Lecture 2 - Number Representations, Fundamental DSP Concepts

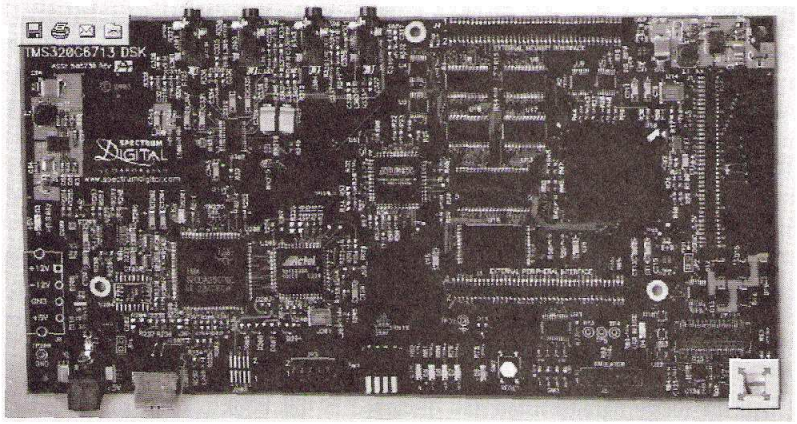
James Barnes (James.Barnes@colostate.edu)

Spring 2014

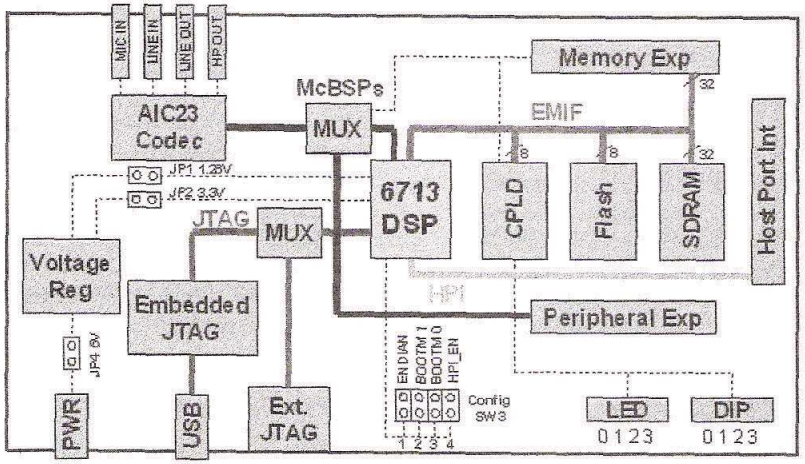
Number Representations

DSK Hardware

4 DSP Development System



(*a*)



(*b*)

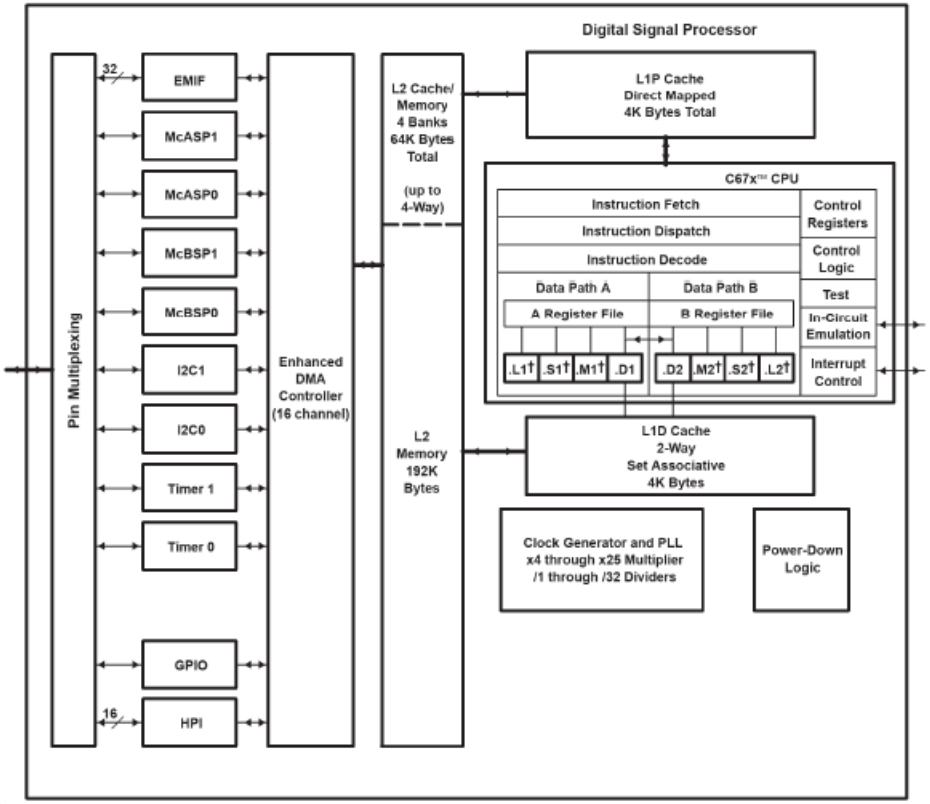
FIGURE 1.1. TMS320C6713-based DSK board: (*a*) board; (*b*) diagram. (Courtesy of Texas Instruments)

