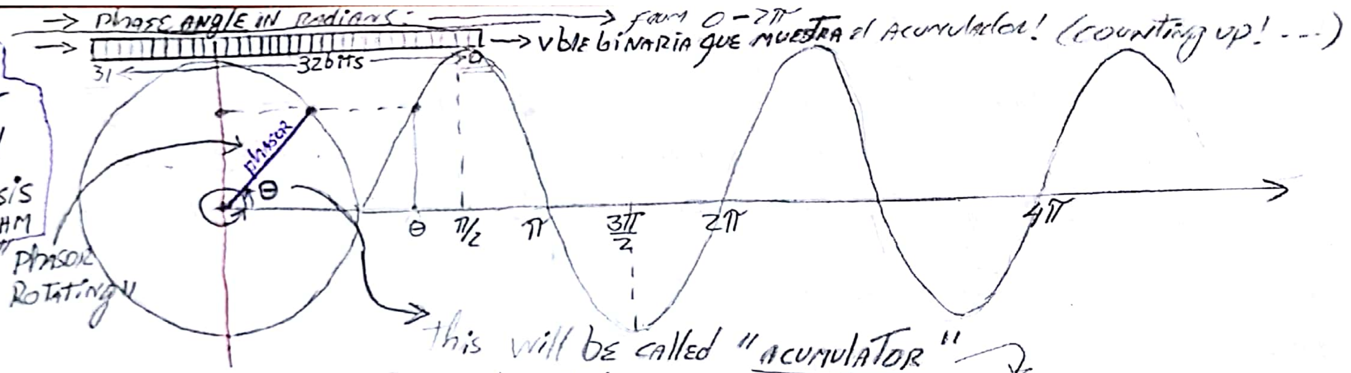
DDS
Direct Digital Synthesis ALGORITHM


→ Phase angle in radians: → from 0 − 2π
→ vble binaria que muestra el acumulador! (counting up! ...)
32 bits
31
0
phasor
θ
θ
π/2
π
3π/2
2π
4π
"Phasor Rotating"

this will be called "**ACUMULATOR**"
And will be represented as a 32-bit variable

- We can "scale" that angle such that it is at value 2^{32} (2 to the 32) when it completes "one rotation". And at value zero (0) when it starts.

$\Leftrightarrow$ Equivalent to say "Divide 2π radians (or 360°) into 2^{32} steps."

- Then when that phasor completes one rotation, the vble that we use to represent that angle will overflow

- That bin vble → is counting up with the phase angle at a different rate } b/c we have it scaled differently.
  - And when the phase ∡ overflows from $2\pi \xrightarrow{\text{back to}} 0$ radians, the bin vable overflows from $\underset{\text{"All ones"}}{2^{32}} \xrightarrow{\text{back to}} \underset{\text{"All zeros"}}{0}$

ESTAMOS Introduciendo una "nueva unidad" (unit) para ángulo y el SIZE of that unit of ∡ is chosen so that "there are exactly 2^{32} of those units in one rotation."

hay...	En una Rotación
2π (in radians)	✓
360° (en grados)	✓
2^{32} (units of acumulator) "vble"	✓

→ Consecuencia:
Rotar el phasor es sumar una cantidad a esa vble (acumulator)

But... How is this actually used?

Miremos el algoritmo y sus pasos y lo entenderemos...

Audiosynthesis ← requires very precise timming

- The faster you increment that "vble" the faster this phasor is interpreted to be rotating ⟹ And the consequence is → we get a higher frequency sine wave out of a very precise frequency. by going through the...

...following steps: (ALGORITHM!)

