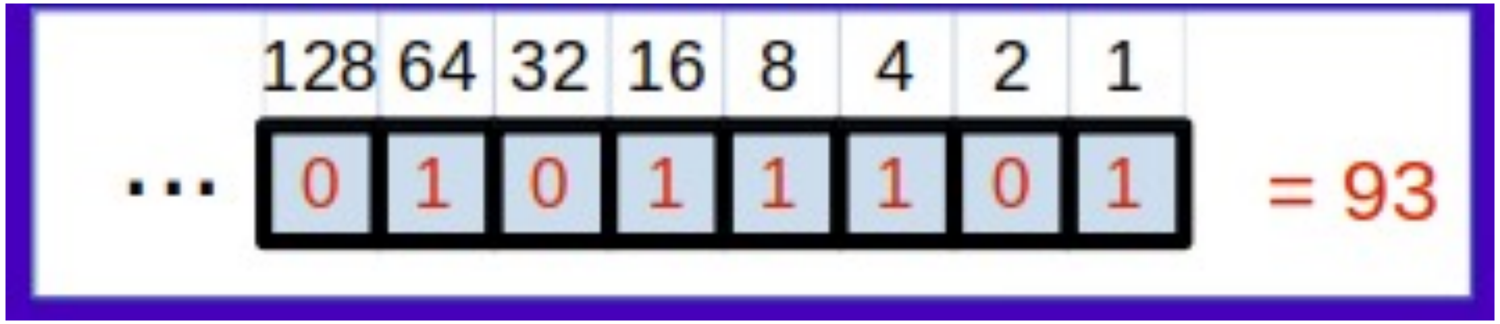
0x00	**GPIOx_MODER** (where x = **B..F**)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

How does a DAC work?

- And what does this left-aligned or right-aligned mean?

First: Number systems

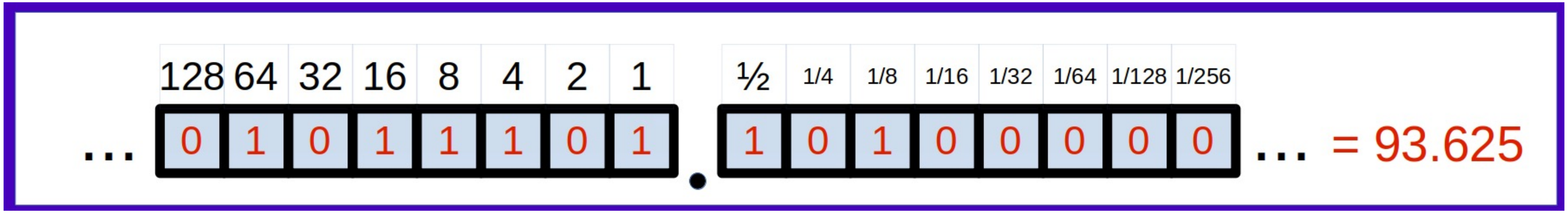
- A group of bits can be used to represent arbitrary things
- A character, a number, an image, …
- "Integers" have a well-known format



- Ellipsis goes off to the left.
- What if we added another bit to the right?

Binary fractions

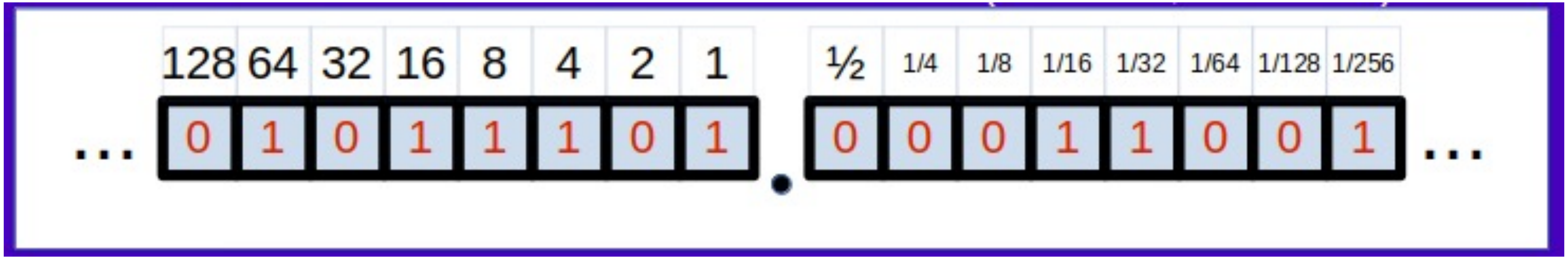
- We can let a 16-bit value represent anything we want, so let's let the upper 8 bits represent an integer, and let the lower 8 bits represent fractional powers of 2
- The dot is called a "binary point"
- Representation, below, is called "fixed point", Q8.8, or just Q8.



- Standard binary addition and subtraction still works.
- Multiplication needs a correcting shift afterward.

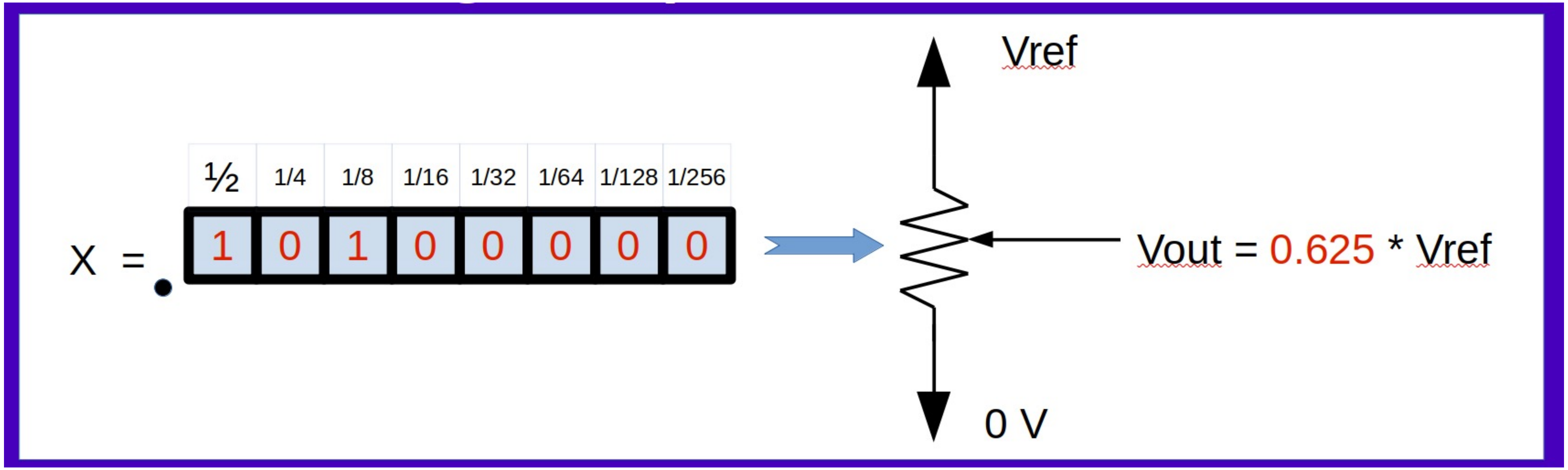
Fixed point is inexact

- Sums of fractional powers of 2, OK.
  - e.g. We can express 3.75 as 00000011.1100….
- Anything else has rounding errors. e.g. 93.1.
- 93 + 1/8 is 93.125 (too big)
- 93 + 1/16 + 1/32 = 93.9375 (too small)
- 93 + 1/16 + 1/32 + 1/256 + 1/512 = 93.099609375 (too small, but closer)