C6713 DSP Chip

TMS320C6713B
FLOATING-POINT DIGITAL SIGNAL PROCESSOR

SPRS294B – OCTOBER 2005 – REVISED JUNE 2006

functional block and CPU (DSP core) diagram



† In addition to fixed-point instructions, these functional units execute floating-point instructions.

EMIF interfaces to:
–SDRAM
–SBSRAM
–SRAM,
–ROM/Flash, and
–I/O devices

McBSPs interface to:
–SPI Control Port
–High-Speed TDM Codecs
–AC97 Codecs
–Serial EEPROM

McASPs interface to:
–I2S Multichannel ADC, DAC, Codec, DIR
–DIT: Multiple Outputs

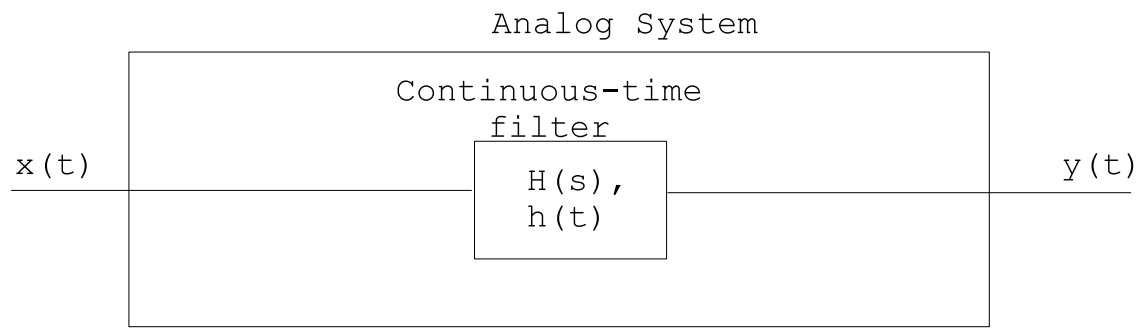
A C Program to Generate a Sine Wave

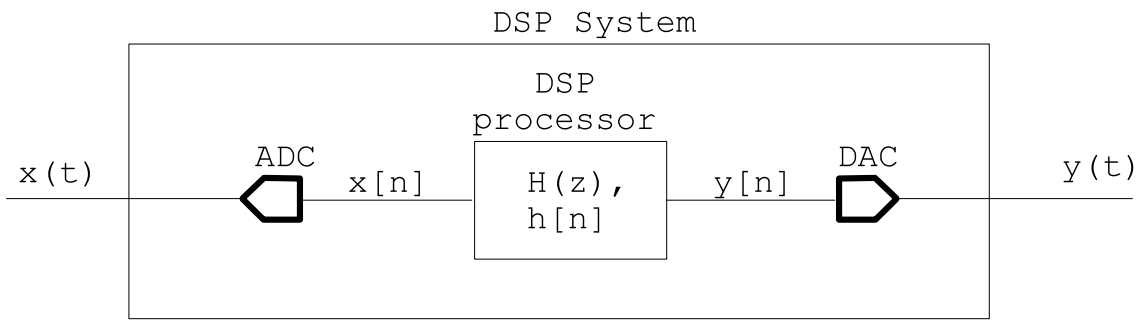
```
/* This project uses support files generated by Rulph Chassaing
   Comm routines included in C6xdskinit.c */
#include "dsk6713_aic23.h"          // needed to access codec functions
#include <math.h>                   // needed for sin(*/ function
#define PI 3.14159265359            // define the constant PI
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  // sampling frequency of codec
float f0=1000;                      // generated sinusoid frequency
float fs_float=8000;                // needed for calculating offset
float angle=0;                      // sin() argument in radians
float offset;                       // sin() argument change per sample period
short amp=20;                       // sine amplitude scaling factor
short sine_value;                   // value sent to codec

interrupt void c_int11()            // interrupt service routine
{
  offset=2*PI*f0/fs_float;           // set offset value
  angle = angle + offset;            // previous angle plus offset
  if (angle > 2*PI)                  // reset angle if > 2*PI
    angle -= 2*PI;                   // angle = angle - 2*PI
  sine_value=(short)1000*amp*sin(angle);  // calculate current output sample
  output_left_sample(sine_value);    // output each sine value
  return;                            // return from interrupt
}

void main()
{
  comm_intr();                        // init DSK, codec, SP0 for interrupts
  while(1);                           // wait for an interrupt to occur
}
```

