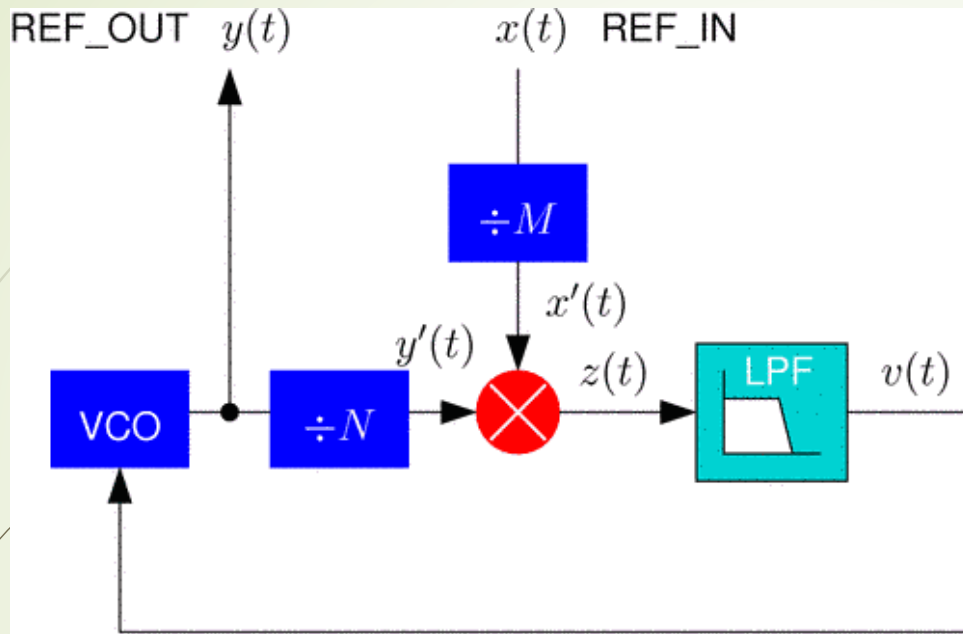




# PLL Implementation

Dr. Fangning HU



$$x(t) = \cos[2\pi ft + \phi(t)]$$

$$y(t) = \sin[2\pi ft + \hat{\phi}(t)]$$

$$z(t) = x(t)y(t)$$

$$z(t) = \frac{1}{2} \sin(\hat{\phi} - \phi) + \frac{1}{2} \sin(4\pi ft + \hat{\phi} + \phi)$$

↓ Low pass filter

$$v(t) \approx \frac{1}{2} \sin(\hat{\phi} - \phi) \approx \frac{1}{2}(\hat{\phi} - \phi)$$