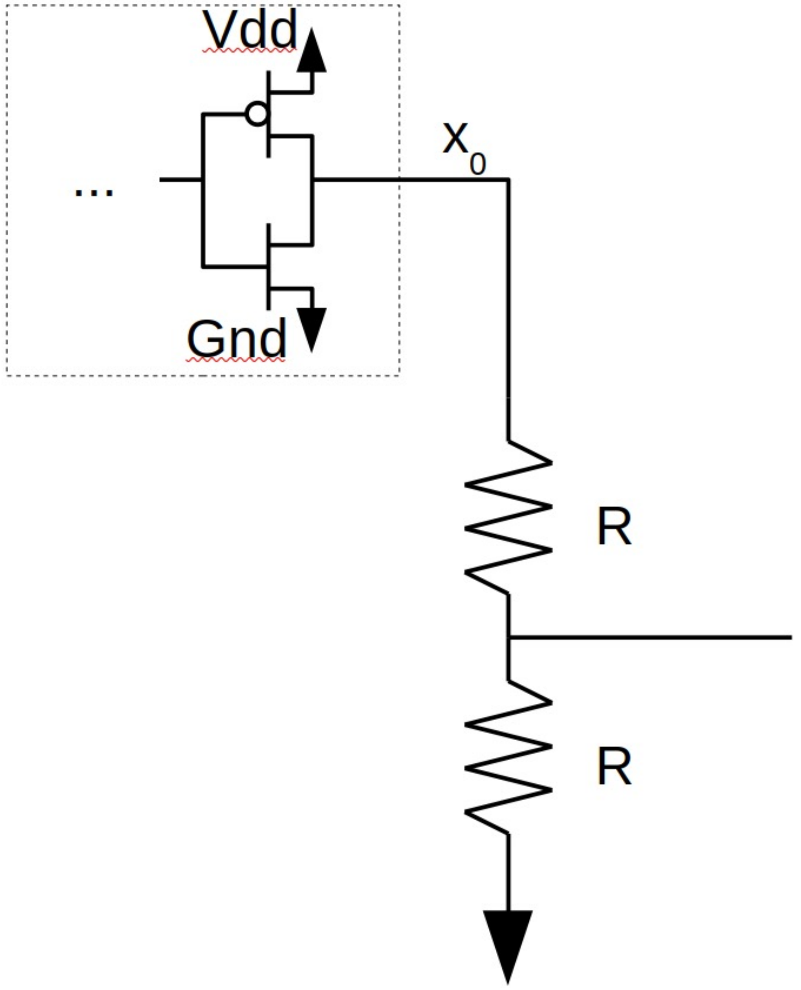
- The precise binary value would repeat forever:
  01011101.000110011001100...

Digital Potentiometer



- We want hardware that can convert a binary fraction to a voltage.
- The most significant bit of the fraction is “worth” half the reference voltage.
- A “left-aligned” conversion value is, effectively, a binary fraction.
- How can we build such a thing?

One-bit DAC



Typical logic gate outputs either "push" high or "pull" low.

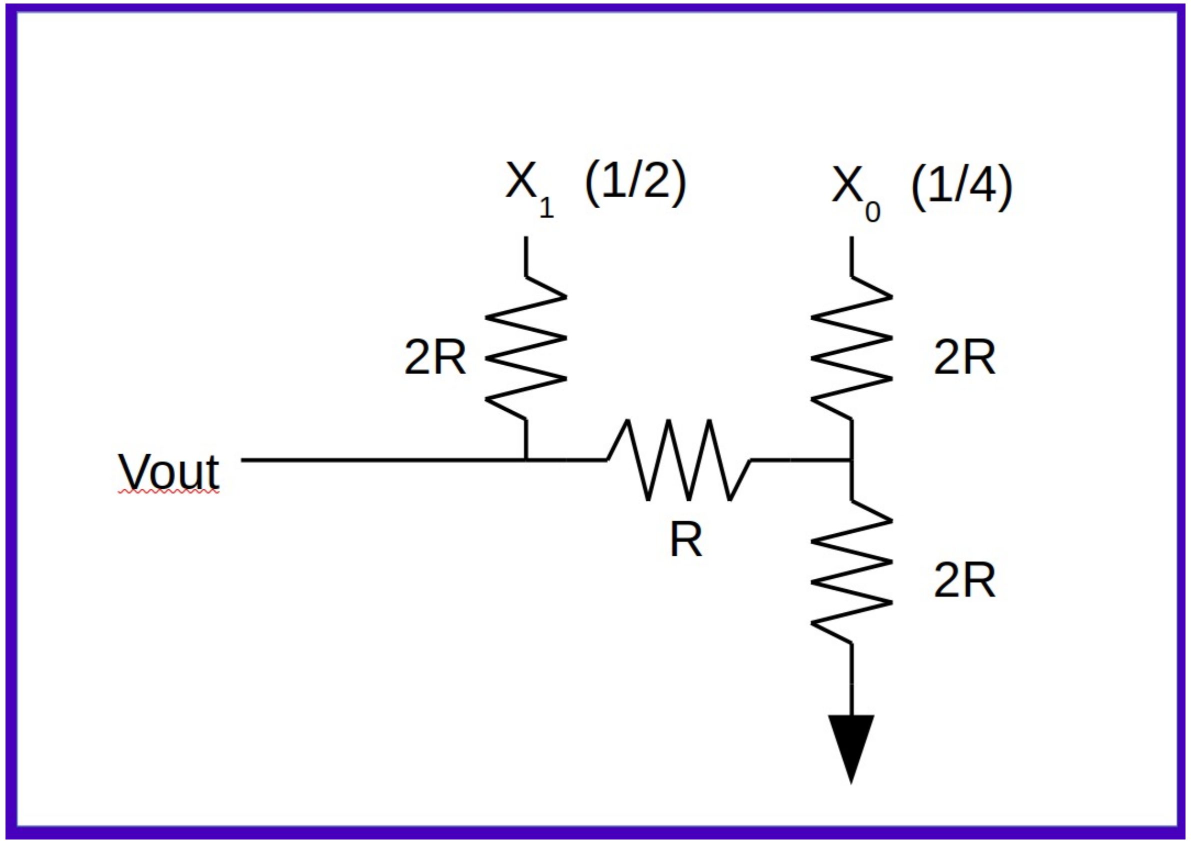
Let Vref = Vdd.

With only one bit, the value x can represent either ½ or 0.

Vout = x_0 * Vref / 2

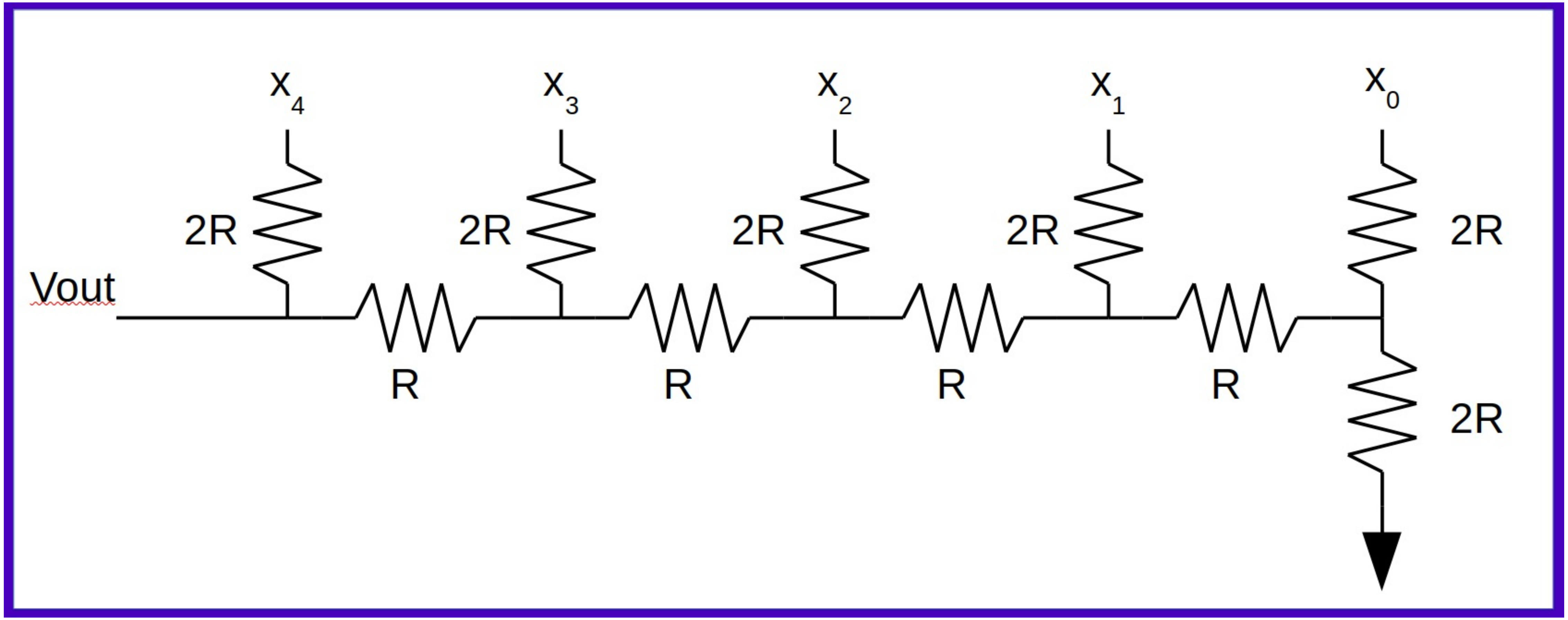
Two-bit DAC

- X can have the values 0, ¼, ½, or ¾.



