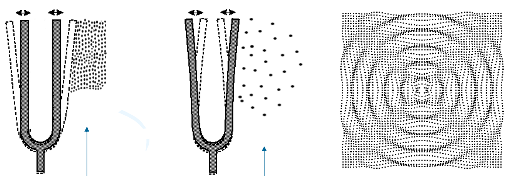
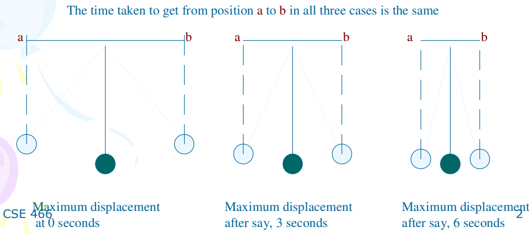
What is Sound?

As the tines move back and forth they exert pressure on the air around them.
(a) The first displacement of the tine compresses the air molecules causing high pressure.
(b) Equal displacement of the tine in the opposite direction forces the molecules to widely disperse themselves and so, causes low pressure.
(c) These rapid variations in pressure over time form a pattern which propogates itself through the air as a wave. Points of high and low pressure are sometimes reffered to as '**compression'** and '**rarefaction**' respectively.



Simple Harmonic Motion -- a Pendulum

- When a pendulum approaches equlibrium it doesn't slow down; it simply travels a smaller distance from the point of rest.
- Any body undergoing simple harmonic motion moves periodically with uniform speed.
- If the tuning fork is moving periodically then the pressure variations it creates will also be periodic.

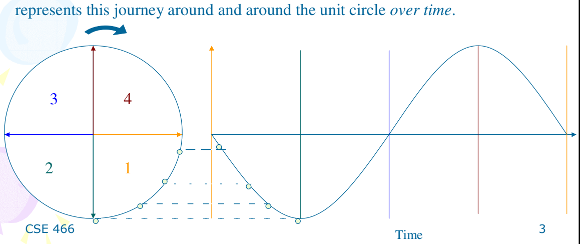


The Unit Circle

These pressure patterns can be represented using as a circle.
Imagine the journey of the pendulum or the tine in four stages:

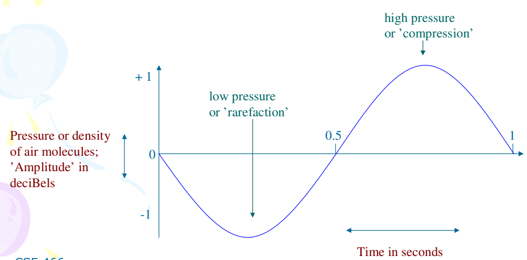
1) from its point of rest to its first point of maximum displacement...
2) its first point of maximum displacement back through the point of rest...
3) ... to its second point of maximum displacement...
4) ... and back from there through its point of rest again

We can map that journey to a circle. This is called the **Unit Circle**. The **sine wave** represents this journey around and around the unit circle *over time*.



Sine Waves

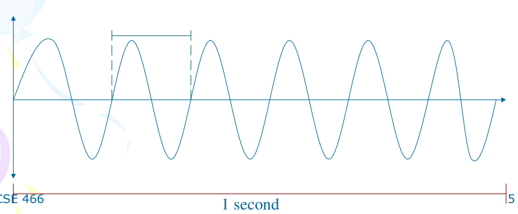
The **sine wave** or **sinusoid** or **sinusoidal signal** is probably the most commonly used graphic representation of sound waves.



Sine Waves

The specific properties of a sine wave are described as follows.

Frequency = the number of cycles per second (this wave has a frequency of 6 hertz)
Amplitude = variations in air pressure (measured in decibels)
Wavelength = physical length of 1 period of a wave (measured in metres per second)
Phase = The starting point of a wave along the y-axis (measured in degrees)



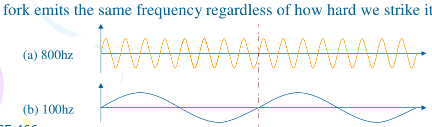
Frequency

Frequency refers to the number of cycles of a wave per second. This is measured in **Hertz**. So if a sinusoid has a frequency of 100hz then one period of that wave repeats itself every 1/100th of a second. Humans can hear frequencies between 20hz and 20,000hz (20Khz).

1) Frequency is closely related to, *but not the same as!!!*, pitch.

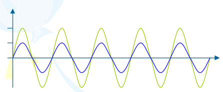
2) Frequency does not determine the speed a wave travels at. Sound waves travel at approximately 340metres/second regardless of frequency.