$$y[n+1] = \sin(2\pi f(n+1) + \hat{\phi}[n+1])$$

$$y[n+1] = \sin(2\pi fn + \hat{\phi}[n] + 2\pi f - kv[n])$$

↓

$$y[n] = \sin(2\pi fn + \hat{\phi}[n])$$

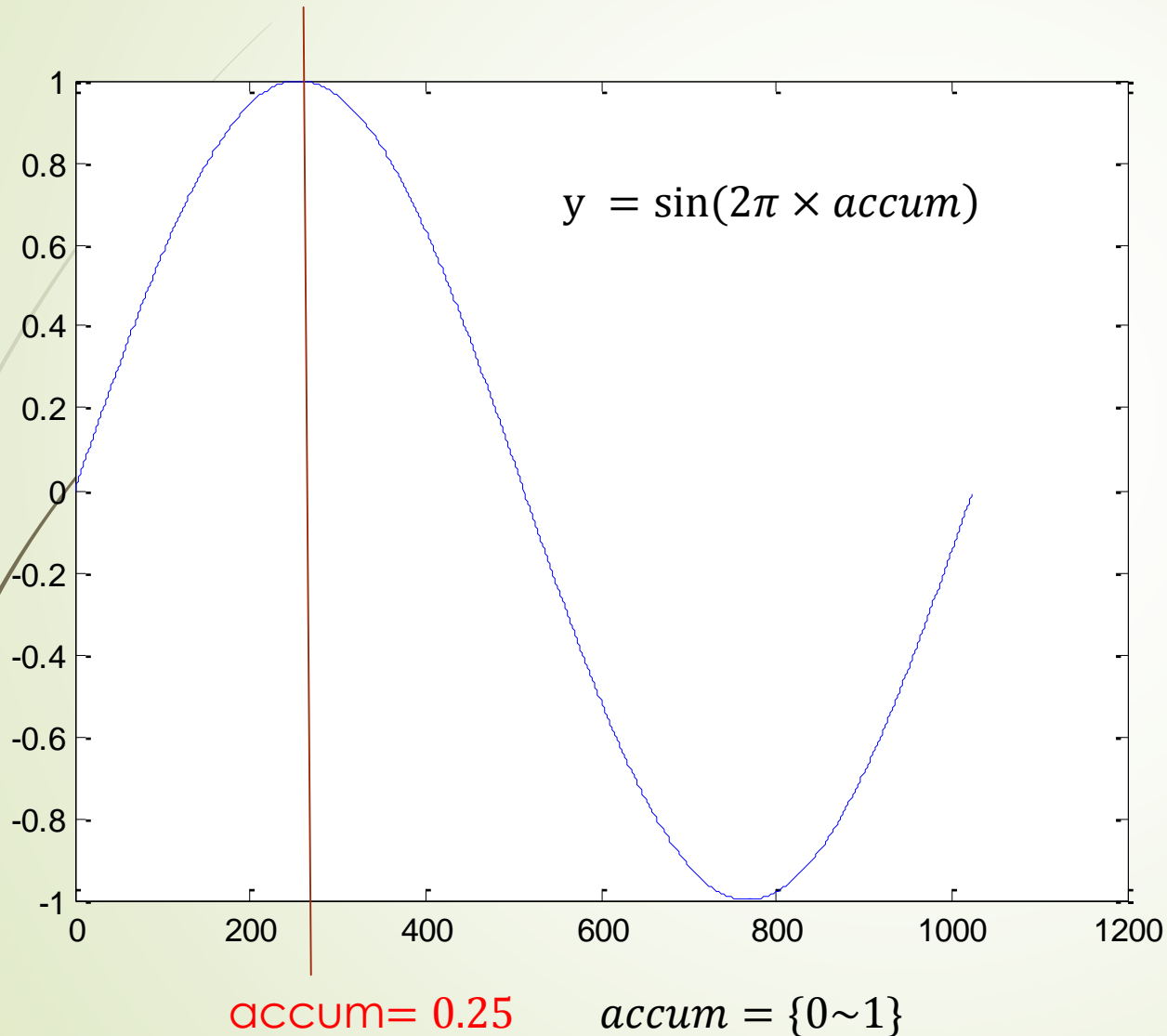
$$y[n] = \sin(2\pi(fn + \hat{\phi}[n]/2\pi)) = \sin(2\pi \times \text{accum}[n])$$