Many-bit DAC

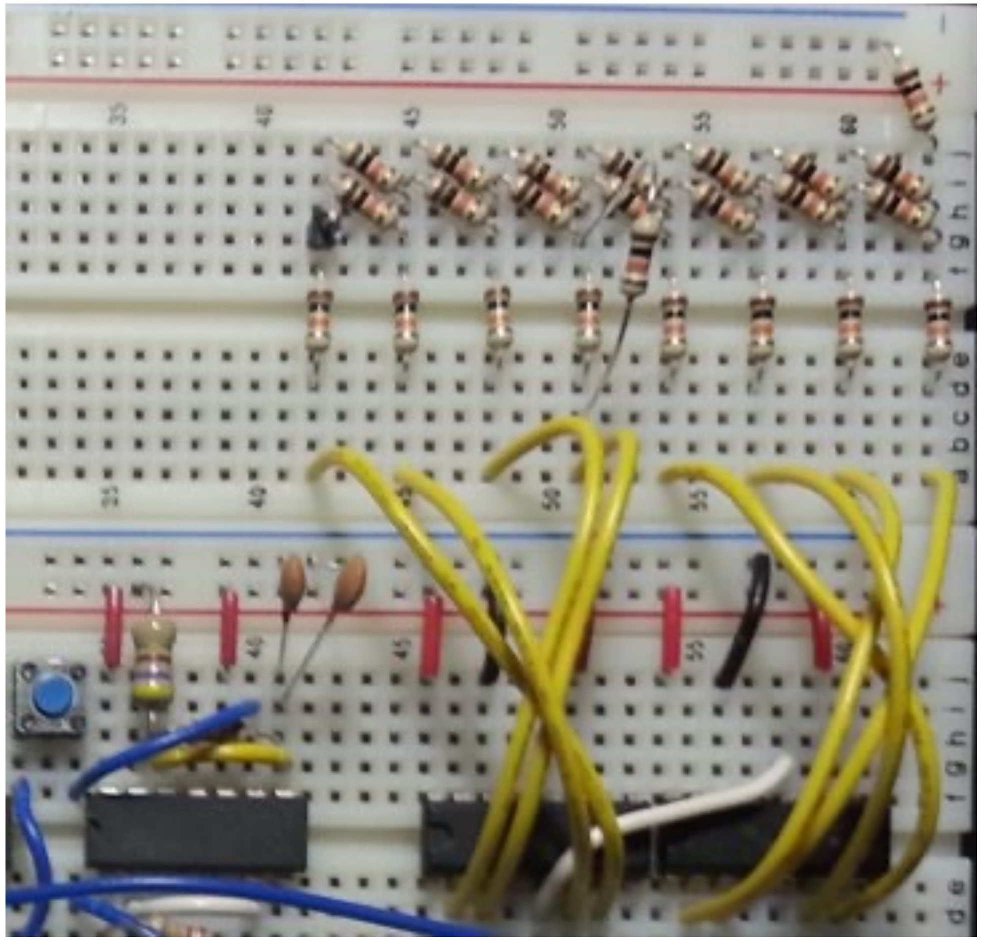
- Add N more bits by adding N more R-2R resistors...
- This is called an R-2R resistor ladder.



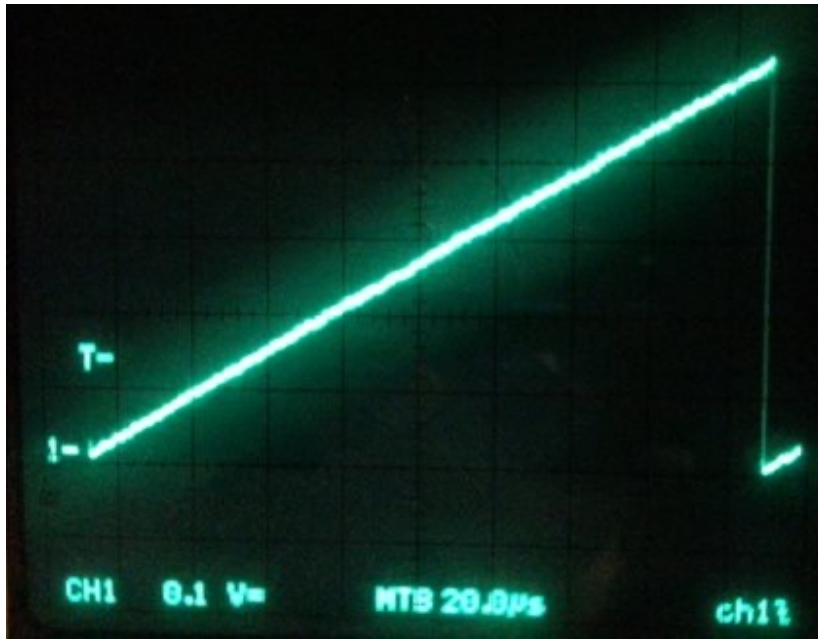
DAC: How fast is the conversion?

- Speed of light
  - Some settling time required due to capacitance (and inductance) in circuit.
  - Small glitches may be present due to differences in delays for turning on and off inputs into the resistor network.
- No iteration or convergence needed.

Build and test...

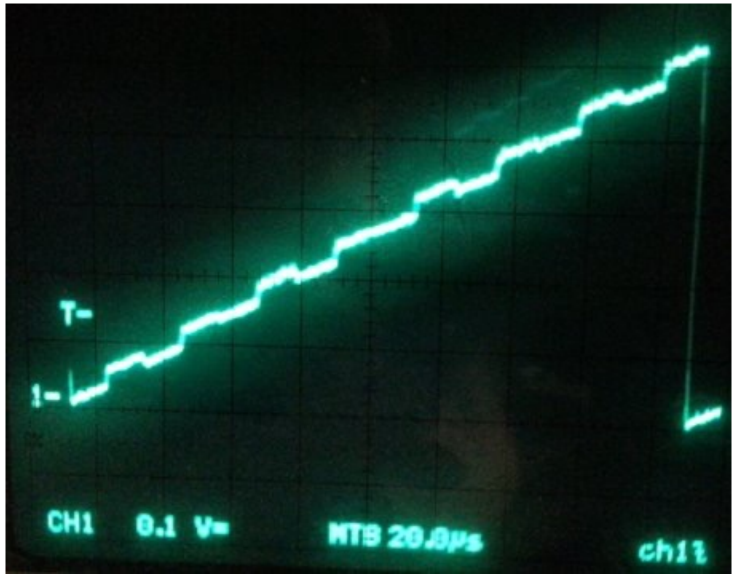


- Build something like this, connect it to a counter, and you get a sawtooth wave.



Limits of DAC resolution

- Resistor tolerance is a limitation.
- Linearity of the output is only as good as the correctness of the resistor ratios.
- A single errant resistor can be problem.
- Small deviation will cause small error.
- Large deviation causes a non-monotonic output for the sawtooth wave.
- To ensure a monotonic sawtooth, each resistor should be less than +/- 1/(2N) its listed value. Why?
    - Consider transition from 01111 to 10000.
        - If most significant resistor was off by -1/(25) the two values would be the same.
    - For an 8-bit DAC, resistor error must be < 0.4%
    - For a 12-bit DAC, resistor error must be < 0.024%
- Helps if all resistors are on same silicon substrate!
    - Which is what you have on your microcontroller.