3) Frequency is inherent to, and determined by the vibrating body – not the amount of energy used to set that body vibrating. For example, the tuning fork emits the same frequency regardless of how hard we strike it.

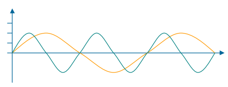


Amplitude

- Amplitude describes the size of the pressure variations.
- Amplitude is measured along the vertical y-axis.
- Amplitude is closely related to *but not the same as!!!*, loudness.



(a) Two signals of equal frequency and varying amplitude



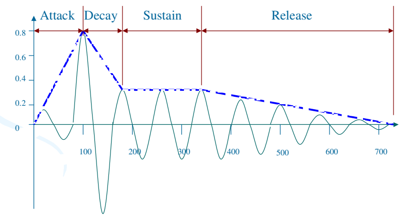
(b) Two signals of varying frequency and equal amplitude

Amplitude Envelope

The amplitude of a wave changes or 'decays' over time as it loses energy.

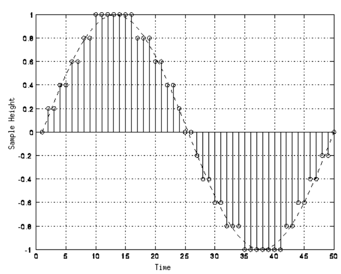
These changes are normally broken down into four stages; **Attack** , **Decay**, **Sustain** and **Release**.

Collectively, the four stages are described as the **amplitude envelope**.



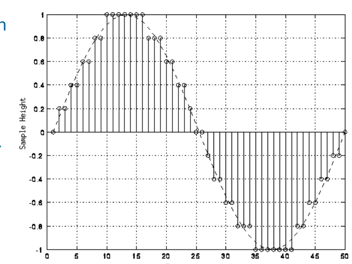
Quantization

- The digital signal is defined only at the points at which it is sampled.



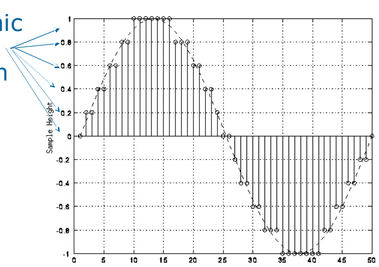
Quantization

- The height of each vertical bar can take on only certain values, shown by horizontal dashed lines, which are sometimes higher and sometimes lower than the original signal, indicated by the dashed curve.



Quantization

- If the graphic has 11 quantization levels, how many bits are needed to encode each sample?



Quantization

- The difference between a quantized representation and an original analog signal is called the *quantization noise*.
- The more bits for quantization of a signal, the more closely the original signal is reproduced.

Quantization

- Using higher sampling frequency and more bits for quantization will produce better quality digital audio.
- But for the same length of audio, the file size will be much larger than the low quality signal.

CSE 466 13

Quantization

- The number of bits available to describe sampling values determines the resolution or accuracy of quantization.
- For example, if you have 8-bit analog to digital converters, the varying analog voltage must be quantized to 1 of 256 discrete values;
- a 16-bit converter has 65,536 values.

CSE 466 14

Nyquist Theorem

- A theorem which states that an analog signal waveform may be uniquely reconstructed, without error, from samples taken at equal time intervals.

CSE 466 15

Nyquist Theorem

- The sampling rate must be equal to, or greater than, twice the highest frequency component in the analog signal.

CSE 466 16

Nyquist Theorem

- Stated differently:
- The highest frequency which can be accurately represented is one-half of the sampling rate.

CSE 466 17

Error

- Sampling an analog signal can introduce ERROR.
- ERROR is the difference between a computed, estimated, or measured value and the true, specified, or theoretically correct value.

CSE 466 18

Nyquist Theorem

- By sampling at TWICE the highest frequency:
  - One number can describe the positive transition, and…
  - One number can describe the negative transition of a single cycle.

Nyquist Error-- aliasing

upper => sampling 6 times per cycle(fs=6f);
middle => sampling 3 times per cycle(fs=3f);

lower=> sampling 6 times in 5 cycles, from[1]

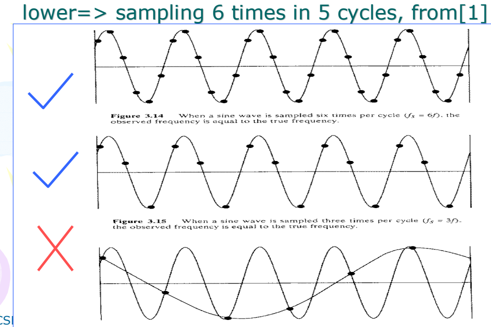


Figure 3.14 When a sine wave is sampled six times per cycle ($f_S = 6f$), the observed frequency is equal to the true frequency.