The DSP Idea - Sampling, Digital Processing, and Reconstruction





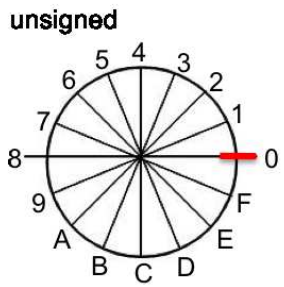
Intro to C Language Data Types

Ref: (Tretter, on E-reserve)

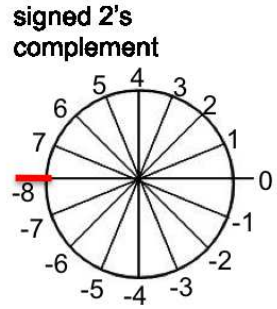
- `short` - 16b signed (default) or unsigned (unsigned short). Most useful for the DSK because we have a 16b ADC/DAC (codec)
- `int, uint, uint32` - 32b integer (signed or unsigned).
- `float` and `double` - 32b and 64b IEEE floating point signed number type.
  - `float` gives 6-8 decimal places of precision, `double` 15-17 dec places.
- Integers (`short,int`) used in fixed-point arithmetic.
  - **2's complement** representation for signed numbers
  - There is no explicit radix point in the number representation – user must keep track. Commonly, the radix point is imagined to lie to the left of the MSB for unsigned integer, to the right of the sign bit for signed.
  - Arithmetic operations (+, X) can move the radix point.

Fixed Point "Hazards" : Overflow

Overflow behavior is different for unsigned and signed numbers. Consider 4 bit unsigned and signed numbers (not a standard C data type):



- Can't do subtraction with both operands unsigned.
- For addition, wraparound occurs when carryout occurs. Strictly speaking, not "overflow" but gives unexpected results when carryout not considered.



- Can only get overflow when both ops have same sign.
- Overflow occurs when result has different sign from ops.
- Carryout does not matter