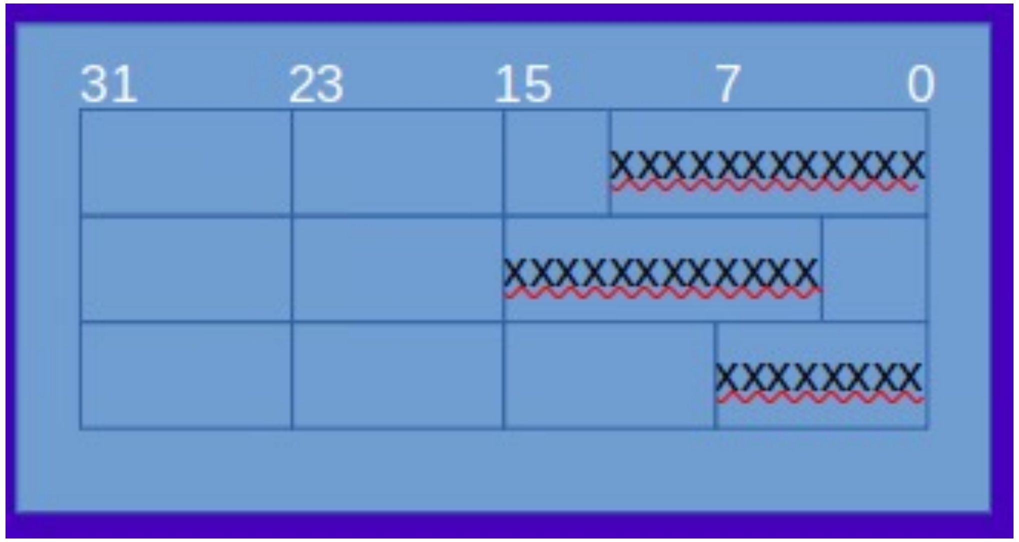


Using the STM32 DAC

- There are two DACs on the STM32F091RCT6.
  - That can be used with two channels / pins.
- Cannot write directly to a DAC output register.
  - Instead, write a digital value to one of three "data holding registers."
  - Some kind of trigger will copy it to the output register.

DAC Holding Registers

- 8-bit or 12-bit
- left-aligned or right-aligned



DAC_DHR12R1: 12-bit, right-aligned, ch1

DAC_DHR12L1: 12-bit, left-aligned, ch1

DAC_DHR8R1: 8-bit, right-aligned, ch1

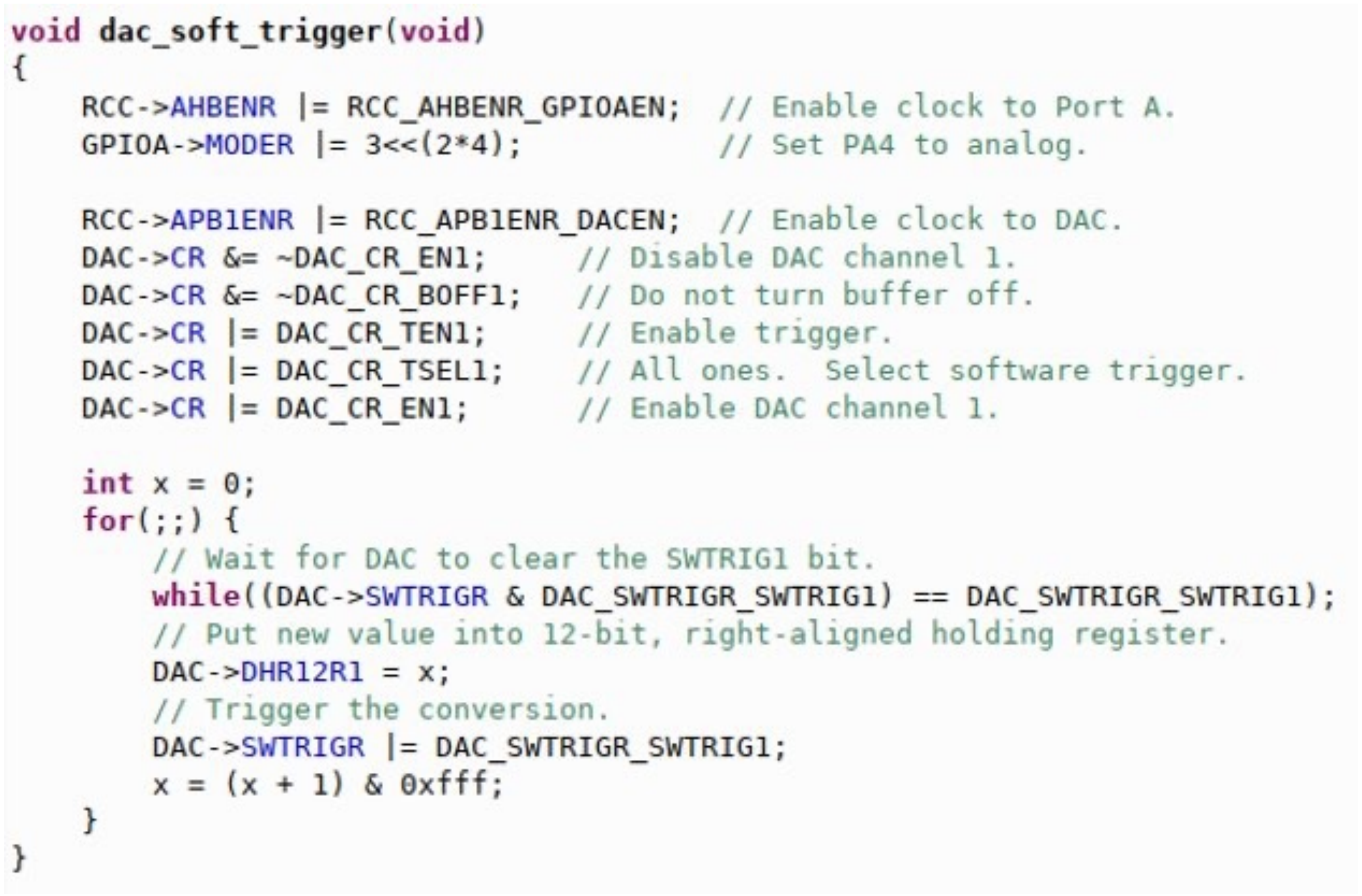
DAC Triggers

- Seven different trigger possibilities
  - 5 timers
  - EXTI line9
  - Software trigger
- Set with the DAC->CR TSEL1/2 fields.
  - 111 is the software trigger. Use this for now.

To trigger the DAC with a software trigger

- Wait while the channel trigger is set. Once it is clear:
    - Write a new value to a holding register.
    - Set the trigger bit.