Figure 3.15 When a sine wave is sampled three times per cycle ($f_S = 3f$), the observed frequency is equal to the true frequency.

Figure 3.16 When a sine wave is sampled six times in five cycles, the observed sine wave has a much lower frequency than the original sine wave.

Analog Synthesis Overview

- Sound is created by controlling electrical current within synthesizer, and amplifying result.
- Basic components:
  - Oscillators
  - Filters
  - Envelope generators
  - Noise generators
- Voltage control

Oscillators

- Creates periodic fluctuations in current, usually with selectable waveform.
- Different waveforms have different harmonic content, or *frequency spectra*.

Filters

- Given an input signal, attenuate or boost a frequency range to produce an output signal
- Basic Types:
  - Low pass
  - High pass
  - Band pass
  - Band reject (notch)

Envelope Generators

- Generate a control function that can be applied to various synthesis parameters, including amplitude, pitch, and filter controls.

Noise Generators

- Generate a random, or semi-random fluctuation in current that produces a signal with all frequencies present.

Digital Synthesis Overview

- Sound is created by manipulating numbers, converting those numbers to an electrical current, and amplifying result.
- Numerical manipulations are the same whether they are done with software or hardware.
- Same capabilities (components) as analog synthesis, plus significant new abilities

Digital Oscillators

- Everything is a Table
  - A table is an indexed list of elements (or values)
  - The index is the address used to find a value

Generate a Sine Tone Digitally (1)

- Compute the sine in real time, every time it is needed.
  - equation:

$$signal(t) = r\sin(\omega t)$$

  - t = a point in time; r = the radius, or amplitude of the signal; w (omega) = 2pi*f the frequency
  - Advantages: It's the perfect sine tone. Every value that you need will be the exact value from the unit circle.
  - Disadvantages: must generate every sample of every oscillator present in a synthesis patch from an algorithm. This is very expensive computationally, and most of the calculation is redundant.

Generate a Sine Tone Digitally (2)

- Compute the sine tone once, store it in a table, and have all oscillators look in the table for needed values.
  - Advantages: Much more efficient, hence faster, for the computer. You are not, literally, re-inventing the wheel every time.
  - Disadvantages: Table values are discrete points in time. Most times you will need a value that falls somewhere in between two already computed values.

Table Lookup Synthesis

- Sound waves are very repetitive.
- For an oscillator, compute and store one cycle (period) of a waveform.
- Read through the wavetable repeatedly to generate a periodic sound.

Changing Frequency

- The Sample Rate doesn't change within a synthesis algorithm.
- You can change the speed that the table is scanned by skipping samples.
- skip size is the increment, better known as the phase increment.

phase increment is a very important concept

Algorithm for a Digital Oscillator

- Basic, two-step program:
  - $phase\_index = \mathrm{mod}_L(previous\_phase + increment)$
  - $output = amplitude \times wavetable[phase\_index]$
- $increment = \dfrac{TableLength \times DesiredFrequency}{SampleRate}$

If You're Wrong, it's Noise

- What happens when the phase increment doesn't land exactly at an index location in the table?
  - It simply looks at the last index location passed for a value.
    In other words, the phase increment is truncated to the integer.
- Quantization
- Noise
- The greater the error, the more the noise.

Interpolation

- Rather than truncate the phase location…
  - look at the values stored before and after the calculated phase location
  - calculate what the value would have been at the calculated phase location if it had been generated and stored.

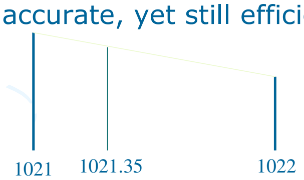
  Interpolate
- More calculations, but a much cleaner signal.