$$v[n] = \frac{1}{2}(\hat{\phi}[n] - \phi[n]),$$

$$\phi[n] = \hat{\phi}[n] - kv[n] \quad (k=2)$$

$$\hat{\phi}[n+1] = \phi[n] = \hat{\phi}[n] - kv[n]$$

# Phase update in discrete time



$$\phi[n+1] = \hat{\phi}[n] - kv[n]$$

$$y[n] = \sin(2\pi f n + \hat{\phi}[n]) = \sin(2\pi \times accum[n])$$

$$accum[n] = f n + \hat{\phi}[n]/2\pi$$

$$y[n+1] = \sin(2\pi f[n+1] \times \hat{\phi}[n+1])$$

$$= \sin(2\pi f n + 2\pi f + \hat{\phi}[n] - kv[n])$$

$$= \sin(2\pi f n + \hat{\phi}[n] + 2\pi f - kv[n])$$

$$= \sin(2\pi(f n + \hat{\phi}[n]/2\pi) + 2\pi f - kv[n])$$

$$= \sin(2\pi(accum[n]) + 2\pi f - kv[n])$$

$$= \sin(2\pi(accum[n] + f - kv[n]/2\pi))$$

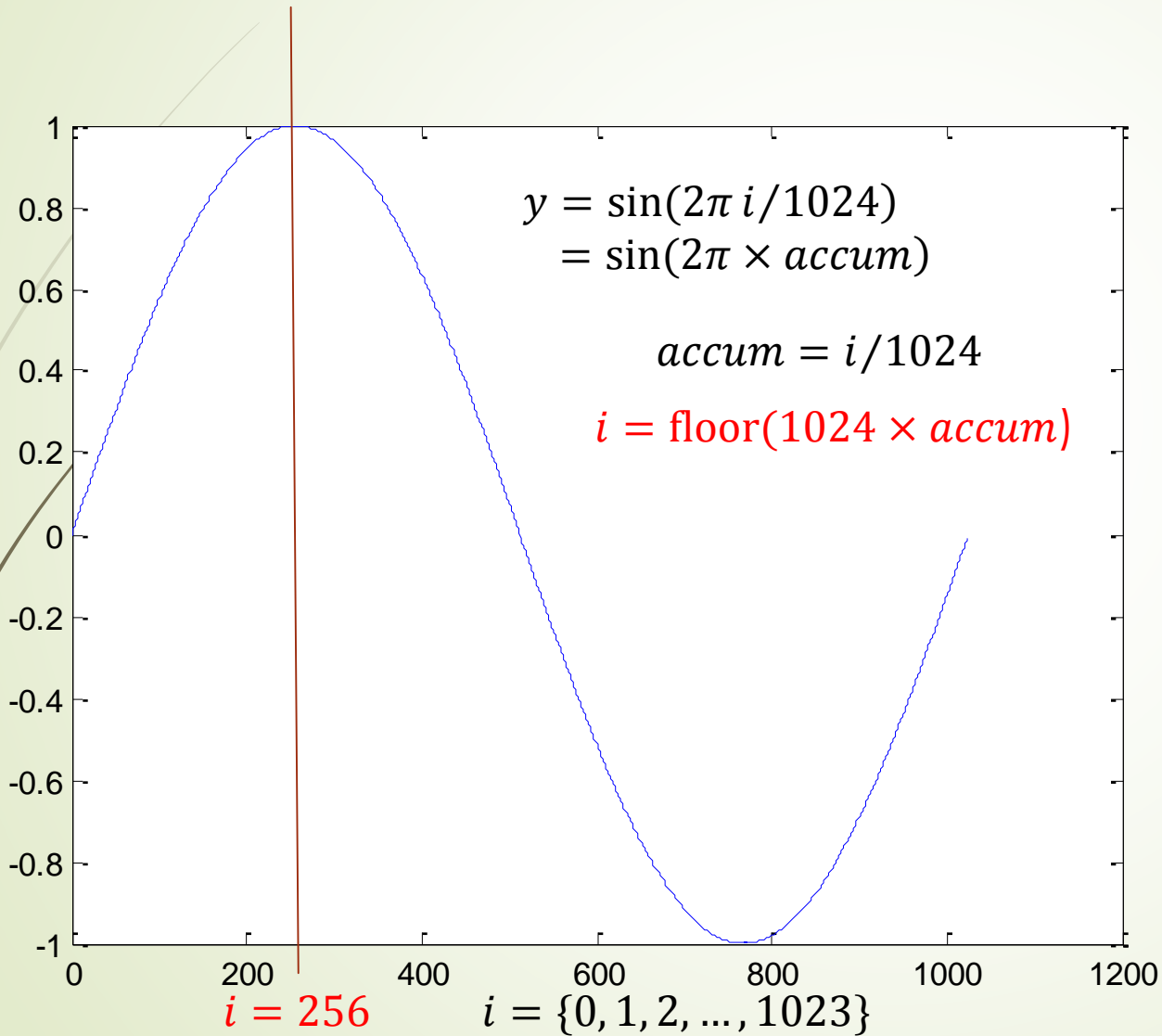
$$accum[n+1] = accum[n] + f - kv[n]/2\pi$$