Example code to generate a sawtooth wave

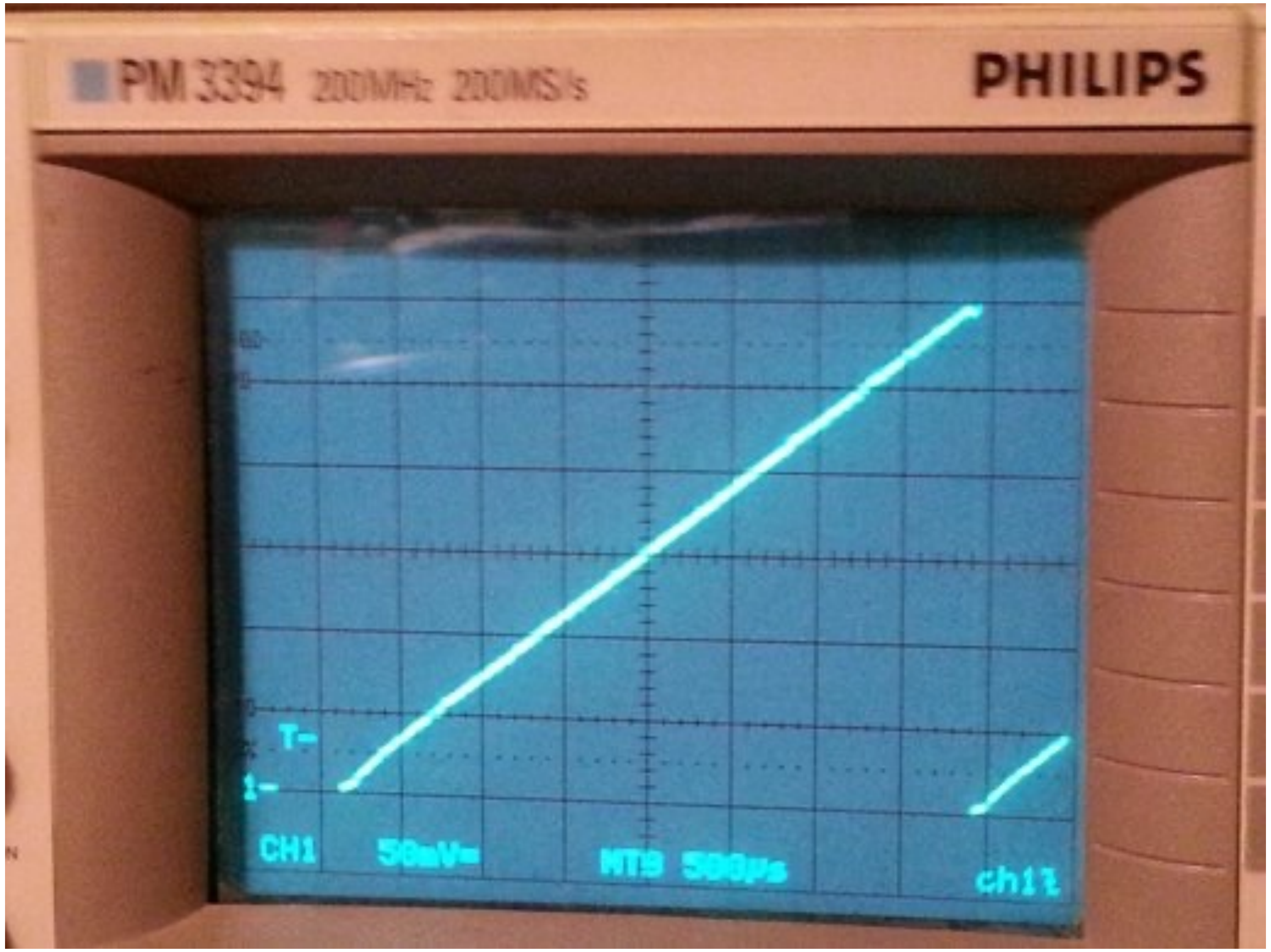


```c
void dac_soft_trigger(void)
{
    RCC->AHBENR |= RCC_AHBENR_GPIOAEN;  // Enable clock to Port A.
    GPIOA->MODER |= 3<<(2*4);           // Set PA4 to analog.

    RCC->APB1ENR |= RCC_APB1ENR_DACEN;  // Enable clock to DAC.
    DAC->CR &= ~DAC_CR_EN1;     // Disable DAC channel 1.
    DAC->CR &= ~DAC_CR_BOFF1;   // Do not turn buffer off.
    DAC->CR |= DAC_CR_TEN1;     // Enable trigger.
    DAC->CR |= DAC_CR_TSEL1;    // All ones.  Select software trigger.
    DAC->CR |= DAC_CR_EN1;      // Enable DAC channel 1.

    int x = 0;
    for(;;) {
        // Wait for DAC to clear the SWTRIG1 bit.
        while((DAC->SWTRIGR & DAC_SWTRIGR_SWTRIG1) == DAC_SWTRIGR_SWTRIG1);
        // Put new value into 12-bit, right-aligned holding register.
        DAC->DHR12R1 = x;
        // Trigger the conversion.
        DAC->SWTRIGR |= DAC_SWTRIGR_SWTRIG1;
        x = (x + 1) & 0xfff;
    }
}
```

Scope view of unloaded pin



- 10x probes.
- 0.5V/div
- 500µs/div
- slight curve on lower end

Conversion Examples

- Assume that VREF is 3.0 V.
- If the DAC's right-aligned 12-bit data holding register was given the following values, what would the analog output voltage be?
- 0 ?
- 4095 ?
- 2047 ?
- 1023 ?

Output linearly related to DAC value

- Generally, the DAC output will be:

$$V_{OUT} = V_{REF} * \frac{N}{4096}$$

Some differences

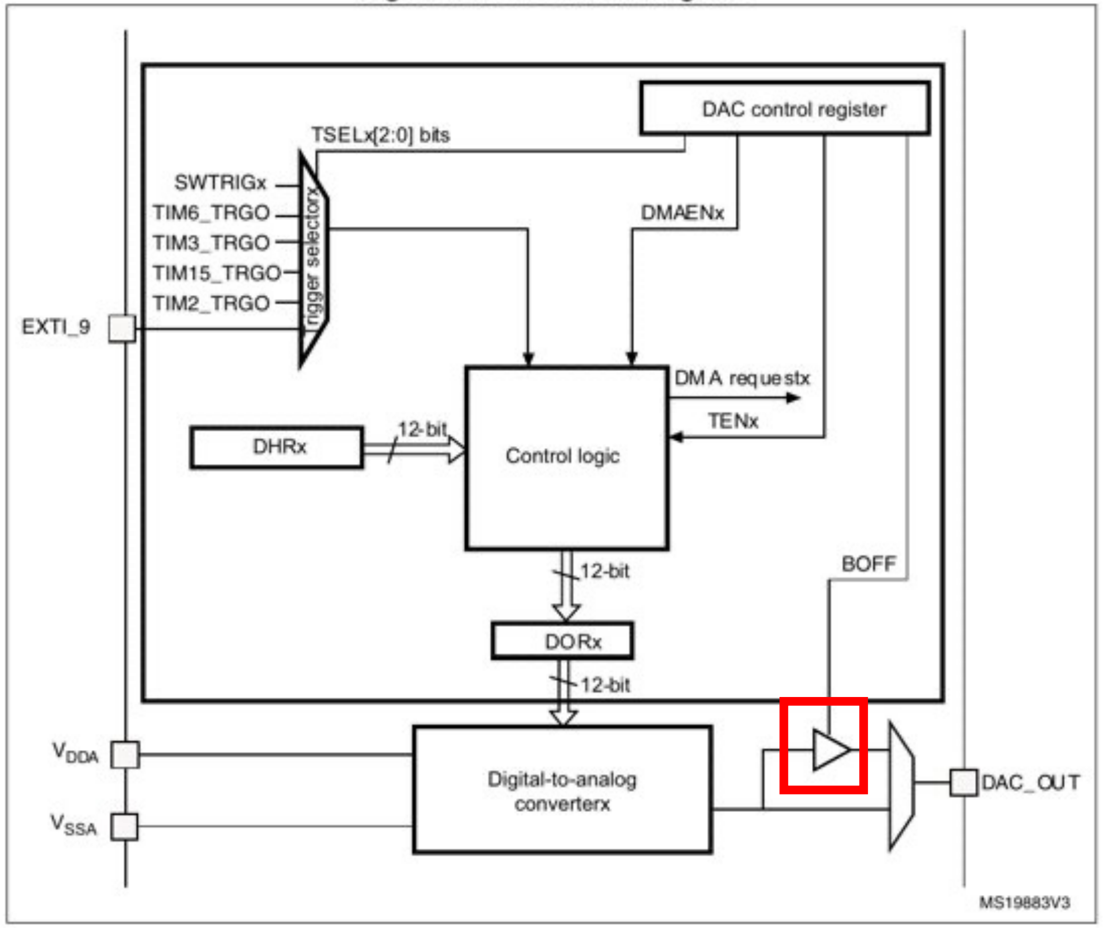
- Lecture explanation of a DAC differs from that of the textbook and the FRM.
  - Lecture: Each increment of the DAC value represents 1/4096th of VREF
    - Minimum: 0 * VREF
    - Maximum: 4095/4096 * VREF
  - FRM: Each increment of the DAC value represents 1/4095th of VREF
    - Minimum: 0 * VREF
    - Maximum: 4095/4095 * VREF

Does this difference matter?