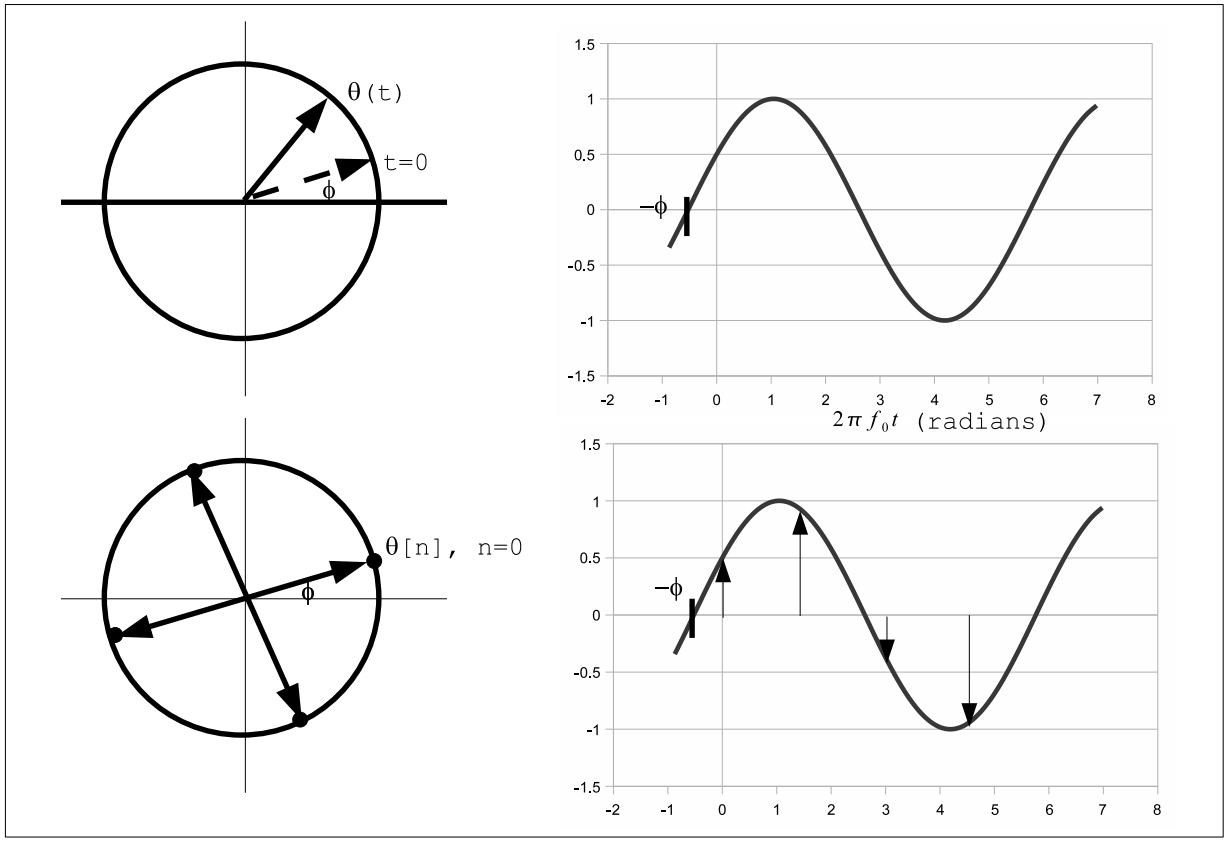
Test for overflow (works for both signed and unsigned numbers): overflow has occurred

- a and b are positive and the sum $(a + b)$ is negative, or
- a and b are negative and the sum $(a + b)$ is positive.

Representing Analog Signals with Numbers

Phasor Representation of Sine Function

- Continuous Time Function : $y(t) = A_o sin(\theta(t)) = A_o sin(2\pi f_0 t + \phi)$
- Discrete Time Sequence: $y[n] = A_o sin(\theta[n]) = A_o sin(2\pi \frac{f_0}{F_s} n + \phi)$