Enter ISR 1) Enter an interrupt service routine → to have some timer interrupt setup to enter an interrupt service routine at a very precise rate f_s. We're using 44 kilohertz

Inc Accum 2) When you get into the interrupt service routine, you add some amount to the accumulator (which is the same as...) ..(rotating the phasor) → a little bit to a slightly new angle

Then... READ Amplitude 3) You look up The Amplitud of the sine wave at that phasor angle (i.e. accumulator value)
In The example code → [we have that (sine Table)] which we use to go get sine amplitudes
How big does need to be? → A sine lookup Table → about 256 elements

SEND output 4) We then send that amplitude to the "DAC" → That's the voltage that we're going to output through the DAC

LEAVE ISR 5) Then... we "leave the interrupt service routine" and we are done with that signal sample

wait! 6) Then... one over $f_s \Rightarrow \left(\frac{1}{f_s}\right)$ later $\left(o\ sea\ \frac{1}{44{,}000}\ (sec)\right)$ seconds later (por Ej: $f_s = 44KHz$)
we "re-enter the interrupt and do it all again" ((1) to 6))

- the question now is...

↓

Ya que cuanto más agreguemos al "acumulador" (cada vez que entramos a la interrupción) más vamos a incremen-tar la velocidad de rotación del phasor y se incre-menta la frecuencia de la señal sinosoidal.

- For some desired output frecuency what's the correct amount to add to to the accumulator every time you enter the interrupt?
- And that's what the DDS tells you!
- it tells you precisely how much to add to that accumulator every time you interrupt to get an output frequency that you desire
- There is an equation that tells you what this increment value should be...

$\boxed{I}$ What if we increment the accumulator variable by 1 unit {every signal sample? [Every time we enter to The ISR] (*)[A one in the zeroth bit]

- Suppose that we are generating audio (signal) samples at Fs (Hz) [and that each time we generate a sample we increment the accumulator variable by 1(*). What will be the frequency of the resulting sine wave, Fout? (let's do a dimensional analisys...)

$$F_{out} = \underbrace{\frac{1\ \text{overflow (i.e. sine period)}}{2^{32}\ \text{Accumulator units}}}_{Ⓐ} \cdot \underbrace{\frac{1\ \text{Accumulator unit}}{1\ \text{signal sample}}}_{Ⓑ} \cdot \underbrace{\frac{F_s\ \text{signal samples}}{1\ \text{sec}}}_{Ⓒ} = \underbrace{\frac{F_s}{2^{32}}\ Hz}_{Ⓓ}$$

This is just a series of unit conversions

Ⓐ units of overflows

Ⓑ Adding one (1) every time we go to ISR

Ⓐ We'll have "one sine period" } Which is to say ∴ one overflow of the variable, every 2^{32} accumulator units Ⓔ

"And the phasor did a complete rotation"

Ⓑ We're considering the case where we're adding exactly <u>one</u> every time we enter the (audio) signal interrupt service routine, so this means we have one accumulator unit every signal sample

Ⓒ And then... we have some fixed signal rates, so we have Fs signal samples per second. [Ej: Fs could be 44KHz]

Which you set up with an ISR

Ⓓ So you can go through that equation and you end up with an output frequency of your signal sample rate over 2^{32} (the max value that can be contained in your 32bit accumulator vble

pero... si mismo ejemplo $\frac{44000}{2^{32}} = 0.00001024454832$ ← es una frecuencia muy pequeña!

y nosotros queremos frecuencias de 1Hz a 12MHz

→ y recordemos que estamos incrementando la vble accumulator en uno "1" cada vez que ingresamos a la ISR. ⟹ y por esto tenemos una tan baja frecuencia.

