HOW TO GENERATE A SIN WAVE WITH DSP

Nowadays, there exists, lot of complex methods to generate all kind of signals on a DSP, this paper shows you how to generate sinusoidal wave on a DSP without using look up tables (table filled of values, previously prepared to be sent to the DAC).

Lets start saying that this algorithm uses IIR filter for wave generating, Infinite Impulse Response (IIR) systems are part from wide theory, in this paper I'll explain nothing about IIR's, but following steps everybody will understand the way of sin wave generation using IIR's.

SOME DEFINITIONS:

F_S =Sample frequency (is the frequency at wich DAC will compose the signal).
Π =3.1416.
F_A =Frequency of the sinwave to be generated.

KEY EQUATIONS:

$$\Delta 1 = 2 \bullet COS\left(\frac{2 \bullet \Pi \bullet F_A}{F_S}\right) \quad (1)$$

$$\Delta 2 = SIN\left(\frac{2 \bullet \Pi \bullet F_A}{F_S}\right) \quad (2)$$

$$S_n(0)=0 \quad (3)$$

$$S_n(1) = \Delta 2 \quad (4)$$

$$S_n(n)=([\Delta 1 * S_n(n-1)]-[S_n(n-2)]) \quad (5)$$

HOW TO USE:

Suppose you need to generate a sinwave; first of all, you need to know the sample frequency, (you can obtain it from DAC Datasheets…).

Then decide what's the frequency for the sinwave to be generated (frequency should meet Nyquist criteria, so desired sin frequency should not be higher than half the sample frequency. This means that DAC, directly puts limit to the maximum signal frequency it can generate, as a rule of thumb, the faster DAC the higher sample frequency, the higher sinwave generated frequency, the most expensive DAC).

Next step, is to obtain $\Delta 1$ $\Delta 2$, , is easy to obtain, but be careful!!!, the trick is to operate in radians, as math lab does, so your calculator should be on "rad" mode when performing sin a cosine operations.

Once you have these two parameters, you'll need an array, for start running the algorithm, it's strongly recommended, to use complete cycles array, unless follow this rule, you will experiment phase changes during generation.

First of all, fill an array with samples using this algorithm, and then use this samples to feed ADC, restarting from the beginning of the array every time loop achieve last position (circular array).

Fill first position of the array with "0" value (see (3)) and second position of the array with "$\Delta 2$" value (see(4)); fter that, start filling the array from the third position, using formula (5).

Note that for filling the third position of the array, you'll use values on first and second positions, for the fourth value, you'll use values second an third, and so on…This philosophy is mathematically expressed on $S_n(n-1)$ and $S_n(n-2)$.

$S_n(n-1)$ means the value on the array just one position before the actual value being calculated $S_n(n)$.

$S_n(n-2)$ means the value on the array just before $S_n(n-1)$,or two positions before $S_n(n)$.

$S_n(n)$ means the value that is being calculated.

As you can see, for every step on array filling, you'll need the closer two previously calculated values, and in the first time, the two previous values needed are filled using (3) and (4) rules.

Of course this algorithm can be used on microcontrollers, not only on DSP, the only needed thing is the DAC!!!.

Hope you to find it useful…

POLOCERO

May 2006