$$accum[n+1] =$$

$$accum[n+1] - \text{floor}(accum[n+1])$$

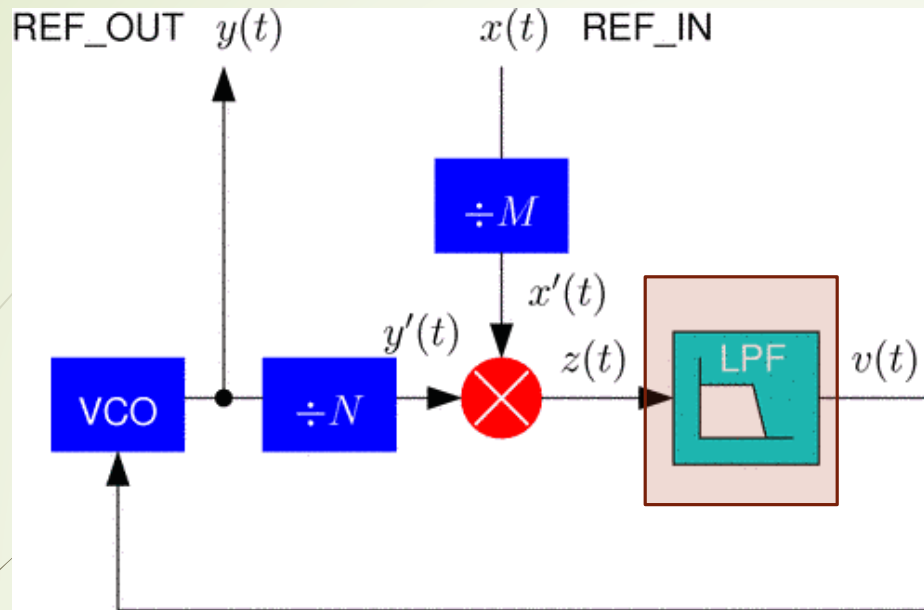
$$y[n+1] = \sin(2\pi \times accum[n+1])$$

# Sin Table (with size 1024)



accum	$i$	$y$
0	0	0
...	...	...
1/4	256	1
...	...	...
1023/1024	1023	0.xxxxxx

$$y[n + 1] = \sin(2\pi \times accum[n + 1]) \\ = \text{sintable}[i]$$



$$x(n) = \cos[2\pi fn + \phi(n)]$$

$$y(n) = \sin[2\pi fn + \hat{\phi}(n)]$$

$$z(n) = x(n)y(n)$$

Input:  $z[n], z[n-1], v[n-1]$     Output:  $v[n]$

$$v[n] = a_1 v[n-1] + b_0 z[n] + b_1 z[n-1]$$

$$a_1 = -\frac{T - 2\tau_1}{T + 2\tau_1}$$

$$b_0 = \frac{T + 2\tau_2}{T + 2\tau_1}$$

$$b_1 = \frac{T - 2\tau_2}{T + 2\tau_1}$$

$$\tau_1 = \frac{k}{\omega_0^2}$$

$$\tau_2 = \frac{2D}{\omega_0} - \frac{1}{k}$$

$D$  is the damping factor = 1 (critical),  
 $>1$  (slower response)

$\omega_0$  is the corner frequency ( $\leq 0.1 \times (2\pi f)$ )

$T$  is the sample period ( $=1$  in our discrete case)

In our experiment


$$k = 1$$

$$D = 1$$

$$f = 0.1$$


$$\omega_0 = (2\pi f) \times 0.1$$

$$T = 1$$



# Normalize the amplitude of input signal $x(n)$

- `amp = 0;`
- `for n = 1:number_samples,`
- `amp = amp + abs(x[n]);`
- `sample_scale = sample[sample_idx]/amp_est; end`
- `amp_est = amp/number_samples/(2/pi);`



**Task1:** Review the PLL implementation which you learned in the Communications lab From my slide or at webpage [http://dsp-fhu.user.jacobs-university.de/?page\\_id=229](http://dsp-fhu.user.jacobs-university.de/?page_id=229) (password: dsp2015) and describe each step in the lab report.