$\boxed{II}$ What if we increment the accumulator variable by 2 units? [Every time you enter the ISR]

- As before, supose we are generating signal samples at Fs(Hz).
- But now each time we generate a sample, we increment the accumulator vble by <u>2</u>

$$F_{out} = \frac{1 \text{ overflow (i.e. sine period)}}{2^{32} \text{ Accumulator units}} \cdot \underbrace{\frac{2 \text{ accumulator units}}{1 \text{ signal sample}}}_{\text{Now we're incrementing our accumulator by two units every signal sample}} \cdot \frac{F_s \text{ signal samples}}{1 \text{ sec}} = \left(\frac{F_s}{2^{32}} \cdot 2\right) Hz$$

... "And we end up with the same expression multiplied by 2 (Hz)"
- And at this point we can recognize "the pattern"
- If we increment the accumulator vble by some number (n) every time we entered the Interrupt Service Routine, we end with the same expression, where you just put n accumulator units per signal sample. And your output frec. is $\left\{ F_{out} = \frac{F_s}{2^{32}} \cdot n \, (Hz) \right.$

But F_{out} is a "known" in that expression. | What is really the unknown?

└→ Because is the desired frequency! (we are trying to synthesize)

F_s is also a "known"

└→ You set up the Timer interrupt that sends signal samples to the DAC

2^{32} is a "known"

└→ You instantiated the accumulator vble as an int (uint32_t)
so you know you can hold 2^{32} units in that binary vble

So... the "unknown" in that expression is the "increment value" $\underline{m}$

We re-arrange that expression...

$$m = \text{increment amount} = \frac{F_{out}}{F_s} \cdot 2^{32}$$

} For some desired Fout, we can solve to obtain the increment amount needed to produce that output frequency!

---

in music... [Middle C]

Ej: | Si quiero producir $F_{out} = 262\text{ Hz}$, y tengo una $F_s = 44\text{kHz}$

$$m = \frac{262\text{ Hz}}{44\times10^3\text{ Hz}} \cdot 2^{32} = 25,574,577.9 \approx 25,574,578$$

↓

The amount by which we need to increment our accumulator variable every time we enter the ISR (every signal interrupt!) in order to produce an output frequency (for a middle C tone) of 262 Hz

---

Next discussion is... What is the resolution (precision) of our signal generator (or synthesizer) using different sizes (of bit word) for our accumulator vble?
a) 32bits (int)

It is the same question as... How much frequency is contained in one unit of the accumulator? (32bit-accumulator and $F_s = 44\text{kHz}$)

a) with "int" → 32bits } $\text{Resolution} = \frac{F_s}{2^{32}} = \frac{44000}{2^{32}} = 1{,}02\times10^{-5}\text{ Hz}$ ⇒ and... We can target an Fout with an exaggerated precision!!!

Do we can just measure this with an oscilloscope? [This is a frequency of one cycle per month]

b) with "short" → 16bits } $\text{Resolution} = \frac{F_s}{2^{16}} = \frac{44.000}{2^{16}} = 0.67\text{ Hz}$ — pretty accurate

c) with "char" → 8bits } $\text{Resolution} = \frac{F_s}{2^{8}} = \frac{44.000}{2^{8}} = 171.8\text{ Hz}$ — "This is just terrible" resolution.

The recomendation is . . .
don't use more powerful aritmetic than you need or you just waste cpu

- "Floating point" aritmethic is 50 times slower than "fixed point, ("than integer)
- And we only have 900 cycles.
- So every { "Floating point" Multiply is 50 cycles off your budget / " Add is 60 cycles " / Divide is 150 cycles }
- So where you can use "fixed point"

900 cycles when.. { We have 44 kilohertz (Fs) / And we have 40 Megahertz CPU } That is... There is about 900 cpu cycles btw signal samples.

- So we have 900 cycles to spend on computing the next signal sample. before that deadline approaches

"

Now... How big does our sine table need to be?

It is 256 a good size for the sine lookup table?

or better... How many is enough?

In gral--- .The less entries we have in our sine lookup table The more harmonics you'l see in your generated signal

it depends on what are your requirements for error harmonics!
- How pure does your tone need to be!