- The difference between the lecture and textbook version of a DAC is 1/4096th VREF – or about 0.024%. But nothing is exact...
  - There are deviations from linearity for the DAC.
  - There is often a noise problem with the analog subsystems of the STM32F0.
    - It's about 0.02%. (More about this when we talk about ADC)
- Difference between this lecture and text doesn't matter.
  - You won't notice a range of [0 – 2.9928] instead of [0 – 3.0]

What is BOFF1?

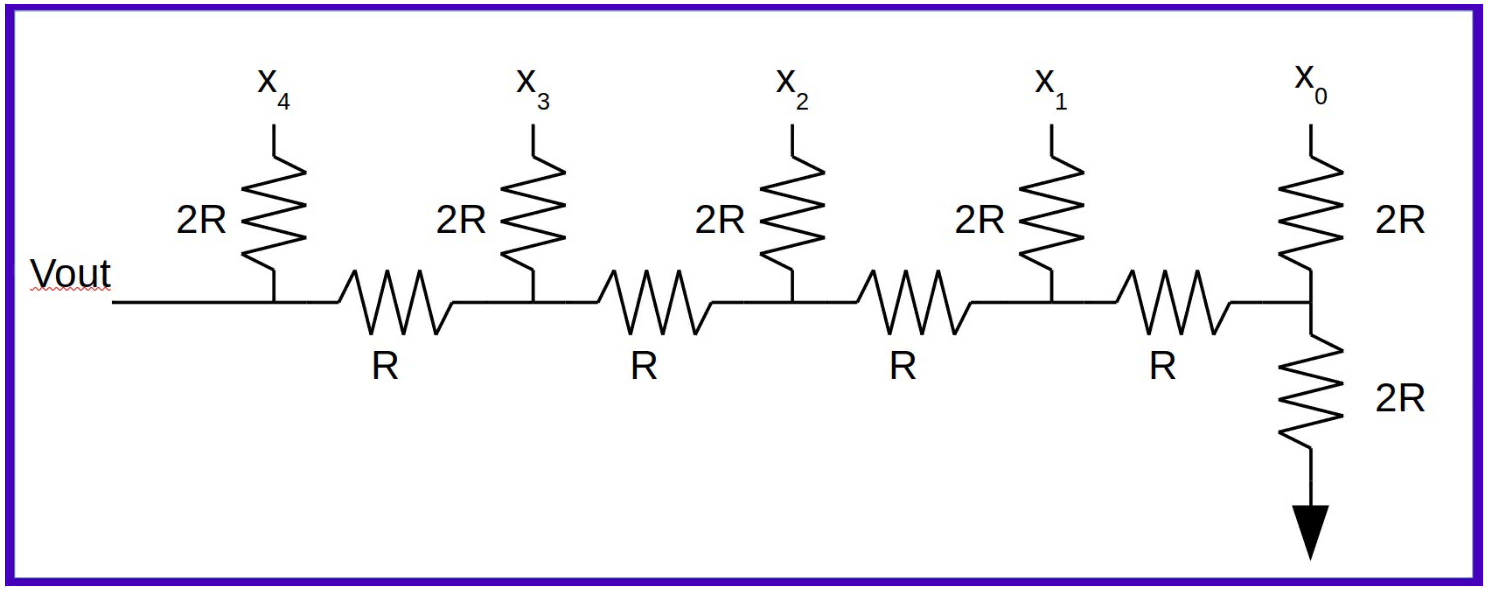
Figure 44. DAC block diagram



- An output buffer can be turned off.
- You probably want to leave it turned on:
- DAC->CR &= ~ DAC_CR_BOFF1;

Why do we need a buffer?

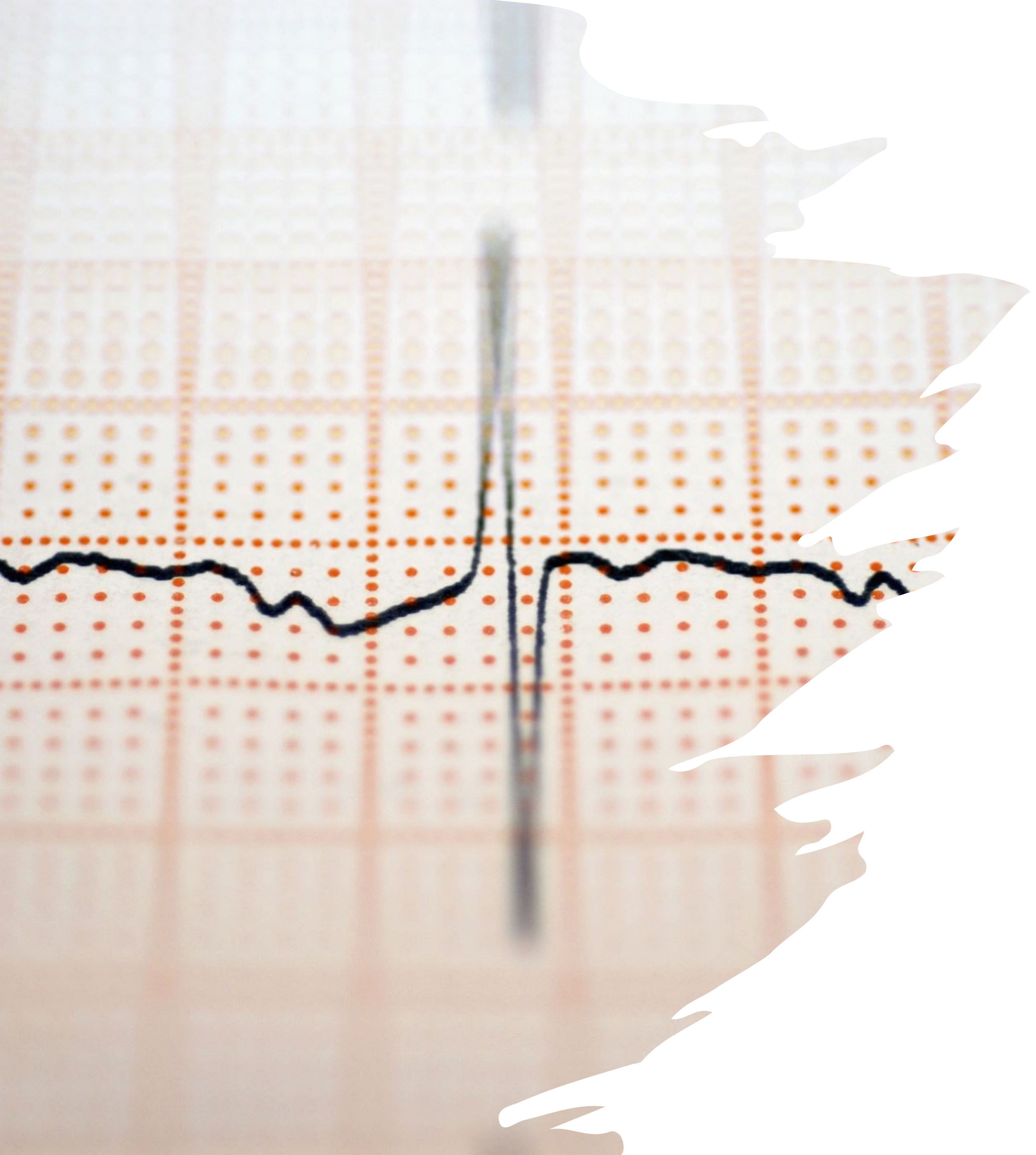
- Recall the hand-built DAC circuit:



- What's the impedance of Vout if R = 10KΩ?

Why would we turn buffer off?

- Maybe you want to use your own high-precision amplifier?
  - Remember that small curve on the low end of the scope view.