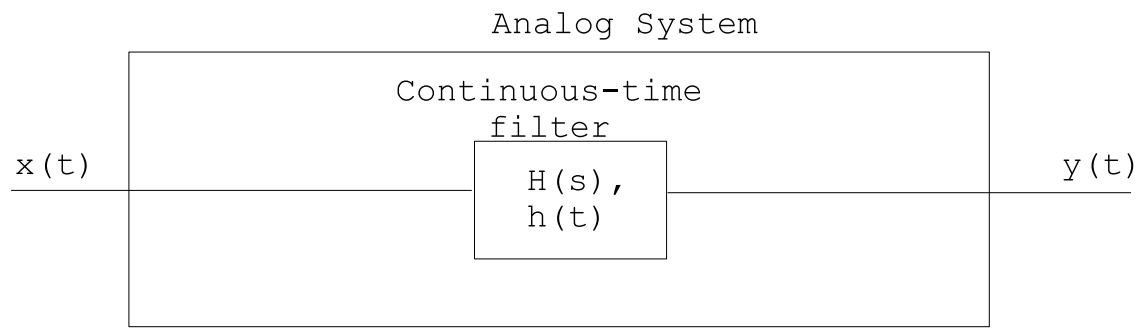


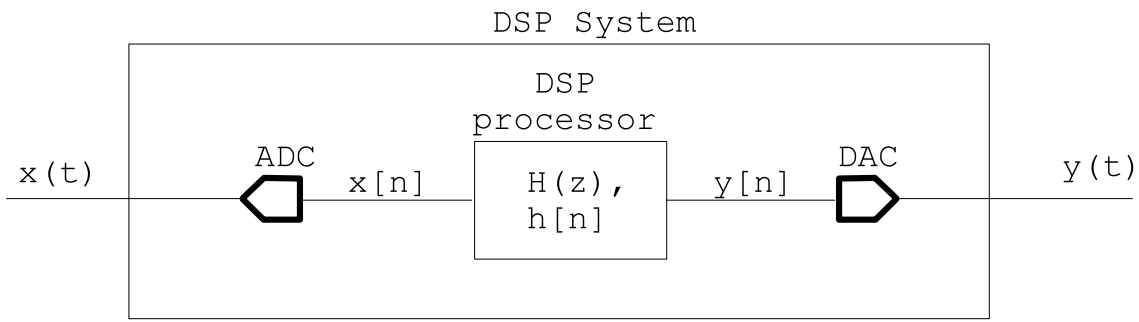
Implications of Phasor Representation

- f_0/F_s is often defined as the discrete-time frequency. This is the fraction of the circle per sample point. It is a unit-less fraction of the sampling frequency
  - Nyquist Criteria limits f_s/F_s to the range -0.5 to 0.5; frequencies outside this range are aliased into the range [-0.5,0.5]
- What about a non-integer divisor f_0/F_s, i.e. 1/4.5 ?
- What about finite precision in computing $2\pi\frac{f_0}{F_s}n$?

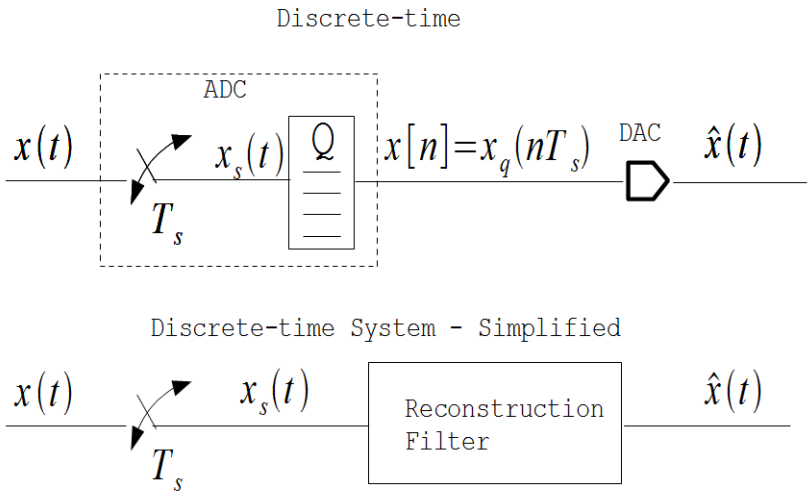
Fundamental DSP Concepts - Sampling and Reconstruction