Linear interpolation

- Interpolate between two audio samples

```
double inbetween = fmod(sample, 1);
return (1. – inbetween) * wave[int(sample)] +
        inbetween * wave[int(sample) + 1];
```

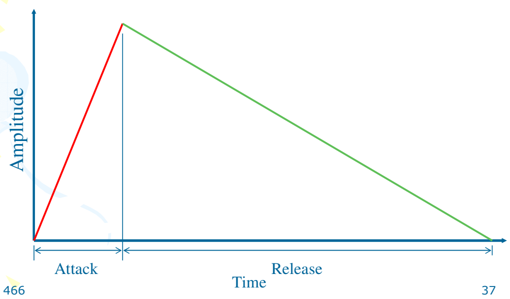
- More accurate, yet still efficient



Envelopes

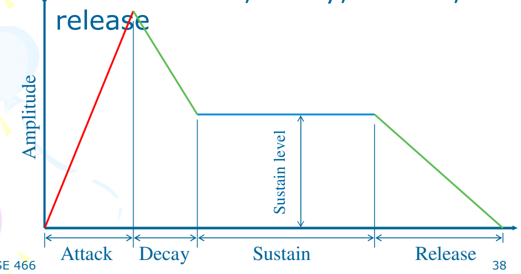
- We commonly will make samples with fixed amplitudes, then make a synthetic *envelope* for the sound event.

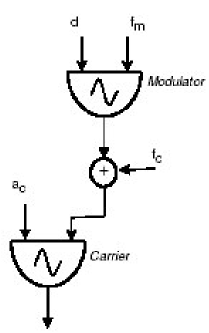
Attack and Release



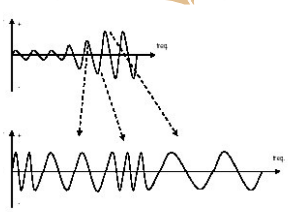
ADSR

- ADSR: Attack, decay, sustain, release

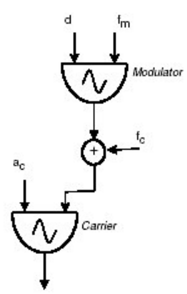




Frequency Modulation



FM: General Description



Simple FM: carrier oscillator has its frequency modulated by the output of a modulating oscillator.

Sidebands produced around carrier at multiples of modulating frequency.

- Number generated depends on the amplitude of the modulator.

Modulator : Carrier Ratio

- Sidebands at C + and - (n * Modulator)
- Ratio of M:C determines whether spectrum is harmonic or not.
  - Simple integer ratio = harmonic
  - Non-integer ratio = inharmonic

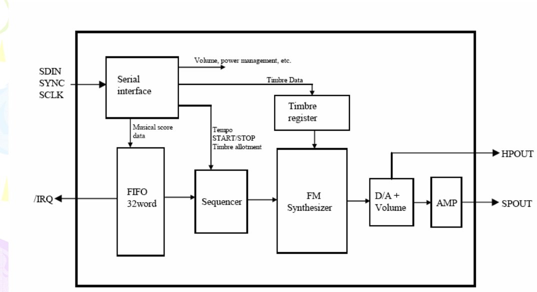
Modulation Index and Bandwidth

- The *bandwidth* of the FM spectrum is the number of sidebands present.
- The bandwidth is determined by the *Modulation Index*
  - *I = depth of modulation / modulator*
  - *D* depth of modulation, which depends on the amount of amplitude applied to modulating oscillator. (*D = A x M*)
- If the index is above zero, then sidebands occur.

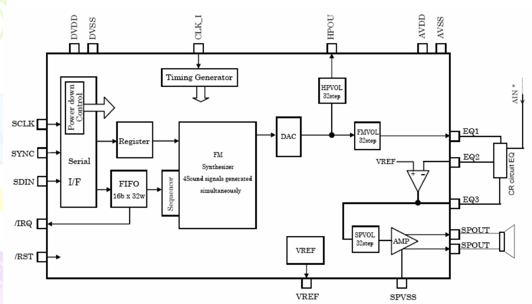
Yamaha YMU757B

- Original FM synthesis function
- Discovered at Stanford in 70's
- Patented and licensed to Yamaha
- Used in famous DX-7 keyboard
  (and many other products)