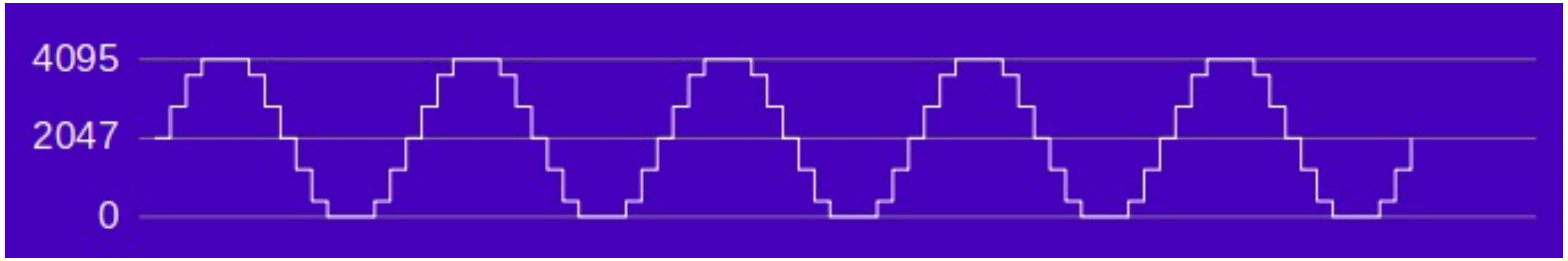


DAC uses

- We can use the DAC for slow things:
  - process control, object location manipulation by voltage, etc.
- The DAC can be updated very quickly.
  - More than 100k/sec.
  - This is useful for producing audio.
  - One example is a sine wave table.

What is a sine table?

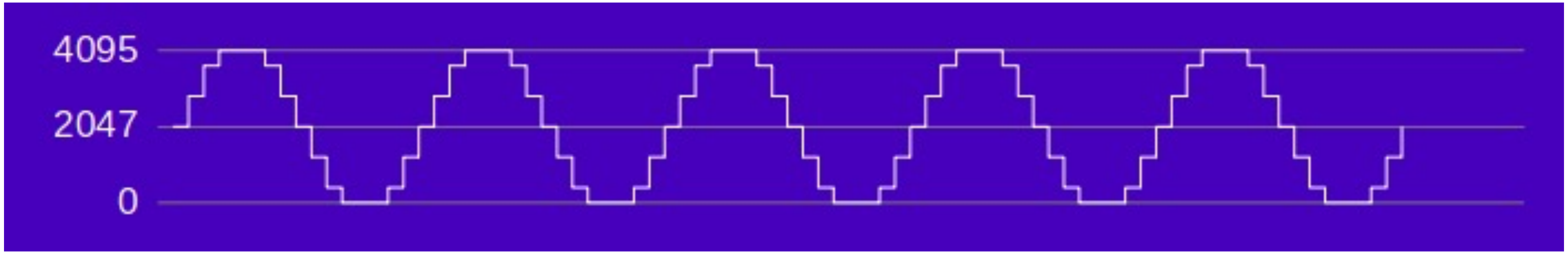
- We saw that writing monotonically-increasing values to the DAC produced a sawtooth wave.
- Imagine a set of 16 samples that look like this, that are sent repeatedly:



- BTW: For a large wave table, we make it "const". Why?

What is the frequency of the wave?

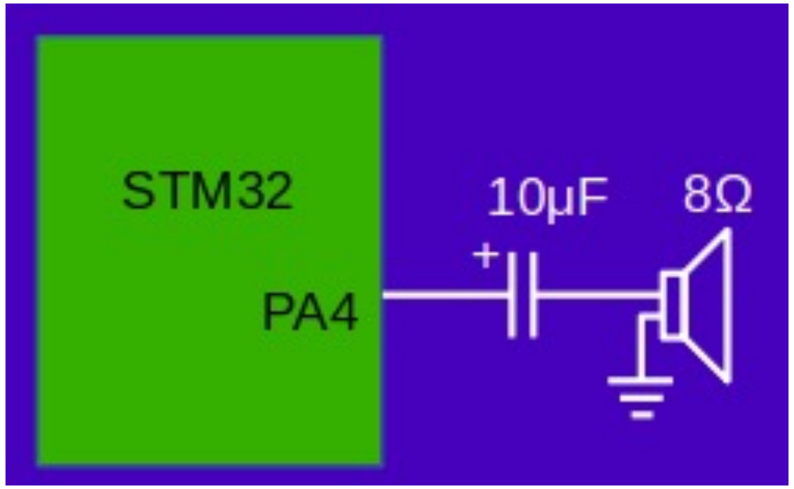
- We can see that this looks like a fuzzy sinusoid.
- How do we calculate the frequency?
- Sample/sec / Number of samples in a cycle
- For 64k samples/sec, 16 samples/cycle…



- 4 kHz

Demonstration 1: Raw output

- First circuit:

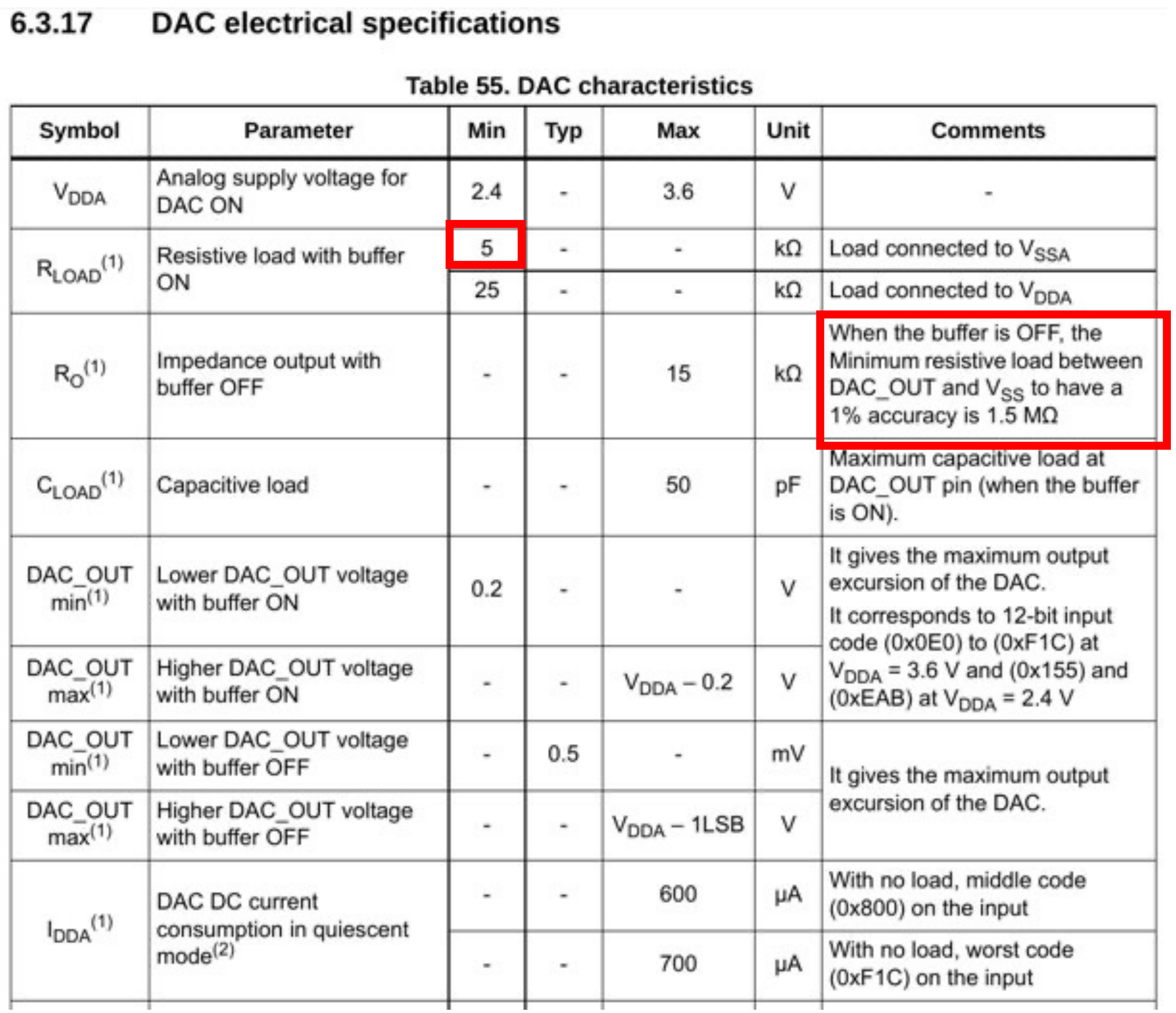


What's the large capacitor for?