The DSP Idea - Sampling, Digital Processing, and Reconstruction





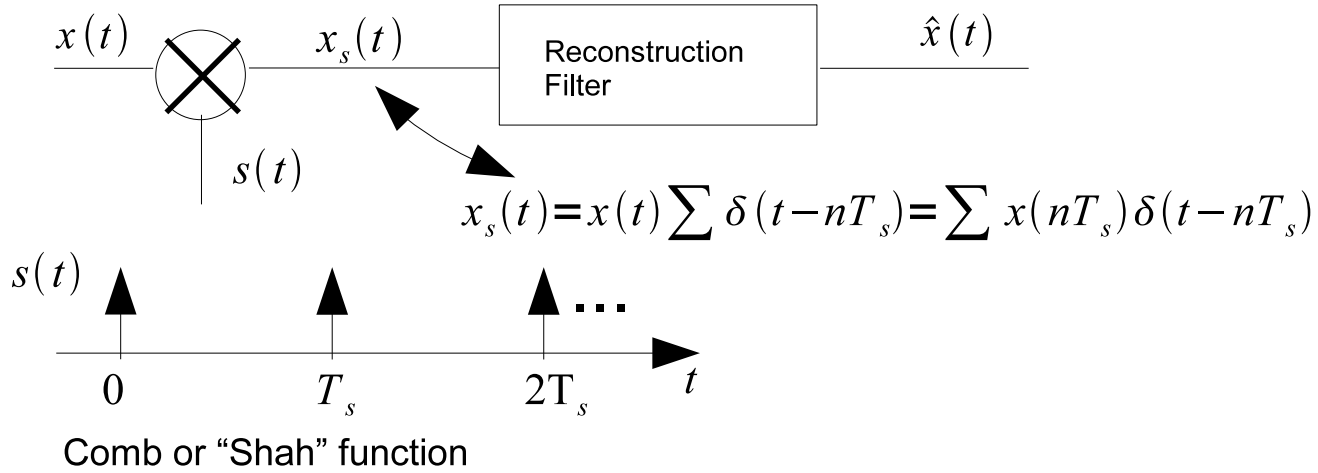
The Sampling Problem



- The sampling problem can be stated: ”Under what conditions will $\hat{x}(t) = x(t)$”. The Shannon-Whitaker Theorem will tell us that.
- We can ignore the Quantizer Q because *it can be modeled as a white noise source* which can have minimal impact with a sufficient number of bits (later). Without the Quantizer, can drop the DAC.
- Note that $x_s(t)$ is a continuous time signal but only defined at times nT_s (we don’t know what the value of $x(t)$ is between sample times. In this analysis, assume $x_s(t)$ is zero between sampling points, as would be the case in a physical circuit with a weak pulldown resistor to ground.

Sampling as Multiplication with the Comb (”Shah”) Function

- Model sampling as multiplication with the Comb or ”Shah” function
  - $x_s(t) = x(t)\sum_{n=-\infty}^{\infty}\delta(t - nT_s)$, where $\delta(t)$ is the Dirac delta function.
  - Note that $x_s(t)$ is still a continuous function, but all the information is contained within a set of discrete time points.



Properties of the Continuous-time Delta Function

- $\delta(\tau) = 0$ for $\tau \neq 0$, undefined at $\tau = 0$
- $\int_{-\epsilon}^{\epsilon} \delta(\tau) d\tau = 1$ for $\epsilon > 0$
- ”Infinite height, zero width, unit area”
- Not really a true function. Actually defined as a distribution.
- Also called the ”sifting function” : $\int_{-\infty}^{\infty} x(t)\delta(t - T_0)dt = x(T_0)$

Sampled Signal Frequency Domain Representation

- Comparing the frequency domain representation of $x_s(t)$ with $x(t)$ will tell us what reconstruction filter needed so that $\hat{x}(t) = x(t)$.
- Convolution Theorem: $x_s(t) = x(t)s(t) \Leftrightarrow X_s(f) = X(f) \otimes S(f)$, where $\otimes$ is the convolution operator
- To perform the convolution, we need the Fourier Transform $S(f)$ of the Shah Function. It can be shown that

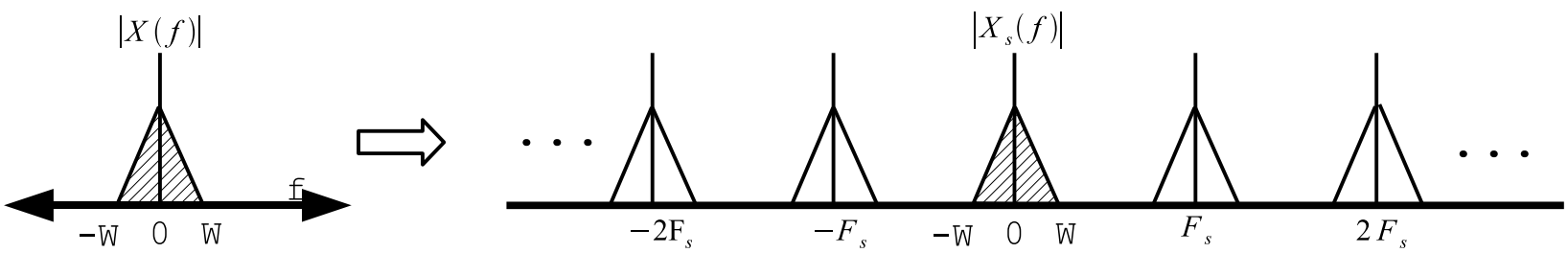
$$S(f) = F_s \sum_{n=-\infty}^{\infty} \delta(f + nF_s). \tag{1}$$

See http://www.engr.colostate.edu/ECE423/docs/ftrans_sample_fn.pdf.