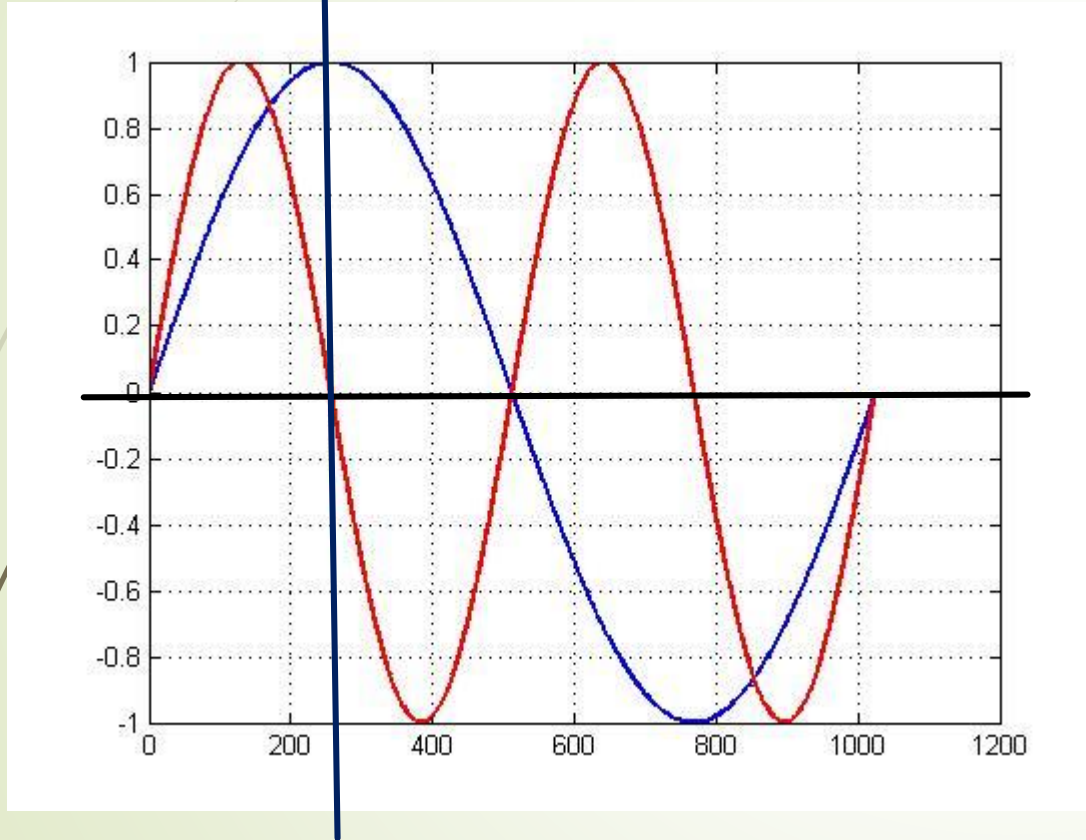
**Task2:** Write pll.c, pll.h and dsp\_ap.c. Like in Delay and FIR project, the input data need to be processed block by block. A struct variable need to be defined to store the state. The state information need to be passed from the previous block to the current block. In pll.c, you need to allocate memory and implement the pll algorithm. In dsp\_ap.c you need to initialize the necessary parameters and call the function pll.