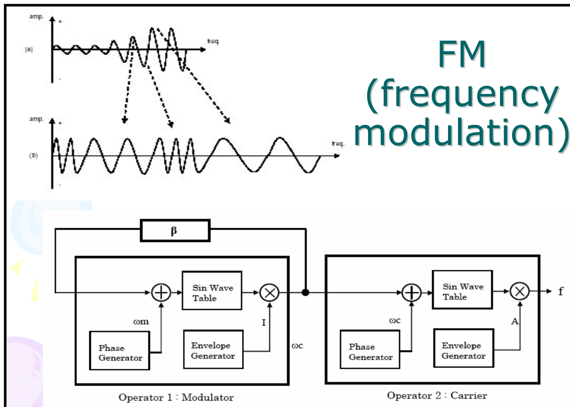
Yamaha YMU757B internals



Yamaha YMU757B



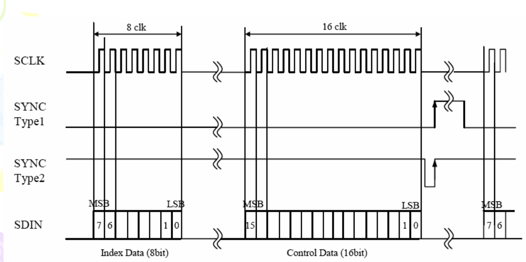
FM (frequency modulation)

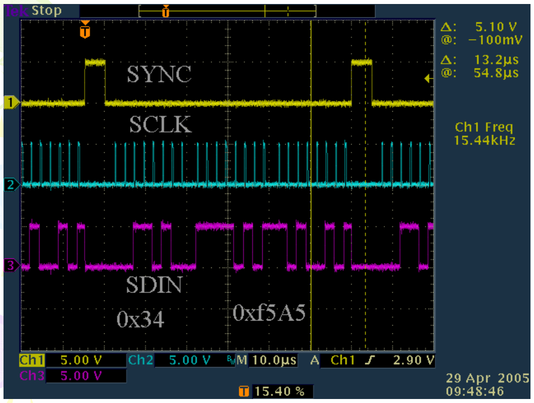


Internal Register Set

Index	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	Description
$00h	BL1	BL0	NT3	NT2	NT1	NT0	CH1	CH0	VIB	TI3	TI2	TI1	TI0	TK2	TK1	TK0	Note data
	0	0	1	1	0	0	CH1	CH0	VCHE	TI3	TI2	TI1	TI0	VCH2	VCH1	VCH0	Rest data
$10 - 2Fh	ML2	ML1	ML0	VIB	EGT	SUS	RR3	RR2	RR1	RR0	DR3	DR2	DR1	DR0	AR3	AR2	Timbre data (Left data
	AR1	AR0	SL3	SL2	SL1	SL0	TL5	TL4	TL3	TL2	TL1	TL0	WAV	FL2	FL1	FL0	for 1 timbre)
$30h	0	V32	V31	V30	0	V22	V21	V20	0	V12	V11	V10	0	V02	V01	V00	Timbre allotment data
$31h	0	0	0	0	0	0	0	0	T7	T6	T5	T4	T3	T2	T1	T0	Tempo data
$32h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLR	ST	FM Control
$33h	0	0	0	0	0	0	0	0	0	0	0	0	0	CLKSEL			CLK_I select
$34h	0	0	0	0	0	0	0	0	0	0	IRQE	IRQ Point					IRQ Control
$35h	0	0	0	0	0	0	0	0	0	0	0	V4	V3	V2	V1	V0	Speaker Volume
$36h	0	0	0	0	0	0	0	0	0	0	0	V4	V3	V2	V1	V0	FM Volume
$37h	0	0	0	0	0	0	0	0	0	0	0	V4	V3	V2	V1	V0	HPOUT Volume
$38h	0	0	0	0	0	0	0	0	0	0	0	AP4	AP3	AP2	AP1	DP	Power Management
$39h	0	0	0	0	0	0	0	CLKSET									CLK_I Select
$40 - EFh	Reserved (access prohibited)																Reserved
$F0 - FFh	For LSI TEST(access prohibited)																LSI TEST

Interface





Settings and procedure to generate melody

Follow the steps as described below/

1. Set the CLKSEL ($33h) according to the clock frequency inputted for CLK_I.
2. Cancel the power-down mode of the analog section. (Refer to "Resetting sequence of analog section".)
3. Set the timbre data ($10-2Fh), timbre allotment data ($30h), tempo data ($31h) and volumes ($35-37h) as desired.
4. Enter 32 musical score data ($00h) until FIFO is full.
5. Set the IRQ point value of $34h. (Default at the center of FIFO).
6. Set "1" for IRQE of $34h
7. Set "1" for the ST bit of $32h and start the melody.

What we're going to do:

- This week:
  - Talk to the FM chip
  - Explore some basic FM sounds
  - Port some birdsongs to our board
- In two weeks:
  - Port sound code to TinyOS and mote
  - Explore some Evolutionary Computation
  - Do the Flock