Ensuring that a constant voltage on pin does not result in a constant current flowing through the speaker coil.

- First, try square waves with GPIO.
- Next, try DAC sawtooth, square wave.

DAC output: Very weak.



6.3.17 DAC electrical specifications

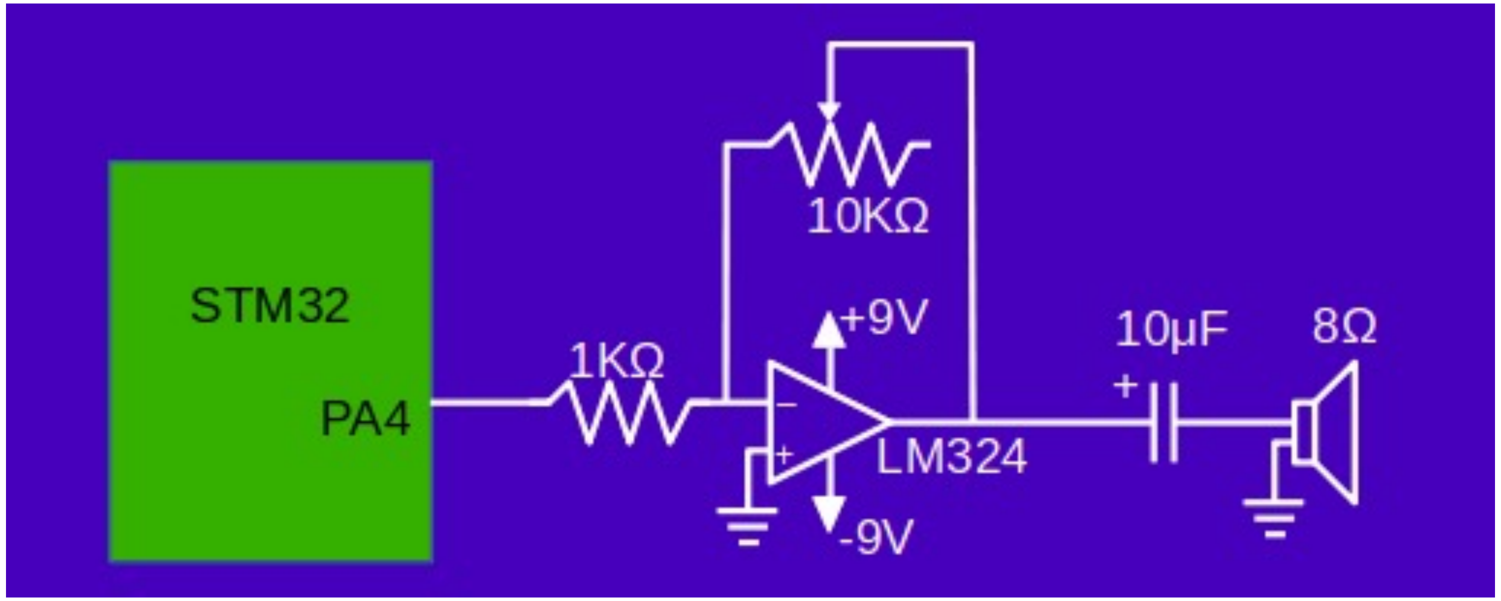
Table 55. DAC characteristics

Symbol	Parameter	Min	Typ	Max	Unit	Comments
V_{DDA}	Analog supply voltage for DAC ON	2.4	-	3.6	V	-
R_{LOAD}(1)	Resistive load with buffer ON	5	-	-	kΩ	Load connected to V_{SSA}
		25	-	-	kΩ	Load connected to V_{DDA}
R_O(1)	Impedance output with buffer OFF	-	-	15	kΩ	When the buffer is OFF, the Minimum resistive load between DAC_OUT and V_{SS} to have a 1% accuracy is 1.5 MΩ
C_{LOAD}(1)	Capacitive load	-	-	50	pF	Maximum capacitive load at DAC_OUT pin (when the buffer is ON).
DAC_OUT min(1)	Lower DAC_OUT voltage with buffer ON	0.2	-	-	V	It gives the maximum output excursion of the DAC. It corresponds to 12-bit input code (0x0E0) to (0xF1C) at V_{DDA} = 3.6 V and (0x155) and (0xEAB) at V_{DDA} = 2.4 V
DAC_OUT max(1)	Higher DAC_OUT voltage with buffer ON	-	-	V_{DDA} – 0.2	V	
DAC_OUT min(1)	Lower DAC_OUT voltage with buffer OFF	-	0.5	-	mV	It gives the maximum output excursion of the DAC.
DAC_OUT max(1)	Higher DAC_OUT voltage with buffer OFF	-	-	V_{DDA} – 1LSB	V	
I_{DDA}(1)	DAC DC current consumption in quiescent mode(2)	-	-	600	µA	With no load, middle code (0x800) on the input
		-	-	700	µA	With no load, worst code (0xF1C) on the input

- DAC output is just a resistor network.
- Need a lot of resistance on the output.
- Even with the buffer on, it's still wimpy.

Demonstration 2: Amplified output

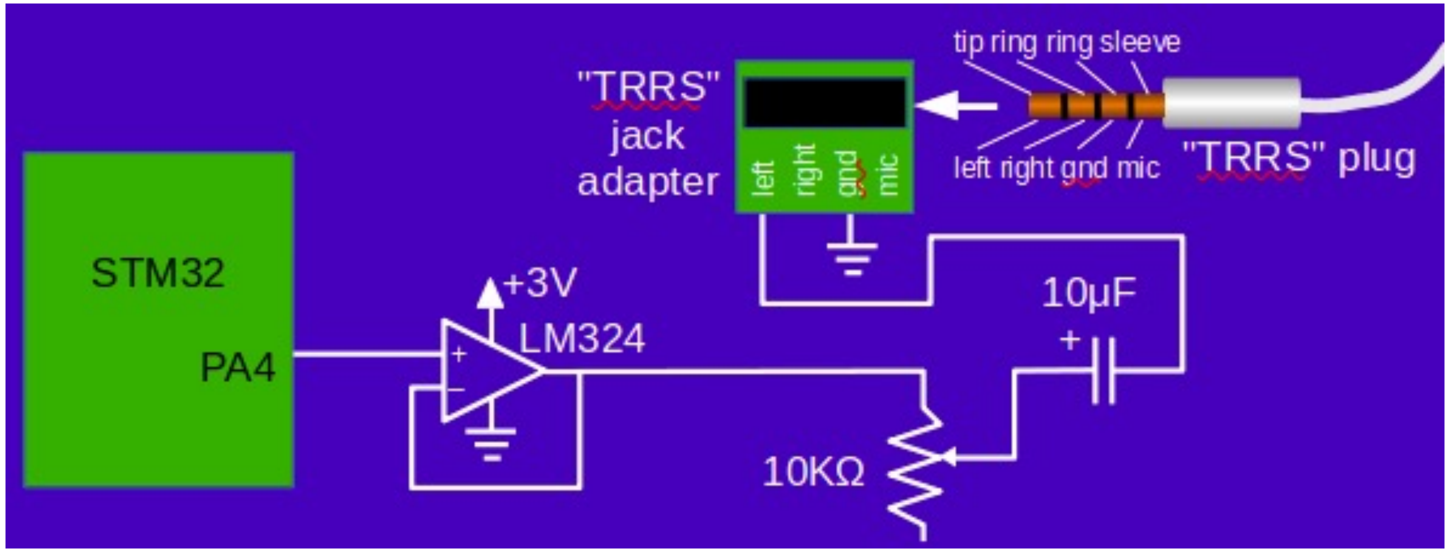
- Next circuit:



- DAC sawtooth again, square waves.
- Sine table.

Recommended Circuit

- A non-inverting op-amp:



- Avoids loading the DAC_OUT pin
- **<u>Protects the analog circuitry of the microcontroller</u>**

Protect the analog circuitry of the microcontroller!

- **CAUTION**: When you configure a pin on the STM32 for analog operation, you connect that pin to sensitive electronics inside the chip.
- If that pin is connected to voltage higher than 4V for just an instant, it **<u>will</u>** permanently, irreversibly damage either the pin or the entire microcontroller.