Sampled Signal Frequency Spectrum

$$
\begin{aligned}
X_s(f) &= X(f) \otimes S(f) \\
&= \int_{-\infty}^{\infty} X(f) S(f + f') d(f') \\
&= F_s \int_{-\infty}^{\infty} X(f) \sum_{n=-\infty}^{\infty} \delta(f + nF_s + f') df' \\
&= F_s \sum_{n=-\infty}^{\infty} X(f + nF_s) \qquad (2)
\end{aligned}
$$

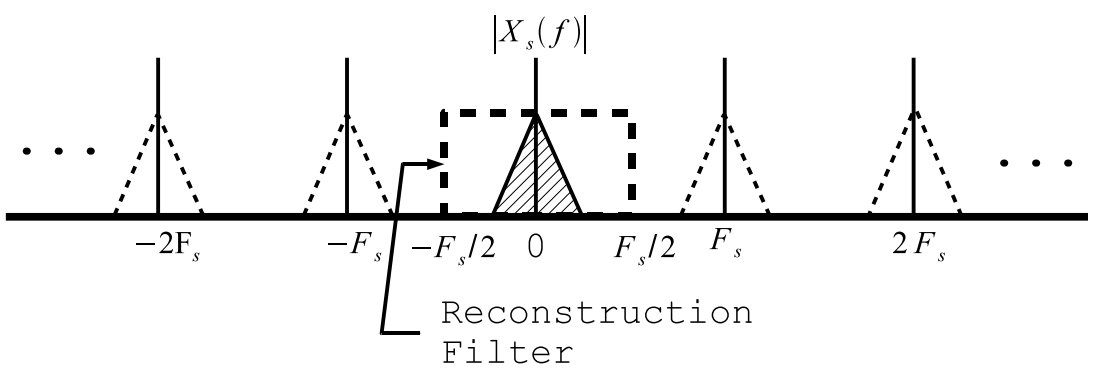
Therefore, sampling produces an infinite number of replicas ("images") of the unsampled spectrum, spaced F_s apart. The original spectrum is periodically-extended.



Reconstruction Filter Requirements

- If the images can be filtered out, the original spectrum can be recovered and therefore the original signal through inverse Fourier Transform. This is the job of the reconstruction filter.
- The ideal reconstruction filter is a brick wall filter of width $[-F_s/2, F_s/2]$ and height $\frac{1}{F_s}$, with unit area.
- The output $\hat{X}(f)$ of the sampled signal AFTER reconstruction filter is:

$$\begin{aligned} \hat{X}(f) &= \frac{1}{F_s} X_s(f) & |F| \leq F_s/2 \\ &= 0 & |F| > F_s/2 \end{aligned} \tag{3}$$



Shannon-Whitaker Theorem

Get $\hat{x}(t)$ by inverse transform of $\hat{X}(f)$, taking into account the effect of the reconstruction filter:

$$\hat{x}(t) = 1/F_s \int_{-F_s/2}^{F_s/2} X_s(f) e^{j2\pi ft} df$$

But $X_s(f)$ is periodic **in frequency** with period F_s, and therefore can be expanded as a Fourier Series:

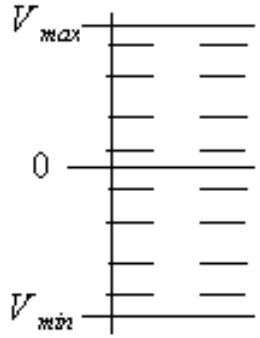
$$X_s(f) = F_s \sum_{-\infty}^{\infty} c_n e^{-j2\pi nf/F_s}$$

where $c_n = x[n] = x(nT_s)$. From this we get

$$\begin{aligned} \hat{x}(t) &= \int_{-F_s/2}^{F_s/2} [\sum_{n=-\infty}^{\infty} x(nT_s) e^{-j2\pi nf/F_s}] e^{j2\pi ft} df \\ &= \sum_{n=-\infty}^{\infty} x(nT_s) \frac{sin(\pi F_s(t - n/F_s))}{\pi F_s(t - n/F_s)} \end{aligned} \tag{4}$$

Quantization

Quantization: Irreversable Information Loss



- Divide voltage range V_R into 2^M levels. M=# of bits.
- Least Significant Bit (LSB) $\Delta = V_R/2^M$
- For DSK, $\Delta = 3.3/65536 = 50\mu V$