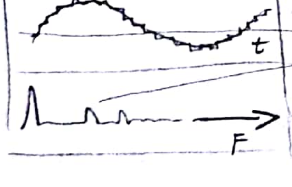
• In Audio... What does it matters is "really what sounds good" → with 256 entries → { For your ear it will sound perfectly pure

And the improvement (by adding) more than 256 entries to our sine table is "indiscernible" (To my ear)

$\underbrace{\hspace{6cm}}$

Suficiente para nuestra práctica o necesitaremos más?

The more entries in the table, the farther the 1st harmonic (error) moves away from the fundamental

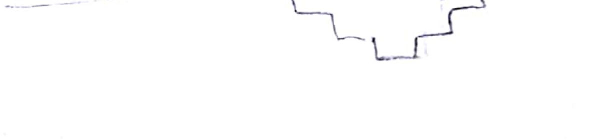


Analisis en f(t)

Análisis en f(F) } Cuando aparece el 1er armónico y qué tan atenuado está?

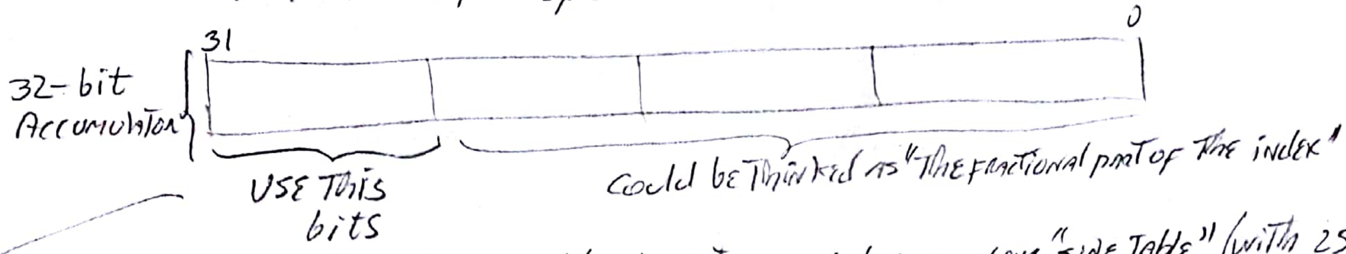
{ "Appears approx... at the # of entries on the table - 1"

El DAC → retiene el último nivel de voltaje enviado a él y solo lo cambia 1 vez que se le envía la próxima muestra =



How do we index into the sine Table?

If audio...
- Use The most significant 8 bits of your accumulator to index into The sine lookup Table. (you TRUNCATE out the whole bottom 24 bits)
- Remember The "shift" operation in The code!



→ With an 8-bit "char" you could index into any index of your "sine Table" (with 256 entries)

→ Just index into whatever These top bits Tells you To.

Ex → For low frequencies → maybe These top bits doesn't change for a few (audio) signal samples, (you may index into The same index of The sine Table a few samples in a row and that's ok, That's exactly what you want

A way to think → The phason will move to The next index at precisely The correct sample
→ Say for ex, That you are adding a Tenth of an index unit every Time you go into the ISR (with a sample). That means That for 10 interrupts in a row, these Top eight (8) integer parts of the index aren't going To change. (but The bottom 24 bit will continue increasing) But on The 11th (which is The correct one) you'll move to the next index of The sine Table because you will have accumulated enough fractional amounts of index To pop up by one //

Remember...
In gral we have more bits in our phase accumulator Than we require To index into our sine lookup Table, and we have that because we want to keep Track of fractional amounts of index, so That we can move to The next element of our sine Table at the correct sample //

H.W.
For audio signals we can have a solution for harmonics with a low pass filter btw your DAC output and the speaker. (with a cutoff frequency somewhere before The first harmonic - and somewhere after The fundamental. This cost a little bit extra hardware.)

USE H.H. when you can! → it's helpful! //

NOTA FINAL: Fs → es un ejemplo para muestreo en frecuencias de audio. pero ud puede elegir la que mejor convenga de acuerdo a la aplicacion y caracteristicas del MCU que se este trabajando!