# Generating double frequency

$$accum = accum + f - kv[n]/2\pi$$

$$y'[n+1] = \sin(2\pi \times accum) = \text{sintable}[i]$$



$$accum2 = accum2 + 2 * (f - kv[n]/2\pi)$$

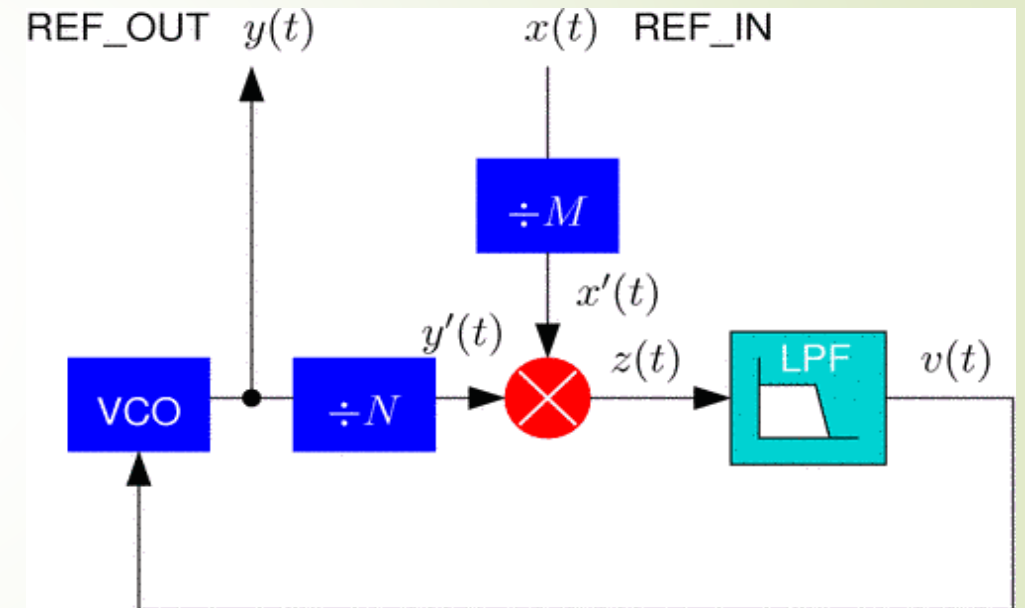
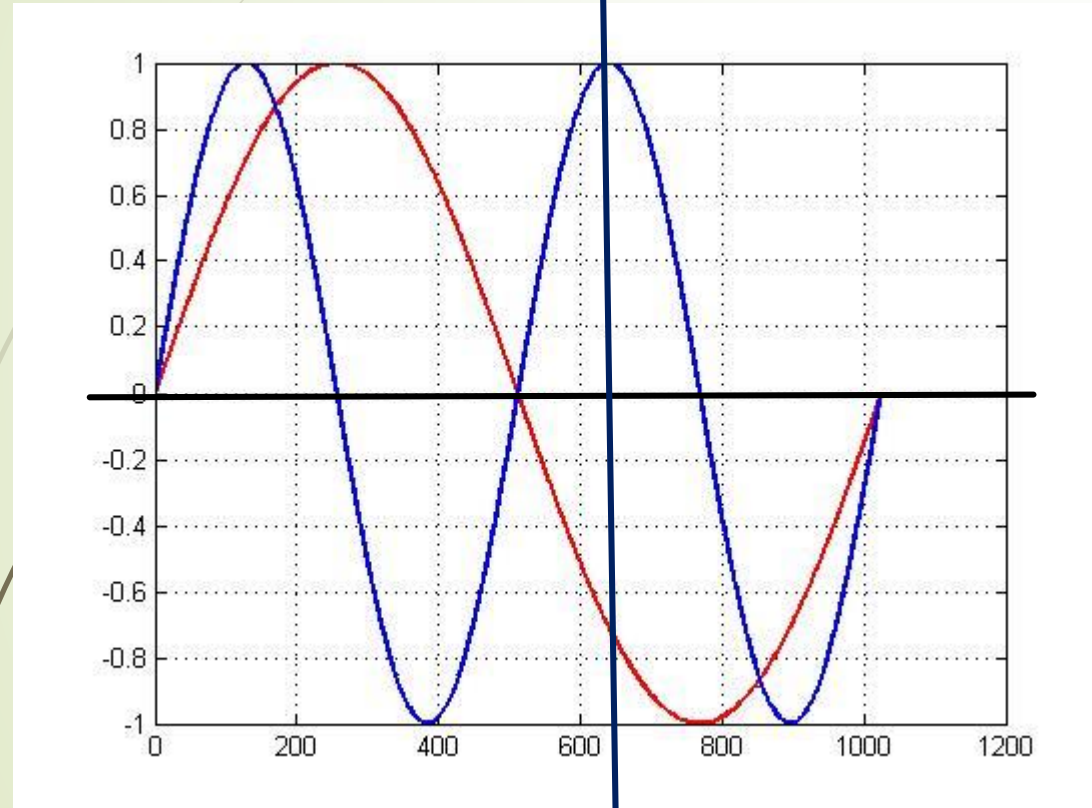
$$accum2 = accum2 + 2 * (f - kv[n]/2\pi)$$

$$y[n+1] = \sin(2\pi \times accum2) = \text{sintable}[i]$$



# Generating half frequencies

$$accum = accum + f - kv[n]/2\pi \quad y'[n+1] = \sin(2\pi \times accum) = \text{sintable}[i]$$



$$accum2 = accum2 + ?$$

$$y[n+1] = \sin(2\pi \times accum2) = \text{sintable}[i]$$



**Task3:** Modify your pll.c to generate double of the input frequency and half of the Input frequency.

**Grading scheme for lab report 3:** Task1 (20%), Task2 (40%), Task3(30%), Conclusion(10%)